Hindawi Publishing Corporation
EURASIP Journal on Advances in Signal Processing
Volume 2011, Article ID 291860, 14 pages
doi:10.1155/2011/291860

Research Article

Real-Time Recognition of Percussive Sounds by a Model-Based Method

Umut Şimşekli,[1] Antti Jylhä,[2] Cumhur Erkut,[2] and A. Taylan Cemgil[1]

[1] *Department of Computer Engineering, Boğaziçi University, Bebek, 34342 İstanbul, Turkey*
[2] *Department of Signal Processing and Acoustics, Aalto University School of Science and Technology, P.O. Box 13000, 00076 Aalto, Finland*

Correspondence should be addressed to Antti Jylhä, antti.jylha@tkk.fi

Received 22 September 2010; Accepted 26 November 2010

Academic Editor: Victor Lazzarini

Interactive musical systems require real-time, low-latency, accurate, and reliable event detection and classification algorithms. In this paper, we introduce a model-based algorithm for detection of percussive events and test the algorithm on the detection and classification of different percussive sounds. We focus on tuning the algorithm for a good compromise between temporal precision, classification accuracy and low latency. The model is trained offline on different percussive sounds using the expectation maximization approach for learning spectral templates for each sound and is able to run online to detect and classify sounds from audio stream input by a Hidden Markov Model. Our results indicate that the approach is promising and applicable in design and development of interactive musical systems.

1. Introduction

Percussion instruments traditionally provide the rhythmic backbone in music. In the past few years, their automatic detection and classification has been studied in the context of music information retrieval for numerous purposes, including metrical analysis, database labeling and searches, automatic transcription, and interactive musical systems. Recently, we have witnessed an increase in the number of interactive applications built around sound detection and classification algorithms, which has been enabled by the increase in computational power of computers and appliances. Many games, toys, and educative applications exploit sophisticated signal processing as part of the interactive system. However, while the computational power enables the use of more and more complex algorithms, the applications still favor algorithmic simplicity and demand low-latency solutions in order to make the interaction fluent.

We explore the domain of realtime automatic detection and labeling of percussive sounds for interactive systems. These systems require an efficient and reliable low-level method for sound analysis. Our solution to the problem is a probabilistic model-based algorithm, which efficiently detects and labels percussive events. While many previous algorithms have been designed for a particular set of instruments or a specific application, our solution is generic; it can be retrained for any desired sound palette and fitted to numerous different applications. As examples, we consider rhythmic tutoring systems, which instruct the user to learn new rhythms, and interactive accompaniment reacting to the user's playing. We demonstrate the performance of the algorithm on different unpitched percussive sounds, including hand clapping and Turkish percussive instruments. The realization of a real-time object capable of labeling the events in an audio stream is discussed, with special attention to its applicability in interactive systems.

The paper is structured as follows. In Section 2, a review of related work related to low-level sound detection and labeling and to interactive musical systems is presented. Sections 3 and 4 summarize the proposed model-based technique. Experiments and results with different percussive sounds are discussed in the two subsequent sections, starting with hand clap sounds in Section 5 and followed by percussive instruments in Section 6. Section 7 discusses the implementation of the proposed technique as a module for real-time audio signal processing environments. Finally, in

Section 8, we discuss the potential musical applications that can make use of the method, followed by conclusions in Section 9.

2. Related Work

Sound recognition in music information retrieval has largely focused on pitched instruments, but automatic detection and labeling of unpitched percussive instruments has started to gain attention as well especially related to automatic music transcription [1]. The vast majority of related research concentrates on offline methods, as automatic transcription and audio database queries have less real-time constraints than interactive musical systems. For interactive systems, also some online methods and applications have been proposed. In this section, we summarize the state of art related to detection and labeling of unpitched percussion. For a more comprehensive review in the context of automatic music transcription, see [1].

The methods applied for sound detection and classification can roughly be grouped into methods with separate steps for onset detection and classification, and methods that combine detection and classification into a single technique. Details of implementations vary a lot, from decision tree based techniques to sophisticated probabilistic machine learning algorithms. Some techniques utilize lots of preprocessing to extract various temporal or spectral features from the audio to be used in the classification, while other methods emphasize the classification step. A review of a set of classification techniques for isolated percussion sounds has been presented by Herrera et al. [2].

The mainstream research on detection of percussive sounds has focused on a limited sound palette, that is, tracking the drum sounds in western popular music for automatic transcription or audio database annotation. Zils et al. [3] approached the problem with an analysis-by-synthesis technique to track snare drum and bass drum sounds in polyphonic music. Yoshii et al. [4] based their AdaMast algorithm for the same problem on template matching and adaptive spectral templates of the drum sounds, whereas Steelant et al. [5] applied high-dimensional feature vectors and Support Vector Machines (SVMs). SVMs were also applied by Tanghe et al. [6], whose experiment included hi-hat sounds along bass and snare drum. While some of these techniques may be efficient enough to be run in real time, no results on real time performance were reported.

The AdaMast algorithm of Yoshii et al. [4] was later applied in an intelligent instrument equalizer INTER:D [7]. The system works as an active music listening feature in a music player, enabling the user to cut or boost snare and bass drum levels or to replace their timbres with other timbres. The system has been reported to work real-time and relies on the AdaMast template matching and adaptation algorithm for low-level drum event detection from polyphonic mix.

As a deviation from western popular music instruments, Gillet and Richard [8] have examined automatic labeling of the sounds of the North Indian percussion instrument Tabla. Their approach consists of three steps. First, an envelope extraction and onset detection technique is used to segment the audio, each segment including a single stroke. After extracting rhythmic information by using beat detection techniques, they train a Gaussian Mixture Model (GMM) which approximated the power spectra of the tabla signals with four Gaussians. They finally apply a classification scheme based on a Hidden Markov Model by using the GMM parameters as feature vectors. Overall accuracy of 94% is reported for real-time transcription of tabla performances.

Paulus and Klapuri have approached the labeling of percussive events with a model-based technique on several occasions. In [9], they propose a method for detection and labeling of arbitrary percussive sounds. Their technique detects event onsets and estimates the musical meter and uses this information to determine temporal locations for feature extraction and to inform a probabilistic model. Several temporal, spectral, and cepstral features are computed. The features are clustered by a fuzzy K-means algorithm, in which the number of clusters is set manually. The temporal locations of clusters are quantized on a metrical grid. The probabilistic model provides a mapping between the feature clusters and event labels.

Recently, Paulus and Klapuri have proposed an HMM-based drum sound detection method with a joint technique for event segmentation and recognition in polyphonic music, that is, without a separate onset detection step [10]. After applying a simple sinusoid-plus-residual model to the audio signal, they use the residual signal in order to circumvent the problems imposed by polyphony. Then Mel-frequency cepstral coefficients (MFCCs) and their first-order derivatives are used as acoustic features. They apply HMMs in two different ways: in the first, they model the combinations of the target drums, whereas in the second, they use different, detector-like HMMs for each target drum. By exploring dimensionality reduction and acoustic adaptation techniques, they report 75% accuracy on polyphonic audio recordings.

Considering early real-time detectors for percussive sounds, Puckette et al. [11] have developed an object called `bonk~` for Max/MSP and Pd software platforms. `bonk~` utilizes bounded-Q transform to decompose the incoming audio signal into frequency bands, computes the signal power on each band for detection, and can be trained to recognize sounds based on template matching. The object has been used in numerous real-time applications built on Max/MSP and Pd, including, for example, interactive accompaniment [12], synchronizing an electronic sequencer with a live drummer [13], and a rhythmic tutoring system [14].

The anthropomorphic robotic percussionist Haile [12] is an example of using real-time percussion recognition for an interactive accompaniment system. The robot listens to the user's playing of a drum and according to the analysis of the performance, can join the user in different ways, for example by mimicking the user's playing or providing improvisational solo patterns. The system incorporates a music perception engine, which requires accurate low-level event detection for high-level analysis.

B-Keeper [13] is a beat-tracking system that enables rhythmic interaction between a drummer and Ableton Live.

The system tracks tempo from the kick drum and uses this to control the tempo of sequencer output. This enables live performances without a click track. The low-level event detection with `bonk~` is in this case easy, since the input signal is monophonic, that is, the kick drum sound, and requires no classification step.

