Temporal Integration for Audio Classification With Application to Musical Instrument Classification

Cyril Joder, Slim Essid, and Gaël Richard, *Senior Member, IEEE*

***Abstract*—Nowadays, it appears essential to design automatic indexing tools which provide meaningful and efficient means to describe the musical audio content. There is in fact a growing interest for music information retrieval (MIR) applications amongst which the most popular are related to music similarity retrieval, artist identification, musical genre or instrument recognition. Current MIR-related classification systems usually do not take into account the mid-term temporal properties of the signal (over several frames) and lie on the assumption that the observations of the features in different frames are statistically independent. The aim of this paper is to demonstrate the usefulness of the information carried by the evolution of these characteristics over time. To that purpose, we propose a number of methods for early and late temporal integration and provide an in-depth experimental study on their interest for the task of musical instrument recognition on solo musical phrases. In particular, the impact of the time horizon over which the temporal integration is performed will be assessed both for fixed and variable frame length analysis. Also, a number of recently proposed alignment kernels will be used for late temporal integration. For all experiments, the results are compared to a state of the art musical instrument recognition system.**

***Index Terms*—Alignment kernels, audio classification, music information retrieval (MIR), musical instrument recognition, support vector machine (SVM), temporal feature integration.**

I. Introduction

A huge amount of varied audio data is nowadays available to the general public, but its volume and its nature often limit its visibility and therefore its accessibility. It then appears essential to design automatic indexing tools which provide meaningful and efficient means to describe the audio content or more specifically in our case the musical audio content. There is in fact a growing interest of the community for the different typical applications of music information retrieval (MIR) amongst which the most popular are related to music similarity retrieval, artist identification, musical genre, or instrument recognition.

Although they are different problems, most of the approaches follow a common multi-class classification strategy. First, an intermediate description of the signal is obtained by means of *features* which capture specific properties of the given signal. These features are usually extracted over short-time analysis windows (hereafter called frames), over which the signal can be considered stationary. Second, a classifier is used to identify the most probable class for the currently observed features. In a supervised approach, such a classifier would be trained on the features extracted from a training corpus containing labeled data for all classes. Except in some specific problems such as multiple fundamental frequencies estimation [1], the MIR-related classification systems usually do not take into account the mid-term temporal properties of the signal (over several frames). Indeed, many of the systems lie on the assumption that the observations of the features in different frames are statistically independent. In other words, they suppose that the evolution of these characteristics over time is not informative about the class membership of a given sound. Thus, a decision is made for each frame independently of the others (see for example [2]–[4]). Nevertheless, this is known to be suboptimal, hence leading a few researchers to propose various strategies to take into account the information conveyed in the temporal evolution of the signal. In previous works, these strategies have been referred to as *temporal integration*, which can be roughly defined as the process of combining several different feature observations in order to make a single decision [5].

Recently, Meng [6] has addressed the issue of temporal integration in the context of musical genre recognition. Although limited, there has also been some attempts for temporal integration in the context of musical instrument recognition (for example using vector quantization [7], using a model of the trajectories of spectral envelopes over a note [8] or applying a hidden Markov model (HMM) classifier with a structure-learning algorithm [9]).

Two kinds of feature integration processes can actually be distinguished: *early* and *late* integration. Early integration refers to the computation of a new feature vector that characterizes the signal at a higher time scale and which sums up the sequence of local features extracted over short-time analysis windows, or frames. The main advantages of such an integration include a reduction of the number of feature vector observations to be processed (and as a consequence a reduction of the complexity of the classification), the modeling of the temporal properties of the features and the possibility to combine features which are extracted over frames of different sizes. For some features, such as Mel-frequency cepstral coefficients (MFCCs), derivatives are widely used to catch some dynamical properties of the signal, but early integration of features has often been limited to straightforward methods (typically mean/variance) which do not model their temporal evolution. Most of them exploit the first statistical moments computed over a sequence of features (e.g., the mean and variance in [10], [11], the skewness and kurtosis in [12]). Other research directions include the concatena-

Manuscript received March 17, 2008; revised August 24, 2008. Current version published December 12, 2008. This work supported in part by the European Commission under Contract FP6-027026-K-SPACE. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Hiroshi Sawada.

The authors are with the Institut TELECOM, TELCOM ParisTech, CNRS LTCI, F-75014 Paris, France.

Digital Object Identifier 10.1109/TASL.2008.2007613

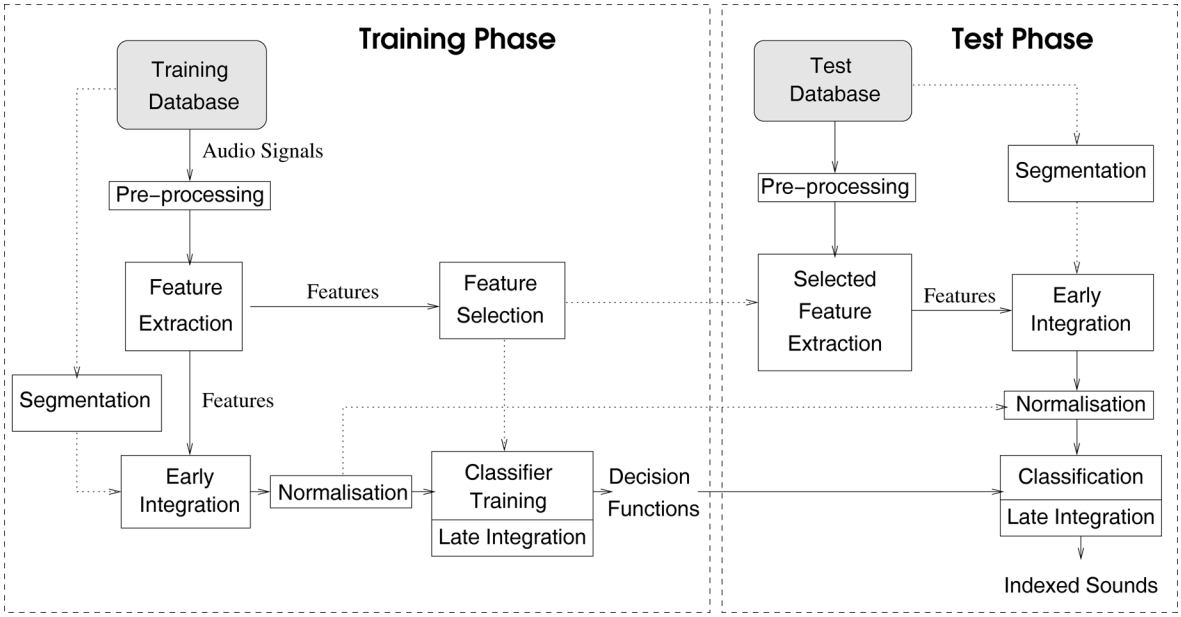

Fig. 1. Architecture of the musical instrument recognition system.

tion of several observation vectors [13], the use of a filterbank to summarize the periodogram of each feature [14], and the exploitation of autoregressive models to approximate the feature temporal dynamics [15], [16].

On the other hand, late temporal integration does not try to explicitly extract the feature dynamics. It operates at the classifier level, either by operating a "fusion" of successive primary decisions of the classifier, or by exploiting a classifier that can handle sequences. The usual way of combining several decisions of the classifier is to compute the product of the posteriors for each class, with the implicit assumption that the observations are independent, but because of this assumption, this approach does not capture any information about the temporal evolution of the features. The most popular way to overcome this problem is probably the use of HMM classifiers (see for example [17], [18]), which handle the sequentiality of the features by fitting a generative model to the features' temporal evolution. Other recent methods exploiting sequence kernel-based support vector machine (SVM) have been proposed, which use a dynamic alignment to measure similarities between sequences. The potential of such kernels was shown in the domain of handwriting recognition [19] or speech recognition [20], [21].

The purpose of this paper is to explore a number of early and late integration techniques, to propose original integration approaches and to provide an in-depth experimental study on their usefulness for the task of musical instrument recognition. The choice of the instrument recognition task is mainly motivated by the fact that temporal integration was not sufficiently exploited for that task while it is widely agreed that the evolution of a sound—in particular the temporal envelope—is crucial for humans to identify the instrument that produced it [22]. It is, furthermore, a well defined task with a clear and reliable ground-truth or annotation. In this paper, the impact of the time horizon on which the integration is performed will be assessed both for a fixed frame length analysis and a variable frame-length obtained by *a priori* sonic unit segmentation (see Section II-A.2). Another important aspect of this work is related to the late integration in which a number of recent SVM alignment kernels are used for the first time in an audio recognition task. For all experiments, the results are compared to a state of the art musical instrument recognition system [23].

The paper is organized as follows: first, the musical instrument recognition system is briefly described in Section II. Then, the theoretical background for early and late integration is presented in Section III. The experimental study is described in Section IV, and some conclusions are suggested in Section V.

II. Musical Instrument Recognition System

In the past, many musical instrument recognition systems have addressed the so-called *isolated notes* problem [24]–[26], but most of the recent systems tackle the more challenging task of instrument recognition from musical phrases of a solo instrument (with no accompaniment) or in polyphonic excerpts (with accompaniment) [27]. In this paper, we test the classification system on solo phrases. This classification scheme could be applied to mixed sounds, either by trying to recognize *ensembles* instead of isolated instruments as in [23], or by running the system on the output of a source separation algorithm.

The musical recognition engine used in this study is based on the core system proposed in [28] for musical phrases of solo instruments. The core system is built on a traditional signal analysis over successive overlapping windows on which a high number of features are extracted. A reduced set of features is then obtained by means of a feature selection algorithm and is used to train binary SVM classifiers. The system proposed in this paper (Fig. 1), however, integrates novel modules, namely a *Sonic Unit segmentation* module which allows the partitioning of the signal into segments corresponding to musical notes and two *temporal integration modules* (for early and late integration).

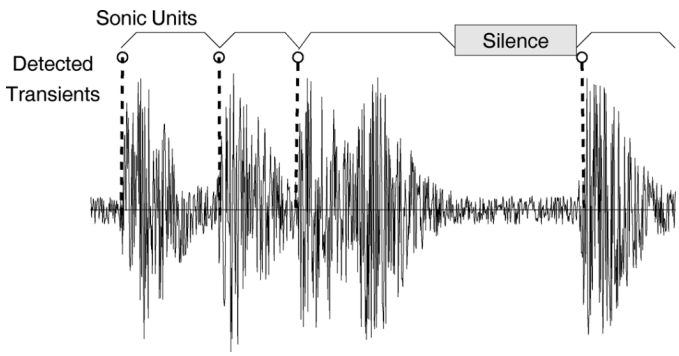

Fig. 2. Sonic unit segmentation.

A. Signal Analysis

As just mentioned, the input signal, downsampled at 32 kHz, is either analyzed using fixed length analysis frames or using a sonic unit segmentation that is further described below.

1) Silence Detection: A silence detector is used to eliminate outliers resulting from the extraction of features on very low energy and silence frames (where the instrument does not actually play). Our approach considers as silence:

- the frames whose maximal amplitude is at least 30 dB lower than the global maximal amplitude;
- the frames with a constant amplitude;
- the segments shorter than 15 frames between two silence frames.

These are indeed simple criteria, but they have proven sufficient for musical data recorded in good conditions (which is usually the case for commercial CDs).