Rhythmic tutoring with an interactive system has been studied by Jylhä et al. in the context of Flamenco [14]. In rhythmic tutoring, the system needs to be able to distinguish between different user inputs, for example, different types of drum strokes, in order to be able to make high-level analysis on the user's performance and give feedback to the user. While this kind of tutoring systems is still rare, they provide an interesting domain for research and applications.

For all these interactive examples, an algorithm that is capable of accurate and reliable low-level detection is crucial. In the following sections, we aim at tackling this problem with a novel, model-based algorithm that is reliable, retrainable according to needs of any given interactive system, and meeting the real-time requirements of fluent interaction.

3. Probabilistic Modeling of Percussive Sounds

In [15], Şimşekli and Cemgil presented two probabilistic models for online pitch tracking. Since the models are template based and do not heavily depend on the application, it is fairly easy to apply the models to percussive events. Instead of detecting pitch labels from streaming audio data, in this study, we adapt one of the probabilistic models to percussive event tracking and aim at inferring a predefined set of short, percussive events. Our goal is low latency without compromising the detection quality.

Our approach to this problem is model based. We construct a probabilistic generative model which relates a latent event label to the actual audio recording. The audio signal is subdivided into frames and represented by their magnitude spectrum, which is calculated with discrete Fourier transform. We define $x_{\nu,\tau}$ as the magnitude spectrum of the audio data with frequency index ν and time frame index τ, where $\nu \in \{1, 2, \ldots, F\}$ and $\tau \in \{1, 2, \ldots, T\}$.

For each time frame τ, we define an indicator variable r_τ on a discrete state space D_r, which determines the label we are interested in. In our case, D_r consists of event labels such as {hand clap, finger snap,..., attack of a conga hit, sustain of a conga hit, release of a conga hit,...}. The indicator variables r_τ are hidden since we do not observe them directly.

In our model, the main idea is that each event has a certain characteristic spectral shape which is rendered by a specific hidden *scaling* variable, v_τ. The spectral shapes that we denote as *spectral templates* are denoted by $t_{\nu,i}$. The ν index is again the frequency index and the index i indicates the event labels. Here, i takes values between 1 and I, where I is the number of different spectral templates. The scaling variables v_τ define the overall amplitude factor, by which the whole template is multiplied. An overall sketch of the model is given in Figure 1.

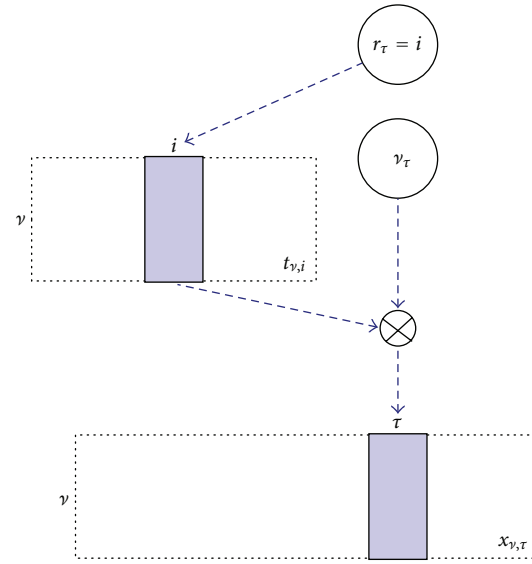

Figure 1: The block diagram of the probabilistic model. The indicator variables r_τ define the template to be used. The chosen template is multiplied by the scaling parameter v_τ in order to obtain the magnitude spectrum, $x_{\nu,\tau}$.

4. Hidden Markov Model

A Hidden Markov Model (HMM) is a well-known statistical model which is basically a Markov chain observed in noise. We utilize HMM as an inference model in our method, that is, for labeling audio frames as relating to hidden variables. Here the underlying Markov chain is not observable, therefore it is hidden. What is observable in an HMM is a stochastic process, which is assumed to be generated from the hidden Markov chain [16]. In our case r_τ form the Markov chain and $x_{\nu,i}$ are the noisy observations. The probabilistic model is defined as follows:

$$\begin{aligned} r_0 &\sim p(r_0) \\ r_\tau \mid r_{\tau-1} &\sim p(r_\tau \mid r_{\tau-1}) \\ v_\tau &\sim \mathcal{G}(v_\tau; a_v, b_v) \\ x_{\nu,\tau} \mid v_\tau, r_\tau &\sim \prod_{i=1}^{I} \mathcal{PO}\left(x_{\nu,\tau}; t_{\nu,i} v_\tau\right)^{[r_\tau = i]}. \end{aligned} \tag{1}$$

Here, $[x] = 1$ if x is true, $[x] = 0$ otherwise, and the symbols $\mathcal{G}$ and $\mathcal{PO}$ represent the Gamma and the Poisson distributions respectively, where

$$\begin{aligned} \mathcal{G}(x; a, b) &= \exp\left((a-1)\log x - bx - \log\Gamma(a) + a\log(b)\right) \\ \mathcal{PO}(y; \lambda) &= \exp\left(y\log\lambda - \lambda - \log\Gamma(y+1)\right), \end{aligned} \tag{2}$$

where Γ is the Gamma function.

In some recent work on music information retrieval, Poisson observation model was used in the Bayesian non-negative matrix factorization (NMF) models [17]. Since our

probabilistic models are similar to NMF models, we choose the Poisson distribution as the observation model. We also choose Gamma prior on v_τ to preserve conjugacy and make use of the scaling property of the Gamma distribution. The conjugate prior grants us an analytic closed-form solution to the posterior; otherwise numerical integration would be required.

Moreover, we choose Markovian prior on the indicator variables, r_τ which means r_τ depends only on $r_{\tau-1}$. We use one single state in order to represent silence and nonresonant events (e.g., hand claps) and we use three states to represent resonant events (e.g., hand drums): one state for the attack part, one for the sustain part, and one for the release part. Figure 2 shows the graphical model of the three-state HMM, which provides an intuitive way to represent the conditional independence structure of the probabilistic model. In the model, the nodes correspond to probability distributions of model variables, and edges to their conditional dependencies. The joint distribution can be rewritten by making use of the directed acyclic graph

$$\begin{aligned} &p(r_{1:T}, v_{1:T}, x_{1:F,1:T}) \\ &= \prod_{\tau=1}^{T} p(r_\tau \mid \mathrm{pa}(r_\tau))\, p(v_\tau \mid \mathrm{pa}(v_\tau)) \\ &\quad \times \prod_{\nu=1}^{F} p(x_{\nu,\tau} \mid \mathrm{pa}(x_{\nu,\tau})), \end{aligned} \tag{3}$$

where $\mathrm{pa}(\chi)$ denotes the *parent nodes* of χ. Figure 3 shows the Markovian structure of the indicator variables in more detail.

The observation model assumes that the subsequent frames are conditionally independent from each other given the latent indicators r_τ. Hence, to conform with this assumption, we calculate the spectra x_τ on nonoverlapping frames. In practice, one could also compute the spectrum using overlapping frames, but then the conditional independence assumption would not be exactly valid.

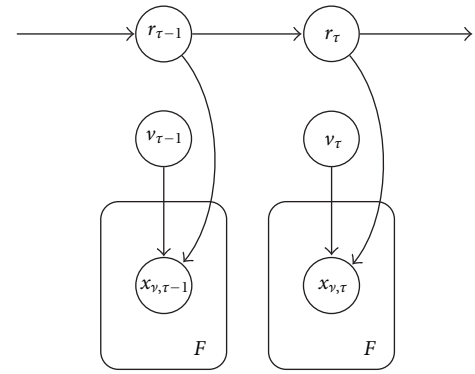

Figure 2: Graphical model of the Hidden Markov Model. This graph visualizes the conditional independence structure between the random variables and allows the joint distribution to be rewritten by utilizing (3). In the model, the nodes correspond to probability distributions of model variables, and edges to their conditional dependencies. Note that we use the plate notation for the observed variables where F distinct nodes (i.e., $x_{\nu,i}$ where $\nu \in \{1,\ldots,F\}$) are grouped and represented as a single node in the graphical model. In this case, F is the number or frequency bins.