2) Sonic Unit Segmentation: The aim of this segmentation is to obtain semantically meaningful segments that contain, in the ideal case, a single musical note. It is in fact assumed that on such semantically rich segments, the temporal integration should be more efficient since the feature trajectories over them should be more distinctive for each instrument. However, in the case of polyphonic instruments, or simply in the case of reverberant recording conditions, a note segmentation is very hard to obtain. Thus, our approach consists in extracting *sonic units* which are defined as inter-onset intervals, or intervals between an onset and a silence segment (see Fig. 2). An additional constraint is imposed to force the temporal units to be at least five frames long (96 ms) and at most 125 frames long (2 s).

The onset detection used in this paper is based on the predictability of the phase increment of sustained sounds ([29], [30]). This method supposes that the signal is composed of a number of stationary sinusoids. The phase increment between two consecutive frames is then constant, and the values of the short-time Fourier transform (STFT) can be predicted at each frame. When a transient appears, the characteristics of the signal change rapidly and the STFT cannot be predicted. An onset is then detected when the derivative of the prediction error is greater than a given threshold. The parameters have been set so that there are approximately as many false alarms as detection errors.

B. Feature Extraction and Selection

1) Feature Extraction: There exists no consensual set of features for the instrument recognition problem. Numerous proposals have been made in the past [31], [26], [3], [32], [25], [33], [34], and many have been integrated in the MPEG-7 standard [35]. Our strategy consists in extracting a wide range of potentially useful features to select the most relevant ones for our task, using a feature selection algorithm (FSA).

Two different analysis window sizes are used for the extraction of features: standard 32-ms frames for most features (used by default) and longer 960-ms windows when needed, with a half-window overlap. The complete set of features includes (see [28] or [34] for a more complete description).

Temporal Features:

- Local temporal waveform moments, including the first four statistical moments measured over both short (32-ms) and long (960-ms) windows. The first and second derivatives are also taken.
- Amplitude modulation (AM) features over two spectral ranges ([4–8 Hz] and [10–40 Hz]), measured over long windows[1]. For each range, three features are computed as described in [26], namely AM frequency, AM strength and AM heuristic strength. In addition, the product of the AM frequency and the AM amplitude in both ranges is also computed.
- Envelope moments measured on the long 960-ms frames[2]. The first and second derivatives are also taken.
- Zero crossing rates (ZCRs) computed over short and long windows.
- Auto-regressive coefficients of a second-order auto-regressive model fitted to the signal on each frame.

Cepstral Features: MFCCs are extracted as well as their first and second derivatives. Two sets of 11 MFCCs are considered, the first using 11 Mel subbands and the other using 30 Mel subbands.

Spectral Features:

- Spectral moment features are calculated from the first four spectrum statistical moments. The features are the spectral centroid, variance, skewness, and kurtosis.
- octave band signal intensities (OBSI) are defined by the logarithm of the energy in eight overlapping octave bands and the logarithm of the energy ratio of each subband to the previous [32].
- Spectral shape features include MPEG-7 audio spectral flatness (ASF), the spectral slope, spectral decrease [34], spectral flux [36], spectral rolloff, and spectral irregularity [31].

Wavelet Features are calculated from a Daubechies wavelet transform. The first three statistical moments and the energy of the coefficients corresponding to the same scale are computed on four different frequency subbands. See [37] for more details.

2) Feature Selection: We extract a total number of 162 feature coefficients. This large set of features may be redundant,

[1]The range 4–8 Hz describes the "tremolo" and the range 10–40 Hz captures the "roughness".

[2]To obtain the amplitude envelope, the modulus of the analytic signal is calculated and then filtered with a half 50-ms Hanning window.

and some features may be "noisy" or simply non-relevant in discriminating the classes. Feature selection is then essential to reduce the complexity of the problem (by reducing the dimensionality) as well as to eliminate the non-discriminant features [38].

The automatic feature selection algorithm used is the Fisher algorithm, which is inspired by discriminant linear analysis [39]. It is a *filter* approach, which does not consider the classification application. The algorithm iteratively selects the features which maximize the *Fisher discriminant*

$$r = \frac{|\tilde{\mu}_p - \tilde{\mu}_q|^2}{\tilde{\sigma_p}^2 - \tilde{\sigma_q}^2},$$

in each bi-class problem (i.e., for $1 \leq p < q \leq Q$, with Q the total number of classes considered), where $\tilde{\mu}_q$ and $\tilde{\sigma}_q^2$ are, respectively, the estimated mean and variance of class q data.

As a result of the selection process, the remaining selected features are:

- the first three spectral moments;
- 13 MFCCs of the two considered types;
- six of the OBSI coefficients and five OBSI ratio coefficients;
- five wavelet transform coefficients;
- three spectral irregularity features, the spectral roll-off, one ASF coefficient;
- the two ZCR coefficients and the AM heuristic strength in the range [10–40 Hz].

C. Support Vector Machines and Kernels

SVMs are powerful classifiers that have proven to be efficient for various classification tasks, such as face recognition, speaker identification, and instrument recognition [23]. These classifiers are known for their good generalization property, even in high dimension. SVMs also have the advantage of being *discriminative*, as opposed to *generative* approaches, in the sense that they do not assume any particular form of the data probability density.

Given training samples $\mathbf{x}_1, \ldots, \mathbf{x}_n$ which are assigned class labels $y_1, \ldots, y_n$, with $y_i \in \{-1, 1\}$, the algorithm searches for the hyperplane which separates the two classes with a maximal margin. This hyperplane is the solution of an optimization problem, and is defined by the equation $f(\mathbf{x}) = 0$, where f has the form

$$f(\mathbf{x}) = \sum_{i=1}^{n_s} \alpha_i y_i \mathbf{s}_i \cdot \mathbf{x} + b.$$

Here, f only depends on a small number of training vectors, called support vectors, which are denoted by $\mathbf{s}_i$. n_s is the number of support vectors and the α_i are Lagrange multipliers. New feature vectors are then classified according to the sign of $f(\mathbf{x})$. See [40] for more details on the optimization problem of the SVM.

Furthermore, non linear decision surfaces can be obtained by mapping the input vectors to a higher dimension space. The dot product in this vector space is given by a function called a *kernel*. Let k be this kernel function, a feature vector $\mathbf{x}$ is classified according to the sign of

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i y_i k(\mathbf{s}_i, \mathbf{x}) + b.$$

Note that neither the corresponding space nor the mapping function need to be known explicitly in order to express the classification function. The knowledge of the kernel, which can be seen as a similarity measure between vectors, is sufficient.

Given such a function and a family of vectors $\mathbf{x}_1, \ldots, \mathbf{x}_m$, the *Gram Matrix* of k with respect to $\mathbf{x}_1, \ldots, \mathbf{x}_m$ is the $m \times m$ matrix G defined by $G_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$. A sufficient condition for k to be a proper kernel function, i.e., to represent the dot product between vectors "mapped" to a Hilbert space, is that for all $m \in \mathbb{N}$ and all vectors $x_1, \ldots, x_m$, the corresponding Gram matrix be positive-definite. Such a kernel is then called a positive-definite kernel. In this case, the Hilbert space into which the feature vectors are mapped, can be explicitly constructed [40]. Note, however, that kernels which are not positive-definite may result in good classification results in practice (see for example [21]), although there is no theoretical proof that their use is well justified.

As reference kernel, we choose the Gaussian radial basis function (RBF) kernel, denoted by k_0, since it achieves the best classification performance for our instrument recognition problem, among the kernels tested in [28]. The following form is used:

$$k_0(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{D\sigma^2}\right)$$

where D is the dimension of the vectors and σ^2 is a parameter of the kernel.

Another kernel ξ proposed by Cuturi *et al.* [21] is also considered, defined by

$$\xi(\mathbf{x}, \mathbf{y}) = \frac{\frac{1}{2}k_0(\mathbf{x}, \mathbf{y})}{1 - \frac{1}{2}k_0(\mathbf{x}, \mathbf{y})}.$$

This kernel, which is numerically similar to the Gaussian kernel, has been introduced so that $\xi/(1-\xi)$ be positive-definite, as will be explained in Section III-B.3.

In order to perform multi-class classification, we adopt a "one versus one" strategy and use Platt's approach [41] which derives posterior class probabilities after the two-class SVMs.

III. Temporal Integration

The temporal integration methods used in this work are presented below. Note that "instantaneous" low-level features, i.e., features computed locally over short sliding *analysis frames* are used. The early integration is then performed over larger time windows called *texture windows*. Yet another time window is used, over which the late integration is performed—and the classification decision is taken. We call these windows *decision horizon* or *decision length*. In order to avoid confusions, we will use the term *frame* to refer to the *analysis frames*, and the term

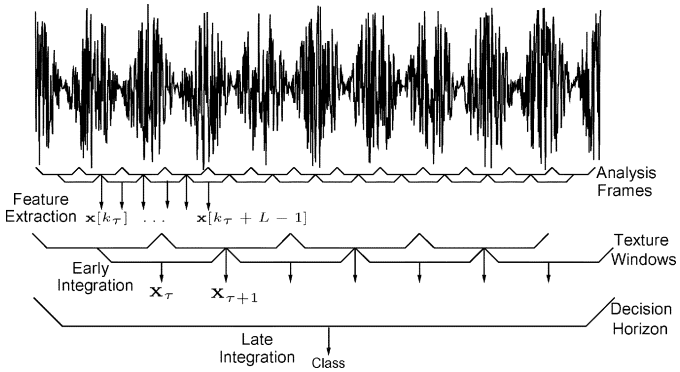

Fig. 3. Illustration of the different windows used (analysis frame, texture window, and decision horizon).

TABLE I
OVERVIEW OF THE EARLY FEATURE INTEGRATION METHODS CONSIDERED. COLUMN "DIMENSION" GIVES THE DIMENSION OF THE INTEGRATED FEATURE VECTOR

Notation	Name	Dimension
mean	mean	D
mVar	mean-variance	$2D$
mCov	mean-covariance	$\frac{D(D+3)}{2}$
MAR(P)	multivariate auto-regressive model (order P)	$D(DP+1)$
DAR(P)	autoregressive model (order P)	$D(P+1)$
CAR(P)	autoregressive model variant (order P)	$D(P+1)$
spec	power spectrum	$D(\frac{N}{2}+1)$
specMom	spectral moments	$5D$
specShape	spectral shape	$4D$
stack	feature stacking	DL

window for the *texture windows*. Fig. 3 illustrates the different time windows considered.

A. Early Integration Methods

Let $\{x_i[k]\}_{i=1...D}$ be the values of the D scalar features observed in the kth frame. The *feature vector* (also called observation vector) corresponding to this frame is denoted by the row vector $\mathbf{x}[k] = \left[x_1[k]\, x_2[k]\, \ldots\, x_D[k]\right]$. Early integration can be represented by a function f of a sequence of L D-dimensional feature vectors, which returns a single vector whose dimension M may be different from D.

The integration over the τth texture window of length L, spanning the frames k_τ to $k_\tau + L - 1$, consists in calculating the *integrated observation vector*

$$\mathbf{X}_\tau = f(\mathbf{x}[k_\tau], \ldots, \mathbf{x}[k_\tau + L - 1]).$$

Table I sums up the early integration methods presented hereafter.