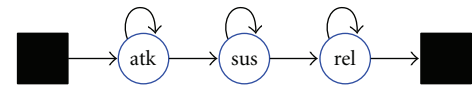

Figure 3: The state transition diagram of the indicator variable r_τ. This figure visualizes the possible states that r_τ can be in given the previous state $r_{\tau-1}$. Here atk, sus, and rel refer to the attack, sustain, and release parts of an event, respectively. The first black square can be either the silence or a release state. Similarly the second black square can be either a silence or an attack state. For example, given that $r_{\tau-1}$ is the release state of event E_i, then r_τ can be either the same state as $r_{\tau-1}$ (i.e., E_i resumes) or it can be the silence state (i.e., E_i ends) or it can be the attack state of a percussive event which is defined in the model (i.e., a new event starts). Similarly, r_τ cannot be a sustain state if given that $r_{\tau-1}$ is a release state.

4.1. Inference. Inference is a fundamental issue in probabilistic modeling. It is characterized by the question "what can be the hidden variables as we have some observations?" [16]. For online processing, we are interested in the computation of the following posterior quantity, also known as the filtering density (Note that we use MATLAB's colon operator syntax in which $(1:T)$ is equivalent to $[1,2,3,\ldots,T]$ and $x_{1:T} = \{x_1, x_2, \ldots, x_T\}$)

$$p(r_\tau \mid x_{1:F,1:\tau}), \tag{4}$$

that is, given an observation, what is the probability of the class labels for that observation? Similarly, we can also compute the most likely label trajectory over time given all the observations

$$r^*_{1:T} = \underset{r_{1:T}}{\mathrm{argmax}}\; p(r_{1:T} \mid x_{1:F,1:T}). \tag{5}$$

This latter quantity requires that we accumulate all data and process in a batch fashion. There are also other quantities, called "fixed lag smoothers" that compromise between those two extremes. For example, at time τ, we can compute

$$\begin{aligned} &p(r_\tau \mid x_{1:F,1:\tau+L}) \\ &r^*_\tau = \underset{r_\tau}{\mathrm{argmax}}\; p(r_{1:\tau+L} \mid x_{1:F,1:\tau+L}), \end{aligned} \tag{6}$$

where L is a specified lag, and it determines the tradeoff between the accuracy and the latency. By accumulating a few observations from the future, the detection at a specific frame can be eventually improved by introducing a slight latency. This way, we do not need to accumulate all data before inference, while still obtaining a smoothed estimate for the hidden variable posterior.

In order to simplify the inference scheme, we can integrate out analytically the scaling variables v_τ. It is easy to check that once we do this, provided the templates $t_{\nu,i}$ are

already known, the model reduces to a standard HMM with a Compound Poisson observation model as shown below

$$\begin{aligned} & p(x_{1:F,\tau} \mid r_\tau = i) \\ & = \int dv_\tau \exp\left(\sum_{\nu=1}^{F} \log \mathcal{PO}(x_{\nu,\tau}; v_\tau t_{\nu,i}) + \log \mathcal{G}(v_\tau; a_\nu, b_\nu)\right) \\ & = \exp\left(\log\Gamma\left(\sum_{\nu=1}^{F} x_{\nu,\tau} + a_\nu\right) + \sum_{\nu=1}^{F} x_{\nu,\tau} t_{\nu,i} - \sum_{\nu=1}^{F} \log\Gamma(x_{\nu,\tau}+1)\right. \\ & \quad \left. - \left(\sum_{\nu=1}^{F} x_{\nu,\tau} + a_\nu\right) \log\left(\sum_{\nu=1}^{F} t_{\nu,i} + b_\nu\right) + a_\nu \log b_\nu - \log\Gamma(a_\nu)\right). \end{aligned} \tag{7}$$

Since we have standard HMM from now on, we can run the well-known forward algorithm in order to compute the filtering density or fixed-lag versions with a few backward steps. Also we can estimate the most probable state sequence by running the Viterbi algorithm. A benefit of having a standard HMM is that the inference algorithm can be made to run very fast. This lets the inference scheme to be implemented in realtime without any approximation [18].

4.2. Training and Parameter Learning. Since we have constructed our inference algorithms with the assumption of the spectral templates $t_{\nu,i}$ to be known, they have to be learned at the beginning. In this study, we utilize the Expectation-Maximization (EM) algorithm for this purpose. This algorithm iteratively maximizes the log-likelihood as follows:

E-step:

$$q(r_{1:T}, v_{1:T})^{(n)} = p\left(r_{1:T}, v_{1:T} \mid x_{1:F,1:T}, t_{1:F,1:I}^{(n-1)},\right) \tag{8}$$

M-step:

$$t_{1:F,1:I}^{(n)} = \underset{t_{1:F,1:I}}{\operatorname{argmax}} \left\langle \log p(r_{1:T}, v_{1:T}, x_{1:F,1:T} \mid t_{1:F,1:I}^{(n)}) \right\rangle_{q(r_{1:T}, v_{1:T})^{(n)}}, \tag{9}$$

where $\langle f(x)\rangle_{p(x)} = \int p(x) f(x) dx$ is the expectation of the function $f(x)$ with respect to $p(x)$.

In the E-step, we compute the posterior distributions of r_τ and v_τ. These quantities can be computed via the methods which we described in Subsection 4.1. In the M-step, we aim to find the $t_{\nu,i}$ that maximize the likelihood. Maximization over $t_{\nu,i}$ yields the following fixed-point equation:

$$t_{\nu,i}^{(n)} = \frac{\sum_{\tau=1}^{T} \langle [r_\tau = i] \rangle^{(n)} x_{\nu,\tau}}{\sum_{\tau=1}^{T} \langle [r_\tau = i] v_\tau \rangle^{(n)}}. \tag{10}$$

Intuitively, we can interpret this result as the weighted average of the normalized audio spectra with respect to v_τ.

5. Experiment 1: Hand Clapping

We have run two sets of experiments to test the applicability of the presented technique for identification of percussive sounds. In the first experiment, we tested the method with different types of hand clapping to identify the clapping type. With this experiment, we tested the performance of the algorithm in the context of very similar, non-resonant sounds. In the second experiment, we recorded sounds of Turkish percussion instruments and applied the method for tracking the percussion events in simple rhythmic patterns. With the second experiment, in addition to testing the detection and labeling accuracy, we specifically looked at the compromise between latency and performance. In this section, we discuss the experiment with hand clapping sounds, and in the following section with the percussive instruments.

Hand claps are a relatively primitive conveyor of sonic information, yet they are widely applied for different purposes [19]. In different cultures hand claps are used in a musical context, and we are used to give feedback of a performance by applause [20], by indicating different levels of enthusiasm to the performers. Hand claps are also an essential part of Flamenco music, in which rhythmic patterns of soft and sharp claps are used as an accompaniment [14]. Furthermore, hand clapping sounds can be used for interacting with musical systems and computer interfaces [21].

As percussive events, hand claps of an individual decay very rapidly and are very short in time. In anechoic conditions, a hand clap sound lasts typically around 5 ms, and it is difficult to pinpoint any systematic differences between different kinds of hand claps only by listening. This performance improves significantly in more casual indoors listening situations, where the room becomes an extended resonator. In fact, our perception seems to process simultaneously both the source characteristics of hand claps and the room characteristics, supporting involvement of special mechanisms during the perception of the self-produced sound [22]. Specifically, according to Repp [19], human observers are able to identify their own hand clapping and deduce the hand configuration of a clapper with a reasonable accuracy in a single-clapper setting by listening to the hand clap sound.

Based on spectral analysis, Repp has proposed eight different hand configurations which have audible differences. These clapping modes were presented in detail in [20, 23]; here we provide only a short description. In P (parallel) modes, the hands are kept parallel and flat, whereas in A (angle) modes they form a right angle with a natural curvature. The following numerical index indicates the position of the right hand relative to the left hand: from palm-to-palm (1) to fingers-to-palm (3), (2) corresponding to about the midpoint. Finally, the curvature of hands varies in A1 mode to result a flat (A1−) or a very cupped (A1+) configuration, compared to A1.