These integration functions compute "higher order" descriptors of the feature vector sequence. Some functions capture the short-time statistics (such as the mean and covariance) of the features whereas others try, by considering the sequence as a random process, to characterize its power-spectral density (PSD). This allows one to better model the dynamics of the feature vector sequence, which is done either by extracting autoregressive coefficients, or by using the periodogram.

1) Simple Statistics: A simple way to perform early integration is to compute first order statistics of the feature process. The *mean* integration function is defined as

$$f_{\text{mean}}(\mathbf{x}[k_\tau], \ldots, \mathbf{x}[k_\tau + L - 1]) = \boldsymbol{\mu}_\tau = \frac{1}{L} \sum_{k=k_\tau}^{k_\tau+L-1} \mathbf{x}[k].$$

It is also possible to include second order statistics such as the covariance matrix. The method referred to as *mCov*, is defined as

$$f_{\text{mCov}}(\mathbf{x}[k_\tau], \ldots, \mathbf{x}[k_\tau + L - 1]) = \left[\boldsymbol{\mu}_\tau\ \text{vec}(\boldsymbol{\Sigma}_\tau)\right]$$

where

$$\boldsymbol{\Sigma}_\tau = \frac{1}{L-1} \sum_{k=k_\tau}^{k_\tau+L-1} (\mathbf{x}[k] - \boldsymbol{\mu}_\tau)^T (\mathbf{x}[k] - \boldsymbol{\mu}_\tau)$$

and the notation vec refers to the concatenation of all the rows of the matrix into a single row vector. This method introduces a great increase of the dimension of the integrated observation vector. However, as the covariance matrix is symmetric, it is only needed to keep $M = D(D+3)/2$ values per texture window.

For a smaller increase of dimension, it is possible to keep only the empirical variance of the features (without the correlations between different features). The function f_{mVar} is then given as

$$f_{mVar}(\mathbf{x}[k_\tau], \ldots, \mathbf{x}[k_\tau + L - 1]) = \left[\boldsymbol{\mu}_\tau\ \text{diag}(\boldsymbol{\Sigma}_\tau)\right]$$

where the notation $\text{diag}(\boldsymbol{\Sigma}_\tau)$ refers to the row vector whose coefficients are the diagonal coefficients of the matrix $\boldsymbol{\Sigma}_\tau$. The dimension here becomes $M = 2D$.

2) Autoregressive Models: The simple statistical methods do not model the temporal dependency between the successive features observations, since a permutation of these observations would result in the same integrated feature vector. To overcome this shortcoming, the following methods try to fit an autoregressive (AR) process to the sequence. Thus, the spectral properties of the process can be "captured" by calculating the coefficients of the optimal (in the least square sense) whitening FIR filter of order P.

The multivariate autoregressive (MAR) model was used by Meng in [6] for musical genre classification. It handles the temporal dependencies between feature vector observations by an affine prediction scheme. The Pth order model, denoted by $MAR(P)$ is defined as

$$\mathbf{x}[k] = \mathbf{w} + \sum_{n=1}^{P} \mathbf{x}[k-n]\mathbf{A}_n + \boldsymbol{\varepsilon}_k$$

where $\mathbf{A}_n$ are $D \times D$ matrices, $\mathbf{w}$ is a vector of dimension D and $\{\boldsymbol{\varepsilon}_k\}_{k\in\mathbb{Z}}$ is a D-dimensional white noise vector.

The integrated observation vector (of dimension $D(PD+1)$) is defined as

$$f_{\mathrm{MAR}}(\mathbf{x}[k_\tau],\ldots,\mathbf{x}[k_\tau+L-1]) = \left[\hat{\mathbf{w}}\ \mathrm{vec}(\hat{\mathbf{A}}_1)\ \cdots\ \mathrm{vec}(\hat{\mathbf{A}}_P)\right]$$

where $\hat{\mathbf{w}}$ and $\{\hat{\mathbf{A}}_n\}_{n=1,\ldots,P}$ are the least-square estimators of the model parameters for the τth texture window (where the index τ has been dropped out for notation simplicity).

The diagonal autoregressive (DAR) model is the same as above, but with the assumption that the features are independent processes. The matrices $\mathbf{A}_n$ are then diagonal and the integrated feature vector of dimension $D(P+1)$ is given by

$$f_{\mathrm{DAR}}(\mathbf{x}[k_\tau],\ldots,\mathbf{x}[k_\tau+L-1]) = \left[\hat{\mathbf{w}}\ \mathrm{diag}(\hat{\mathbf{D}}_1)\cdots\mathrm{diag}(\hat{\mathbf{D}}_P)\right]$$

where the $\hat{\mathbf{D}}_n$ are estimates of the $\mathbf{A}_n$ under the constraint that they are diagonal. This is done by estimating the parameters for each feature independently of the others.

Another integration function exploits the same diagonal model. The centered autoregressive (CAR) function is given by

$$f_{\mathrm{CAR}}(\mathbf{x}[k_\tau],\ldots,\mathbf{x}[k_\tau+L-1]) = \left[\boldsymbol{\mu}_\tau\ \mathrm{diag}(\tilde{\mathbf{D}}_1)\cdots\mathrm{diag}(\tilde{\mathbf{D}}_P)\right]$$

where the $\tilde{\mathbf{D}}_n$ are the mean-square estimators of the model parameters for the "centered" process $\bar{\mathbf{x}}[k] = \mathbf{x}[k] - \boldsymbol{\mu}_\tau$. They can be computed thanks to the Levinson–Durbin algorithm [42].

Note that although the models for the DAR and CAR methods are equivalent, the integration functions are different. With the DAR function, all the parameters are jointly estimated, whereas for the CAR model, the mean is computed before the AR coefficients are estimated on the centered observations. Note that the relation between $\mathbf{w}$ and $\boldsymbol{\mu}$ is complex as it depends on the other dimensions (the exact relation is $\mathbf{w} = (\mathbf{I} - \sum_{n=1}^{P}\mathbf{A}_n)\boldsymbol{\mu}$).

3) Spectral Features: Temporal information about the feature processes are also extracted by considering spectral characteristics of these features. This is inspired by Mckinney's and Breebart's work [14], where the modulation energy of several features (namely the MFCCs) are computed over four subbands. We propose here three methods which handle the features' spectral information by computing a STFT for every feature, over the texture windows. A N-point Fourier transform, with zero-padding depending on the size of the texture window is used.

Spectrum: The *spectrum*-based integration function (*spec*) returns the power spectrum of the features (in dB): $s_{i,\tau}^{\mathrm{dB}}[n] = 20\log|s_{i,\tau}[n]|$.

The integrated observation vector is then[3]

$$f_{\mathrm{spec}}(\mathbf{x}[k_\tau],\ldots,\mathbf{x}[k_\tau+L-1]) = \left[\mu_{1,\tau}\ \mathbf{s}_\tau^{dB}[1]\ \ldots\ \mathbf{s}_\tau^{dB}\left[\frac{N}{2}\right]\right]$$

where $\mathbf{s}_\tau^{dB}[n] = \left[s_{1,\tau}^{dB}[n]\ \ldots\ s_{i,\tau}^{dB}[n]\ \ldots\ s_{D,\tau}^{dB}[n]\right]$. The dimension of this vector is $D(N/2+1)$.

Spectral Moments: The mean $m_{i,\tau}$, variance $v_{i,\tau}$, skewness $\gamma_{i,\tau}$, and kurtosis $\kappa_{i,\tau}$ of the amplitude spectrum of the feature sequence over the texture window are considered for the *spectral moments*-based integration method (*specMom*).[4]

The corresponding integration function is defined as

$$f_{\mathrm{specMom}}(\mathbf{x}[k_\tau],\ldots,\mathbf{x}[k_\tau+L-1]) = \left[\boldsymbol{\mu}_\tau\ \boldsymbol{m}_\tau\ \boldsymbol{v}_\tau\ \boldsymbol{\gamma}_\tau\ \boldsymbol{\kappa}_\tau\right].$$

The dimension of the integrated feature vector is thus $5D$.

Features Spectral Shape: The *specShape* integration method considers spectral shape parameters of each feature, namely the spectral slope (Ss_i), the spectral decrease (Sd_i) and the spectral roll-off (Ro_i). They are computed, respectively, as

$$\mathrm{Ss}_i = \frac{K\sum_{n=0}^{K}\omega_n|s_i[n]| - \sum_{n=0}^{K}\omega_n\sum_{n=0}^{K}|s_i[n]|}{K\sum_{n=0}^{K}\omega_n^2 - \left(\sum_{n=1}^{K}|s_i[n]|\right)^2}$$

$$\mathrm{Sd}_i = \frac{1}{\sum_{n=1}^{K}|s_i[n]|}\sum_{n=0}^{K}\frac{|s_i[n]| - |s_i[0]|}{n}$$

with $K = N/2$ and $\omega_n = n/N$. The spectral roll-off is here defined as the frequency below which 99% of the spectral energy is accounted for. The integrated feature vector (of dimension $4D$) is then

$$f_{\mathrm{specShape}}(\mathbf{x}[k_\tau],\ldots,\mathbf{x}[k_\tau+L-1]) = \left[\boldsymbol{\mu}_\tau\ \mathbf{Ss}_\tau\ \mathbf{Sd}_\tau\ \mathbf{Ro}_\tau\right]$$

4) Feature Stacking: The last method, which will be referred to as *stack*, was initially proposed by Slaney in [13] and also used in [6], [43]. It consists in concatenating all the feature vectors observed over a texture window into a single vector. The temporal dependency between the features is then implicitly modeled by the classifier. The integration function returns $f_{stack}(\mathbf{x}[k_\tau],\ldots,\mathbf{x}[k_\tau+L-1]) = \left[\mathbf{x}[k_\tau]\ldots\mathbf{x}[k_\tau+L-1]\right]$. This introduces a significant increase of the feature space dimension, as the size of the integrated feature vector is DL.

B. Late Integration Methods

In [6] and [44], Meng explores several late integration methods using SVM kernels in a musical genre recognition system. For instance, he tests the product probability kernel (PPK) and the convolution kernel. The former operates a late integration, but leads to scores which are lower than the ones obtained with the static Gaussian kernel in our case for music instrument recognition. Hence, we rather focus here on recently proposed *alignment kernels*, to assess their performances in our classification problem. A three-state Gaussian-mixture HMM is used as reference of dynamic classifier, since it is widely used in music applications [17]. In order to have another reference for static classifiers, Gaussian mixture models (GMMs) [39] are also used.

1) Fusion of Decisions: Let $\mathbf{x}_\tau,\ldots,\mathbf{x}_{\tau+\theta}$ be a sequence of feature vectors. The output of a "static" probabilistic classifier is an estimate of the probability $\mathrm{Prob}(q|\mathbf{x}_\tau)$ of a class q, given

[3]Note that the first coefficient which should correspond to the log-spectrum mean is simply replaced by the mean to put more emphasis on this important coefficient.