To evaluate the real-time event detection technique with hand clapping sounds, we have conducted several experiments. The experiment design focused on obtaining audio data systematically for inferring the hand configuration and the identity of an individual by the technique proposed in Section 4, using a single-state HMM. We evaluated the technique with hand clap signals recorded in an anechoic chamber and in a quiet room.

5.1. Hand Claps in an Anechoic Room. Three subjects (one female and two male) participated in this experiment. They were given verbal, as well as pictorial description about the clapping modes, and instructed to produce five claps in each mode. The recordings were segmented into three different files, and high pass was filtered with a fifth-order Chebyshev Type I filter with a cutoff frequency of 100 Hz to remove the structural vibration of the anechoic chamber at low frequencies. The phase delay of the filter has been compensated by zero-phase filtering using the `filtfilt` MATLAB function.

5.2. Hand Claps in a Quiet Room. We recorded sequences of hand claps in an ITU-R BS.116 standard listening room, with reverberation time of 0.3 s. Two male subjects A and B performed 20 claps of each type. The recordings were segmented into 16 separate files, each containing one clapping mode of one subject.

5.3. Training and Classification Procedure. In the anechoic case, we trained the system with three claps from each user in each mode, and tested it on the whole data set. Instead of the three-state HMM, we applied a -state model, since clapping sounds do not incorporate sustain. For the quiet room recordings, we randomly divided the recorded data into training and test data. We divided each of the audio files into four equally long segments and randomly chose two of these segments for training and two for testing the classification performance of the algorithm.

As explained in Section 4, the algorithm learns templates of different classes in the training phase and provides a posterior distribution of audio frames corresponding to each spectral template in the test phase. In the anechoic case, each mode of each subject was provided as classes, totaling 24 templates for clapping modes, and an additional one for silence. We have also tested using one template for each mode of all subjects and silence (i.e., 9 templates). From the posterior distribution, we classified detected nonsilent sounds into one of the trained classes based on the classification results of all non-silent frames that met the condition $p(r_\tau \mid \cdot) > \varepsilon$, where ε is a threshold for the decision. The value of $\varepsilon = 0.5$ was used in the tests. Note that the chance level classification of 8 modes is 0.125.

In the quiet room dataset, this technique caused difficulties in reverberant conditions because some clapping modes were prone to being confused with room reverberation. Therefore, we have added a post processing step, in which we only used the first frame of a non-silent segment and skipped the following 10 frames (23.2 ms) to overcome the reverberation effect. The thresholding was otherwise similar to the anechoic case.

5.4. Results. In both anechoic and quiet conditions, the algorithm performs well as a detector of clapping, with event frames being classified as non-silent nearly 100% of the time. We have done clapping detection previously with the Pure Data external `bonk~` [11], for which the clap detection rate in similar conditions has been around 85% (without optimizing its detection parameters).

In anechoic settings, we tested how well the algorithm identifies the eight clapping modes of three clappers. The results are presented in Table 1, for the offline (top) and real-time (bottom) operation, where the rows are target class labels and columns give the percentage of classifying that target as each of the eight classes. The overall accuracy of the offline simulation is 66% and the real-time simulation with forward-backward smoothing yields 61%. If a single template for each mode is used, the accuracy reduces to 53% (offline) and 49% (online), which indicates that there are differences between subjects producing these modes. We observe that the real-time case reduces the latency with a slight effect on accuracy. In the following, we present only the real-time results.

We also tested how well the algorithm identifies the eight clapping modes of two clappers in a quiet room setting. The results are presented in Table 2. The overall performance of using the algorithm for classification is 65% for subject A and 41% for subject B. While these rates may seem low, it should be noted that the algorithm performs way beyond chance level (12.5%) in all cases and that some hand configurations are rather similar with respect to the sound output. All the hand claps were detected as hand clap events.

Next, we have trained the system with one subject's clapping and tested it with the other subject's clapping. The overall correct classification performance is around 30%. The reason for this not being lower is mostly due to the fact that mode A1+ is so prominently different from other modes that even with two different clappers that mode is classified correctly most of the time. On the other hand, since this performance clearly indicates that a system trained by one person cannot be reliably used by another one, we find these results interesting for customizing a system to work properly only with one person's clapping.

These results indicate that, while our algorithm can be trained to identify different modes of clapping, the division into eight modes is likely suboptimal for reliable identification of the modes. We can hypothesize on more robust groupings of modes for example, by grouping the modes into just A and P modes, or into number modes so that all "1" modes (palm on palm) make one class, all "2" modes another class, and "3" modes (fingers on palm) a third class. In addition, since the curvature of A1+ mode leads to pronouncedly low resonances and good classification results, this mode could be its own class in any grouping.

6. Experiment 2: Percussion Instrument Detection and Identification

We have run a second set of experiments with two Turkish percussion instruments, namely the Darbuka (goblet drum) and the Bendir (frame drum). Both drums are membranophones, but with major differences in the sound. The sound of the darbuka is typically shorter and inherently more damped than that of the bendir, which has a larger and more elastic membrane. The playing style of the instruments is similar, however, in that both instruments are often used to produce "düm" and "tek" strokes. The first is a shot with fingers hitting the membrane near the rim but not on the

Table 1: Relative classification results for the eight modes of three subjects in an anechoic chamber, the offline inference results (top, cf. (5)) and the online inference results (bottom, cf. (4)). Row labels indicate target class and columns the percentage of classifying the claps of target class to all classes. Note that, in some cases such as P1 in the top part, the rows do not add to 100% because of the rounding performed in the presentation of the results in a tabular form.

Offline, overall accuracy 66%								
Label	P1	P2	P3	A1	A2	A3	A1+	A1−
P1	**81**	13	0	7	0	0	0	0
P2	15	**77**	0	0	0	0	0	8
P3	7	7	**60**	7	0	7	0	13
A1	0	0	0	**73**	0	7	7	13
A2	7	7	0	7	**60**	0	7	13
A3	0	0	20	0	7	**47**	7	20
A1+	0	0	0	13	7	0	**80**	0
A1−	0	13	7	7	0	7	13	**53**
Real-time, overall accuracy 61%								
Label	P1	P2	P3	A1	A2	A3	A1+	A1−
P1	**69**	19	0	6	6	0	0	0
P2	15	**69**	0	0	0	8	0	8
P3	7	13	**53**	7	0	7	0	0
A1	0	0	0	**67**	0	0	20	13
A2	7	7	0	7	**60**	0	7	13
A3	0	0	20	0	7	**40**	13	20
A1+	0	0	0	13	7	7	**73**	0
A1−	0	13	7	7	7	7	7	**53**

Table 2: Relative classification results for the eight modes of two subjects. Row labels indicate target class and columns the percentage of classifying the claps of target class to all classes. Overall classification rate is 65% for subject A and 41% for subject B.

Subject A								
Label	P1	P2	P3	A1	A2	A3	A1+	A1−
P1	**56**	0	0	22	22	0	0	0
P2	0	**78**	0	0	11	11	0	0
P3	0	40	**20**	0	30	10	0	0
A1	0	0	0	**90**	10	0	0	0
A2	0	0	20	10	**60**	10	0	0
A3	0	10	0	20	0	**70**	0	0
A1+	0	0	0	0	0	0	**100**	0
A1−	0	10	0	40	0	0	0	**50**
Subject B								
Label	P1	P2	P3	A1	A2	A3	A1+	A1−
P1	**56**	0	11	0	22	0	0	11
P2	0	**56**	11	0	22	11	0	0
P3	0	0	**22**	0	11	67	0	0
A1	11	0	0	**33**	33	0	11	11
A2	0	0	0	0	**20**	30	0	50
A3	0	0	0	0	56	**44**	0	0
A1+	0	0	0	30	10	0	**60**	0
A1−	11	11	0	33	11	0	0	**33**

rim, whereas the latter is produced by hitting the rim with a finger. Therefore, "düm" typically resonates longer on lower frequencies than "tek." There are other types of strokes, too, but in this study we focus on these two types. Exemplary spectrograms of düm and tek sounds on both instruments can be seen in Figure 4. From these plots the difference in the duration of the sound between the two instruments stands out, with the bendir sounds resonating 3-4 times longer than the darbuka sounds. Also, we notice differences in the spectral characteristics between düm and tek strokes, the latter ones resonating more on higher frequencies. The bendir sound is in general darker.

We recorded the playing of both instruments with an expert percussionist in both an anechoic chamber and normal room conditions with a portable recorder (ZOOM H4). For the darbuka, we recorded 40 isolated düm strokes, 40 isolated tek strokes, and a simple rhythmic pattern with one düm followed by two teks. For the bendir, we did the same, but we separately recorded two types of isolated tek strokes, one with the index finger hitting the rim, and one with the ring finger hitting the rim, because in many practical cases the player can alternate between the two fingers, especially with two subsequent tek strokes in the pattern. For the Bendir, we also recorded the düm-tek-tek pattern and in addition a simpler düm-tek pattern.

6.1. Test Procedure. We ran several tests on the algorithm with the percussion instrument sounds. We used the isolated recordings of darbuka and bendir as training data for learning the templates, and the rhythmic patterns for testing the detection and classification performance. For test runs, the recorded patterns were manually labeled.

Considering that the attack part of a percussive event lasts approximately 2-3 ms, we tried to find a short frame size where the frames could capture the whole attack part of a percussive event. We then applied frames of 128 samples (about 3 ms at 44100 Hz) to compute STFT. Since the successive audio frames are assumed to be distributed conditionally independent from each other in our probabilistic model, we used nonoverlapping frames in the calculation of the spectra. Using overlapping frames requires a slightly different probabilistic model (a so-called Gabor Regression model [24]), which is not investigated in this study.

To eliminate the detection of a "double attack" due to ambiguities in the sound around the transition from attack to sustain, we added a post-processing step similar to that used in the hand clap experiment, disregarding any detected attacks after a fixed period of time following each detected attack. While this may result in missed detections in the case of two very rapidly played strokes, it is not a problem in this monophonic case.

With these tests, we aimed at examining the effect of the fixed lag amount in the HMM procedure (see Section 4) on algorithmic performance. We defined a correctly detected and classified event based on the attack segment of the sound, because it captures best the sounding differences between different strokes. The sustain and release parts were still used in the inference procedure to inform the model, but the overall classification decision was made on the attack segment.

As metrics for the performance, we utilized the precision rate, the recall rate, and the latency. These are defined as

$$\begin{aligned}\text{precision} &= \frac{\#\text{ correctly recognized events}}{\#\text{ events recognized by the method}},\\ \text{recall} &= \frac{\#\text{ correctly recognized events}}{\#\text{ true events}},\\ \text{latency} &= \text{estimated on set time} - \text{true on set time}.\end{aligned} \tag{11}$$

We separately trained and tested the model with recordings of one instrument from one room, for example, with darbuka in anechoic conditions. That is, we exclusively focused on the classification of düm strokes and tek strokes under fixed conditions with fixed instruments. This resulted in four test cases (darbuka anechoic, darbuka normal room, bendir anechoic, and bendir normal room).

6.2. Overall Results. For labeling the events, the overall performance with respect to the HMM lag is presented in Figure 5. In the figure, the dashed line indicates results with the offline algorithm used as a baseline.

We notice from the results that in general a lag of around 20–30 ms seems a reasonable compromise between latency and accuracy, with the overall precision of over 85% and recall of over 80%. With longer lags, the precision and recall improve slightly, but with the expense of adding more overall latency. Note that the overall latency is the sum of the computational latency (on y-axis in the figure) and the fixed lag used in smoothing. We return to the latency discussion and its implications for real-time systems in Section 8.

An interesting aspect is that in some cases the fixed-lag smoothed results (solid lines) are observed to outperform the offline results (dashed lines). This is a typical model mismatch situation which shows that the HMM cannot perfectly model the percussive audio. As a natural result of this, increasing the size of the accumulated data does not yield better results.

6.3. Comparison between Different Room Conditions and Instruments. We then compared the same metrics between different room conditions. As can be seen from Figure 6, the results are better for anechoic conditions, which is hardly a surprise given that there is no reverberation or other noise corrupting the sounds. With the lag of 20 ms, precision and recall rates are around 85% in both cases.

In Figure 7, we compare the results between the two instruments. We notice that the results for bendir are significantly better in terms of precision and recall. This is clear especially in normal room conditions. We did not anticipate this, as the bendir is significantly longersounding and in the rhythmic patterns its decaying sound from previous hits is practically always overlapping with the attack segment of the following sounds. In contrast, the darbuka sound decays faster and does not introduce this masking. An explanation

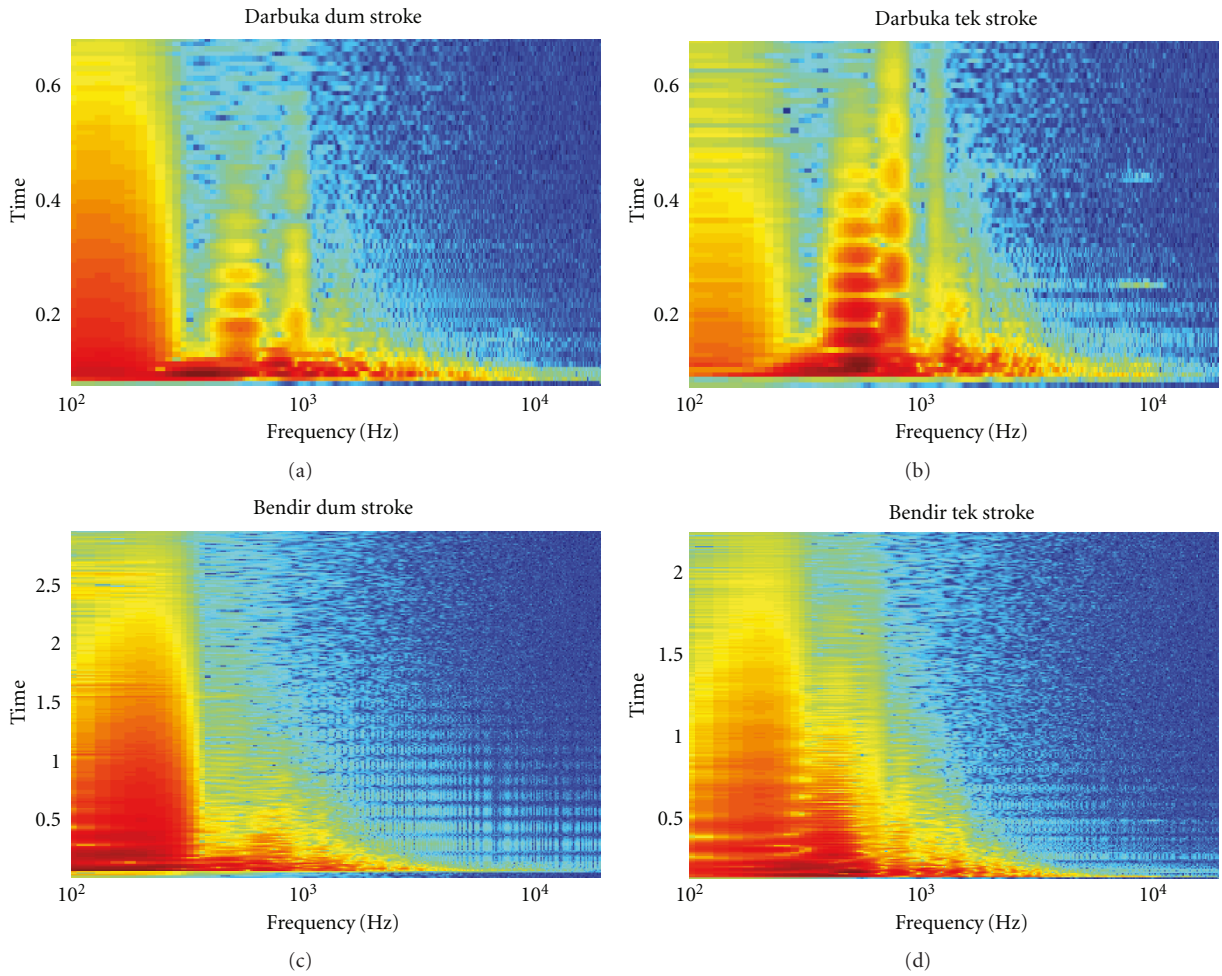

Figure 4: Spectrograms of exemplary düm and tek strokes on the Darbuka (top) and the Bendir (bottom).