[4]The spectral moments are obtained as follows: $m_{i,\tau} = (2/N)\sum_{n=0}^{N/2}|s_{i,\tau}[n]|$; $v_{i,\tau} = (2/N)\sum_{n=0}^{N/2}(|s_{i,\tau}[n]| - m_{i,\tau})^2$; $\gamma_{i,\tau} = (2/Nv_{i,\tau}{}^3)\sum_{n=0}^{N/2}(|s_{i,\tau}[n]| - m_{i,\tau})^3$; $\kappa_{i,\tau} = (2/Nv_{i,\tau}{}^4)\sum_{n=0}^{N/2}(|s_{i,\tau}[n]| - m_{i,\tau})^4 - 3$.

observation $\mathbf{x}_\tau$. In order to make a decision over the whole sequence, a strategy must be adopted which combines these "partial decisions." In the case of independent feature vector observations, the probability of class q, given the whole sequence is

$$\mathrm{Prob}(q|\mathbf{x}_\tau,\ldots,\mathbf{x}_{\tau+\theta}) \propto \prod_{i=0}^{\theta} \mathrm{Prob}(q|\mathbf{x}_{\tau+i}).$$

Here, the prior $\mathrm{Prob}(q)$ and the marginals $\mathrm{Prob}(\mathbf{x}_\tau)$ have been dropped out, since it is assumed that the prior is uniform and the marginals are not needed in the subsequent calculation. Following this approach, the baseline late integration method operates as follows: the sum of all the class log-probabilities (used instead of the probabilities for better numerical stability) over the sequence is computed for each class, then the class associated with the maximum value is chosen. This late integration method will be referred to as *decision fusion* (DF). In this paper, it is applied to SVM as well as GMM classifiers. Note that several other strategies are possible in order to perform late integration based on independent partial decisions. See [45] for a study on some of these techniques.

2) Hidden Markov Model (HMM) Classifier: One of the most common approaches for the implementation of more elaborate late integration is using an HMM classifier. The feature vectors are then no longer considered as independent random variables. The model supposes a certain structure of the process, that will not be detailed here and we refer the interested reader to one of the many good tutorials about HMMs, for example [46], [47]. An HMM captures the statistical dependencies of the feature vectors, and allows for the straightforward calculation of the likelihood of a model, given a whole sequence. In order to perform the classification, a model is trained for each class. Then, the most probable model is associated with every sequence that needs to be classified.

This standard classifier is our reference system for dynamic late integration.

3) Alignment Kernels: Let $\underline{\mathbf{x}} = (\mathbf{x}_1,\ldots,\mathbf{x}_n)$ be a finite series of feature vectors which needs to be classified. In a "static" strategy, each vector $\mathbf{x}_i$ is compared (thanks to the kernel function) to every support vector, and then classified according to $f(\mathbf{x}_i)$ as in (II-C). However, it is not always relevant to classify a feature vector independently of the surrounding others. As the temporal structure of music is important, it may be more meaningful to compare whole sequences of vectors.

Sequence kernels allow for the comparison of trajectories of feature vectors, instead of operating on single observations. Thus, a sequence $\underline{\mathbf{x}}$ can be classified "as a whole," according to a decision function

$$f(\underline{\mathbf{x}}) = \sum_{i=1}^{n_s} \alpha_i y_i k(\underline{\mathbf{s}}_i, \underline{\mathbf{x}}) + b \qquad (1)$$

where $\underline{\mathbf{s}}_i$ are sequences instead of isolated feature vectors. Alignment kernels are special cases of sequence kernels, whose value is not invariant with respect to a permutation of the feature vectors. One advantage of this method is that it does not assume any specific parametric form for the feature probability distribution, nor for the structure of their temporal dependencies. As can be seen in (1), the classification is performed by comparing a vector sequence to the "support vector sequences," which can be seen as template sequences. Thus, in theory any trajectory can be taken into account, whereas our Gaussian mixture HMM models a particular form of the conditional distribution as well as a certain type of temporal dependency.

In order to cope with the problems of feature sequence synchronization, the comparison is made after a *temporal alignment* of the sequences, which may be of different lengths. We now briefly describe the alignment algorithm before presenting the alignment kernels used.

Let $\underline{\mathbf{y}} = (\mathbf{y}_1,\ldots,\mathbf{y}_m)$ be another finite feature series. An *alignment path* π of length $p < m+n$ between $\underline{\mathbf{x}}$ and $\underline{\mathbf{y}}$ is a function from $\{1,\ldots,p\}$ to $\{1,\ldots,n\}\times\{1,\ldots,m\}$ such that, with the notation $\pi(i) = \big(\pi_1(i),\pi_2(i)\big)$, where $i \in \{1,\ldots,p\}$

1) the functions $i \mapsto \pi_1(i)$ and $i \mapsto \pi_2(i)$ are increasing;
2) the function π is injective.

The series $\big(\pi(1),\ldots,\pi(p)\big)$ represents a sequence of p pairs of indexes which align the two series $\underline{\mathbf{x}}$ and $\underline{\mathbf{y}}$ without changing the order of the feature vectors (property 1) and with no repetition (property 2). The constraint that all the vectors of both series are taken into account in the alignment is also added (in order to forbid heaps in the alignment path). This imposes that the functions π_1 and π_2 be surjective onto, respectively, $\{1,\ldots,n\}$ and $\{1,\ldots,m\}$.

Let $\mathcal{A}$ be the set of all possible alignment paths between $\underline{\mathbf{x}}$ and $\underline{\mathbf{y}}$. The traditional dynamic time-warping (DTW) algorithm returns the value of the *DTW distance*, $\Delta(\underline{\mathbf{x}},\underline{\mathbf{y}})$ defined as a distance between the two aligned sequences along the *optimal* path according to the following criterion:

$$\Delta(\underline{\mathbf{x}},\underline{\mathbf{y}})^2 = \min_{\pi\in\mathcal{A}} \frac{1}{M_\pi} \sum_{i=1}^{p} m_\pi(i) \|\mathbf{x}_{\pi_1(i)} - \mathbf{y}_{\pi_2(i)}\|^2 \qquad (2)$$

where $m_\pi(i)$ are non-negative weighting coefficients and $M_\pi = \sum_{i=1}^{p} m_\pi(i)$ is the normalization factor. The value of the weighting coefficients is a function of the increment $\big(\pi_1(i)-\pi_1(i-1),\pi_2(i)-\pi_2(i-1)\big)$. This function influences the optimality criterion, hence favoring or penalizing certain of paths.