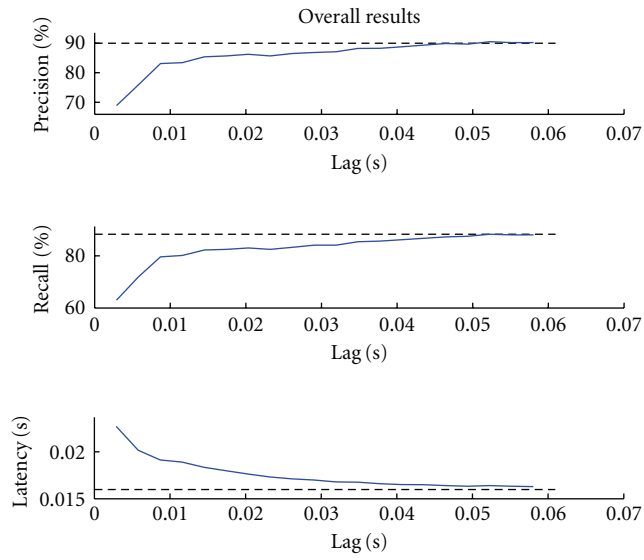

Figure 5: Overall results for the two percussion instruments. The dashed line indicates offline performance. The curves show how the performance depends on the amount of lag used in the HMM procedure. The overall input-output latency of the method is determined by the sum of lag and latency in the bottom plot.

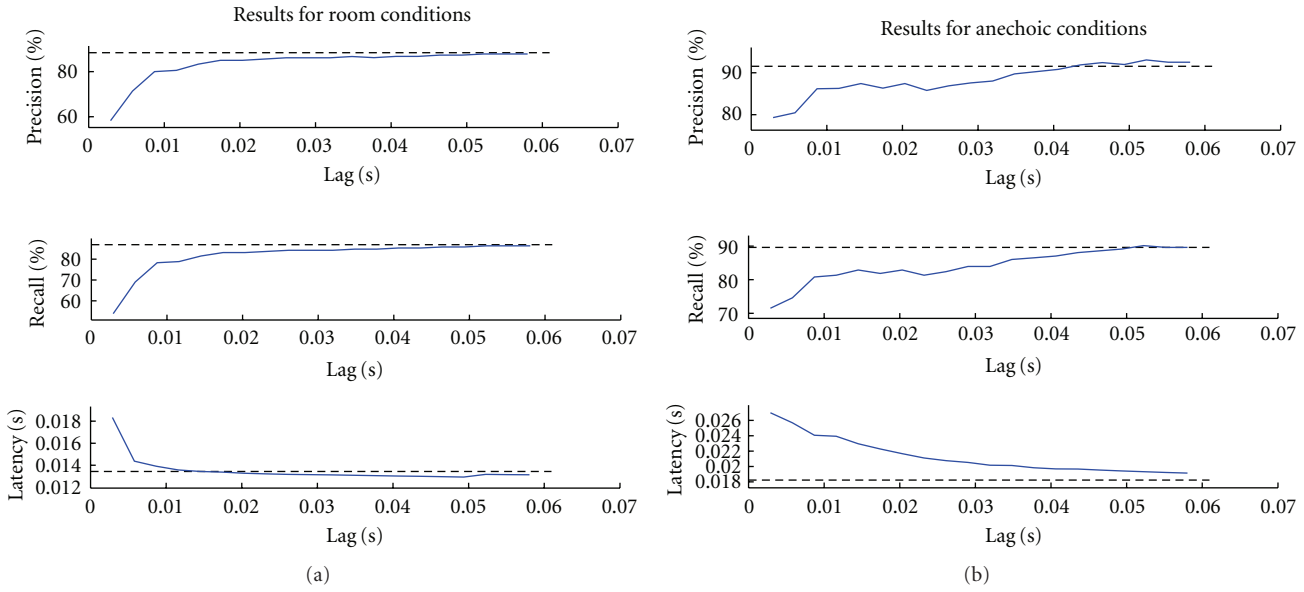

Figure 6: Comparison of algorithm performance between (a) normal room conditions and (b) anechoic conditions. The dashed line indicates offline performance.

for the results is that the spectral difference between the düm stroke and tek stroke is not as obvious with the darbuka.

6.4. Auditory Comparison. To give a concrete example of the results in practice, we used the algorithm output (event times and labels) as control data to synthesize an audible pattern for comparison with the original recorded pattern by triggering recorded samples according to the automatic recognition. These samples, along with excerpts of the original recordings, are available at http://www.acoustics.hut.fi/~ajylha/percussion/. By listening to the samples, it is clear that the results are better for the bendir. It is noteworthy that the original temporal structure is well preserved.

7. Percussion Detection Module for Interactive Systems

As indicated in Section 4.1, we can run the inference scheme in realtime with a standard HMM. In this section, we report two real-time implementations of our algorithm. They both rely on converting (7) in a suitable form for computation.

The first one, presented in Section 7.1, is realized as a prototype on MATLAB using the Data Acquisition Toolbox. We have then ported the HMM and the inference algorithm to C++ by using the `boost` libraries (http://www.boost.org/) for special distributions, matrix and vector calculations. We have finally used the `flext` C++ development layer (http://puredata.info/Members/thomas/flext/) for cross compiling our algorithm as a plugin for popular real-time audio signal processing environments Pure Data (http://puredata.info/) and Max/MSP (http://cycling74.com/products/maxmspjitter/). This implementation is presented in Section 7.2.

Both real-time implementations operate well with the FFT size of 128 under the sampling frequency 44.1 kHz, and the buffer sizes corresponding to the optimal latency range indicated in Section 6 (20–30 ms). They give the same results as the MATLAB simulations we have used in testing.

7.1. Real-Time Prototype. The Data Acquisition Toolbox consists of a shared library, dynamically-linked interfaces to specific hardware, and collection of functions, for example, a pointer to the callback function that fills the preallocated buffer shared by the Data Acquisition Engine and the MATLAB platform. The toolbox currently runs only on 32-bit MS Windows and interfaces with the sound drivers natively supported by this operating system (i.e., no low-latency ASIO drivers). Despite of this, we have obtained good results with the same portable recorder (ZOOM H4) we have previously used in measurements. This device also functions as a low-latency audio interface.

After initializing and setting up this device to operate with chunks of 10 blocks (of size 128 samples, corresponding to about 30 ms), we have provided the same code block we have used in training (see (7)) within the callback function. In addition, the callback dynamically plots the acquired data and prompts the recognized events to the command line.

The toolbox would return an error if the number of samples acquired or queued would exceed the allocated memory. The reason for excess queuing would then be our processing time, that is, our calculations could not be completed within the time between two chunks. However, even the chunk of 5 blocks (about 15 ms) did not indicate an error in our tests, and the results were as reliable as the offline operation.

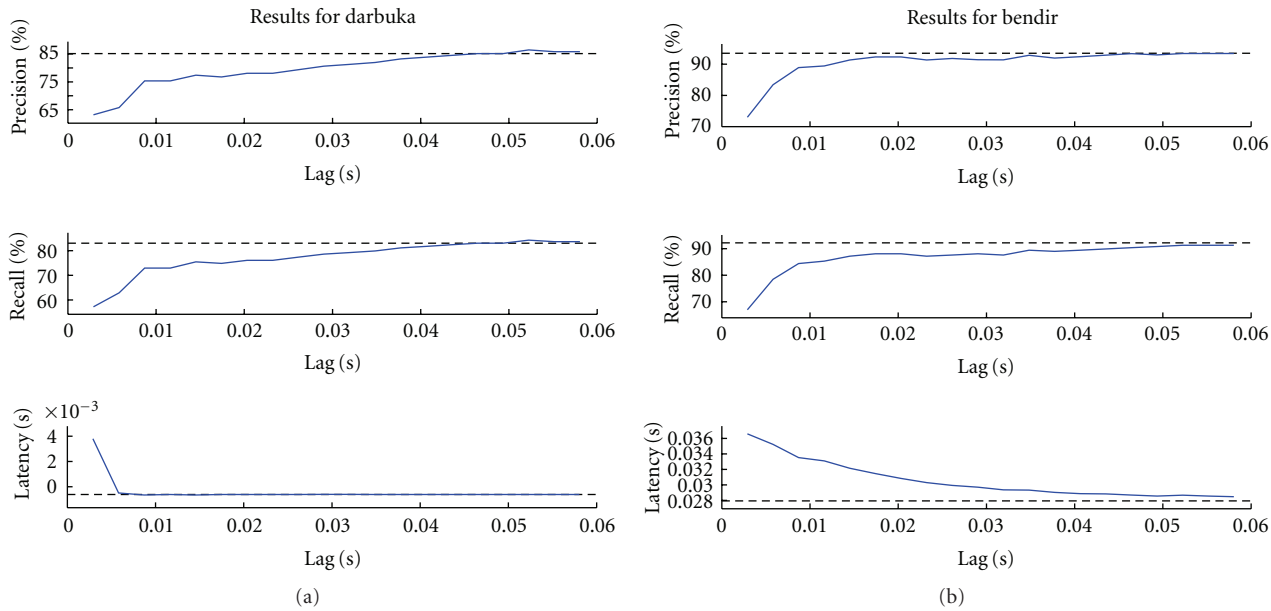

Figure 7: Comparison of algorithm performance between (a) darbuka and (b) bendir. The dashed line indicates offline performance.

7.2. Real-Time Implementation. Encouraged by the results of the real-time prototype, we have implemented the essential HMM operations as a C++ class with the help of the `boost` libraries (`HMM`), specialized these operations for our specific problem according to our algorithm (`PercussionHMM`), and wrapped these classes within the cross-platform plugins using the `flext` layer (`aed_hmm`). The resulting structure is depicted in Figure 8 as a class diagram, one of the 14 models of the *Universal Modeling Language* (UML 2.2).

The base class `HMM` declares the private variables (attributes) `A` and `I`, corresponding to the state transition matrix and number of states, respectively.

The subclass `PercussionHmm` specializes the base class `HMM` to initiate an `HMM` (`pt`) and operate it on spectral template matrices `t`, each having `I` templates and `F` frequency bins (65 in our case). `a_v` and `b_v` are the scale and shape parameters of the Gamma prior, respectively. The other private variables are for storing the intermediate results and decreasing the computational burden. In the public member functions, besides the accessor methods and the usual HMM functions for prediction, estimation, and filtering, we have methods for loading the templates and initializing the HMM either parametrically or by a file.

Finally, the wrapper `aed_hmm` contains a `PercussionHMM` object `pt` and the previous estimate. It constructs a plug-in with four inlets and one outlet. The outlet is the detected onset. The two inlets are real and imaginary parts of the FFT of the incoming audio signal, and the other two set the variance and scaling of the spectral slices in comparison to the templates. These instantaneous variance and scaling parameters are also the private variables `f_Av` and `f_scaling`, respectively, and their changes are processed via callbacks. The main processing happens within the protected virtual function `m_signal`, which calls the filtering and prediction methods of `pt`.

As in the prototype, the plug-in is observed to work smoothly and efficiently on the host platforms with optimal block sizes.

8. Applications

As the results indicate, the presented technique works with good accuracy for percussive sound detection and labeling. It also runs in real time, so it is operable in real-time interactive systems requiring reliable recognition of percussive sounds.

We acknowledge that the constant lag required by the HMM procedure would be an issue in systems requiring immediate response to the detection, such as directly triggering synthetic sounds with the detected events. The lag in such systems would be perceivable between user input and system output, at least for expert musicians, although there are techniques that can be applied in delayed decision-making situations. However, there are a number of real-time applications that do not require such a direct response. Here, we consider two specific cases, which are rhythmic tutoring and interactive musical accompaniment.

Previously, Jylhä et al. [14] have presented an interactive Flamenco tutoring system, which aims at teaching Flamenco hand clapping patterns to a novice user. This virtual tutor provides synthetic exemplary clapping of the pattern, listens to the user's clapping sounds, and gives feedback on the user's performance and learning. This feedback is based on the analysis of the user's performed accentuation (soft and loud claps) and clapping tempo, derived from the time instances of detected events. Here, a small amount of latency is perfectly tolerable, as the tempo estimation is not dependent on maximally fast response. Furthermore, since the system operates with rhythmic patterns, a constant lag is easily compensated by the virtual tutor. This is achieved

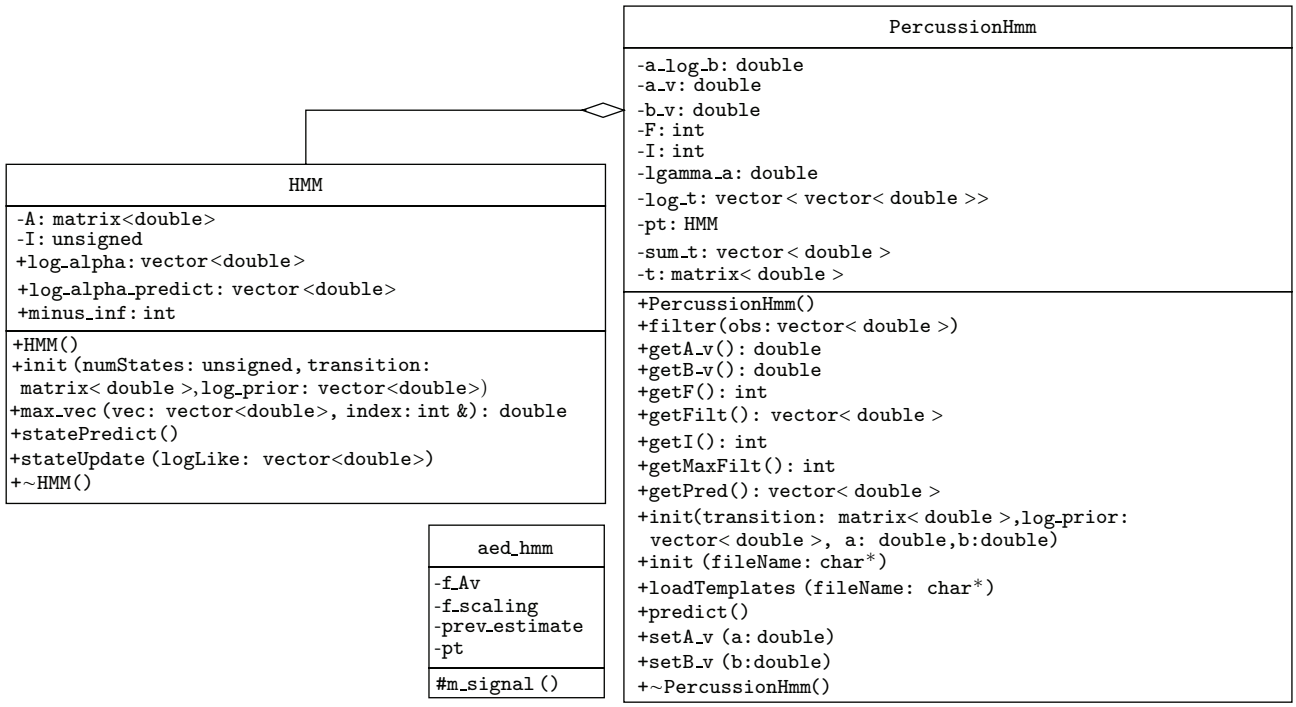

Figure 8: The UML class diagram of the real-time implementation.

by anticipating the temporal location of the next clapping event, which can be done by simply computing the difference between the estimated average interval between subsequent clapping events and the latency. This way, it is possible to keep the tutor's audio feedback in sync with the user's clapping. However, even this is not often necessary, as our previous explorations indicate that the user naturally syncs with the tutor's clapping anyway.

With regards to virtual tutoring, more sophisticated systems for different instruments and musical styles are enabled by the event labeling in conjunction with event detection. Namely, the algorithm can be applied in tutoring systems evaluating the user's performance on any percussion instrument to provide low-level data on the user's playing. This data can be further applied in higher-level analysis techniques to compute, for example, the tempo, temporal deviations, correctness of the user's performance gestures (e.g., how clear is the difference of a rimshot and middle shot on the hand drum), and overall success in performing the correct pattern.

An exciting possibility of using probabilistic inference in tutoring systems is measuring the user's clarity and consistence in playing desired types of events. Considering that the algorithm can be trained, for example, with ideal sounds resulting from a rimshot to a Darbuka, the inference probability could be directly used as a measure of "goodness" in the sound reproduced by the user.

Another domain where the method can be applied is interactive accompaniment, to provide low-level analysis of the user's playing. The same compensation for a constant lag as in the tutoring scheme can be applied in these systems to sync the accompaniment with the human player. Systems such as Haile discussed in Section 2 already utilize low-cost event detection modules, but we believe that our technique surpasses the state of art with respect to labeling performance.

9. Conclusion

We have presented a real-time, model-based technique for detecting and labeling percussive sounds. To date, most related work on percussion recognition has concentrated on offline algorithms not suitable for interactive systems. Furthermore, the existing techniques typically have separate steps for sound detection and classification, whereas this technique performs both functions with a single method.

The algorithm utilizes supervised learning and is retrainable for any desired palette of relatively short percussive sounds. Therefore, the technique can be utilized in a multitude of real-time interactive musical systems requiring reliable detection and identification of percussive sounds performed by the user. The real-time implementation of the algorithm has been tested with different types of hand clapping sounds and percussive instrument sounds. The overall precision and recall rates are high with a moderate time lag used by the HMM procedure. This lag is not an issue in certain real-time interactive systems, such as rhythmic tutoring or automatic accompaniment.

In the current implementation, the templates are directly based on relatively low-order Fourier transform, which is

suitable for a generic algorithm. However, investigating other transforms more tuned especially for percussive sounds would potentially increase the performance. For example, filterbank-based transforms have been previously utilized successfully with percussive sounds [11] and are worth investigating with this technique as well with a comparative study.

The algorithm is currently tuned for monophonic sounds and has not been tested under polyphonic conditions. An interesting future step will be extending the algorithm for the detection and labeling of multiple simultaneous sounds, which would make it suitable for multiplayer systems with two or more percussion players performing simultaneously. We have already started to gather data on such situations and consider this as another major research step. However, for polyphonic event tracking, the model needs to be redefined and the inference scheme adjusted for simultaneously occurring events, and meeting the real-time requirements with an inevitably more complex algorithm requires special attention.

Acknowledgments

The authors would like to thank Reha Dişçioğlu for percussion recordings. This research has been funded by the Academy of Finland (Project no. 140826 SCHEMA-SID) and the Graduate School, Faculty of Electronics, Communications and Automation at Aalto University School of Science and Technology. This work is also supported by Boğaziçi University research fund no. BAP 09A105P.

References

[1] D. FitzGerald and J. Paulus, "Unpitched percussion transcription," in *Signal Processing Methods for Music*, A. Klapuri and M. Davy, Eds., Springer, New York, NY, USA, 2006.

[2] P. Herrera, A. Dehamel, and F. Gouyon, "Automatic labeling of unpitched percussion sounds," in *Proceedings of the 114th Convention of the Audio Engineering Society*, pp. 1–14, Amsterdam, The Netherlands, January 2003.

[3] A. Zils, F. Pachet, O. Delerue, and F. Gouyon, "Automatic extraction of drum tracks from polyphonic music signals," in *Proceedings of the 2nd International Conference on Web Delivering of Music (WEDELMUSIC '02)*, pp. 179–183, Darmstadt, Germany, January 2002.

[4] K. Yoshii, M. Goto, and H. Okuno, "Automatic drum sound description for real-world music using template adaptation and matching methods," in *Proceedings of the International Conference on Music Information Retrieval (ISMIR '04)*, pp. 184–191, Barcelona, Spain, October 2004.

[5] D. V. Steelant, K. Tanghe, S. Degroeve et al., "Classification of percussive sounds usingsupport vector machines," in *Proceedings of the Annual Machine Learning Conference of Belgium and The Netherlands (BENELEARN '04)*, pp. 146–152, Brussels, Belgium, January 2004.

[6] K. Tanghe, S. Degroeve, and B. D. Baets, "An algorithm for detecting and labeling drum events in polyphonic music," in *Proceedings of the 1st Annual Music Information Retrieval Evaluation Exchange*, pp. 11–15, London, UK, September 2005.

[7] K. Yoshii, M. Goto, and H. Okuno, "INTER: D: a drum sound equalizer for controlling volume and timbre of drums," in *Proceedings of the 2nd European Workshop on the Integration of Knowledge, Semantic and Digital Media Technologies (EWIMT '05)*, pp. 205–212, London, UK, November 2005.

[8] O. Gillet and G. Richard, "Automatic labelling of tabla signals," in *Proceedings of the International Conference on Music Information Retrieval (ISMIR '03)*, Baltimore, Md, USA, October 2003.

[9] J. Paulus and A. Klapuri, "Model-based event labeling in the transcription of percussive audio signals," in *Proceedings of the Digital Audio Effects Workshop (DAFx '03)*, pp. 73–77, London, UK, Sepember 2003.

[10] J. Paulus and A. Klapuri, "Drum sound detection in polyphonic music with hidden Markov models," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2009, Article ID 497292, 10 pages, 2009.

[11] M. Puckette, T. Apel, and D. Zicarelli, "Real-time audio analysis tools for pd and msp," in *Proceedings of the International Computer Music Conference (ICMC '98)*, pp. 109–112, Ann Arbor, Mich, USA, January 1998.

[12] G. Weinberg and S. Driscoll, "Toward robotic musicianship," *Computer Music Journal*, vol. 30, no. 4, pp. 28–45, 2006.

[13] A. Robertson and M. Plumbley, "B-Keeper: a beat-tracker for live performance," in *Proceedings of the 7th International Conference on New Interfaces for Musical Expression (NIME '07)*, pp. 234–237, June 2007.

[14] A. Jylhä, I. Ekman, C. Erkut, and K. Tahiroglu, "iPalmas—an interactive flamenco rhythm machine," in *Proceedings of Audio Mostly*, pp. 69–76, Glasgow, UK, September 2009.

[15] U. Şimşekli and A. T. Cemgil, "A comparison of probabilistic models for online pitch tracking," in *Proceedings of the 7th Sound and Music Computing Conference (SMC '10)*, pp. 502–509, July 2010.

[16] O. Cappe, E. Moulines, and T. Ryden, *Inference in Hidden Markov Models*, Springer Series in Statistics, Springer, Secaucus, NJ, USA, 2005.

[17] A. T. Cemgil, "Bayesian inference for nonnegative matrix factorisation models," *Computational Intelligence and Neuroscience*, vol. 2009, Article ID 785152, 17 pages, 2009.

[18] E. Alpaydın, *Introduction to Machine Learning*, Adaptive Computation and Machine Learning, MIT Press, Cambridge, Mass, USA, 2004.

[19] B. H. Repp, "The sound of two hands clapping: an exploratory study," *Journal of the Acoustical Society of America*, vol. 81, no. 4, pp. 1100–1109, 1987.

[20] L. Peltola, C. Erkut, P. R. Cook, and V. Välimäki, "Synthesis of hand clapping sounds," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 15, no. 3, Article ID 4100694, pp. 1021–1029, 2007.

[21] A. Jylhä and C. Erkut, "A hand clap interface for sonic interaction with the computer," in *Proceedings of Human Factors in Computing Systems (CHI '09)*, pp. 3175–3180, Boston, Mass, USA, April 2009.

[22] J. Ballas, "Self-produced sound: tightly binding haptics and audio," in *Proceedings of Haptic and Audio Interaction Design (HAID '07)*, I. Oakley and S. Brewster, Eds., pp. 1–8, Glasgow, UK, September 2007.

[23] A. Jylhä and C. Erkut, "Inferring the hand configuration from hand clapping sounds," in *Proceedings of the Digital Audio Effects Workshop (DAFx '08)*, pp. 301–304, Espoo, Finland, September 2008.

[24] P. Wolfe and S. Godsill, "Interpolation of missing data values for audio signal restoration using a Gabor regression model," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '05)*, pp. 517–520, Philadelphia, Pa, USA, March 2005.