The Gaussian dynamic time-warping kernel (GDTW) introduced in [19] uses this alignment between two sequences. The idea is to exploit the DTW distance instead of the Euclidian distance in the calculation of a Gaussian kernel. Thus, the resulting value of this kernel is the geometric mean of the Gaussian kernel values along the optimal alignment path. The GDTW kernel is then defined as

$$\begin{aligned} K_{\mathrm{GDTW}}(\underline{\mathbf{x}},\underline{\mathbf{y}}) &= \exp\left(-\frac{1}{2D\sigma^2}\Delta(\underline{\mathbf{x}},\underline{\mathbf{y}})^2\right) \\ &= \max_{\pi\in\mathcal{A}} \prod_{i=1}^{p} k_0\left(\mathbf{x}_{\pi_1(i)},\mathbf{y}_{\pi_2(i)}\right)^{m_\pi(i)/M_\pi}. \end{aligned}$$

The dynamic time-alignment kernel (DTAK) proposed in [20] calculates another similarity measure between two sequences by considering the arithmetic mean of the kernel values

along the alignment path. Applying this idea to the Gaussian kernel, the DTAK kernel is then defined as

$$K_{\mathrm{DTAK}}(\underline{\mathbf{x}},\underline{\mathbf{y}}) = \max_{\pi\in\mathcal{A}} \frac{1}{M_\pi} \sum_{i=1}^{p} m_\pi(i) k_0(\mathbf{x}_{\pi_1(i)}, \mathbf{y}_{\pi_2(i)}). \quad (3)$$

Note that the optimal alignment paths considered by these two kernels may be different. Indeed, the local similarity used for the GDTW kernel is the Euclidian distance, whereas the one used in (3) is the Gaussian kernel value.

Cuturi *et al.* [21] emphasize the fact that these alignment kernels have not been proven positive-definite. In the same work, they introduce another alignment kernel type which is positive-definite under a certain assumption. It is similar to the GDTW kernel but sums the values obtained with all the possible alignments. Given a "static" kernel κ, a corresponding alignment kernel $\mathcal{K}_\kappa$ can be defined as

$$\mathcal{K}_\kappa(\underline{\mathbf{x}},\underline{\mathbf{y}}) = \sum_{\pi\in\mathcal{A}} \prod_{i=1}^{p} \kappa(\mathbf{x}_{\pi_1(i)}, \mathbf{y}_{\pi_2(i)}). \quad (4)$$

Here, all the weighting coefficients are equal to 1. There is no normalization neither with respect to the alignment length, nor with respect to the number of alignments considered. Thus, the kernel values depend on the length of the sequences. The authors prove that if κ is such that $\kappa/(1+\kappa)$ is positive-definite, then $\mathcal{K}_\kappa$ is also positive-definite.

The similarity measure induced by this kernel is different from the previous alignment kernels. Indeed, the sum in (4) takes advantage of every possible alignment instead of only the optimal one. Thus, two sequences are similar in the sense of $\mathcal{K}_\kappa$ not only if they have an alignment which results in a small DTW distance, but also if they share numerous suitable alignments.

We consider two instances of (4) which are proposed in [21]. The first alignment kernel is the application of this framework with the "static" kernel $\xi = (1/2)k_0/(1-(1/2)k_0)$. Thus, the alignment kernel obtained is positive-definite. Its formulation is

$$\mathcal{K}_\xi(\underline{\mathbf{x}},\underline{\mathbf{y}}) = \sum_{\pi\in\mathcal{A}} \prod_{i=1}^{p} \frac{\frac{1}{2}k_0(\mathbf{x}_{\pi_1(i)}, \mathbf{y}_{\pi_2(i)})}{1-\frac{1}{2}k_0(\mathbf{x}_{\pi_1(i)}, \mathbf{y}_{\pi_2(i)})}.$$

The second one uses the kernel $\chi = e^{k_0}$. Cuturi *et al.* found that the Gram matrices obtained with the latter were exceedingly diagonally dominant, that is the diagonal values of these matrices are many orders of magnitude larger than the other values. Thus, the different vectors are almost orthogonal in the reproducing space and it has been observed in practice that the SVMs do not perform well in such situations [48]. The authors of that work suggest using the logarithm of these values, arguing that although it does not conserve positive-definiteness, it achieves good classification performances. This kernel is thus defined as

$$\mathcal{K}_\chi(\mathbf{x},\mathbf{y}) = \log\left(\sum_{\pi\in\mathcal{A}} \prod_{i=1}^{p} e^{k_0(x_{\pi_1(i)}, y_{\pi_2(i)})}\right).$$

As pointed out by the authors, this kernel calculates in fact the soft-max[5] of scores of all possible alignments, rather than the simple maximum as for the first two alignment kernels.

These alignment kernels introduce an increase of the kernel computation complexity. With a dynamic programming implementation, K_{DTAK} and $\mathcal{K}_\kappa$ require mn static kernel calculations. K_{DTAK} involves only one static kernel computation, but the alignment process also requires $O(mn)$ operations.

C. Combined Integration Methods

Early and late integration methods can also be applied jointly. In that case, the late integration is performed on "early integrated" features. To that aim, the integration windows are split in n segments on which early integration is performed. The series of n "early integrated" feature vectors can then be used with a late integration classifier. For example, in the case where the early integration is *mean* and $n = 3$, the sequence $\mathbf{X}_\tau = \left[\boldsymbol{\mu}_{\tau,1}\ \boldsymbol{\mu}_{\tau,2}\ \boldsymbol{\mu}_{\tau,3}\right]$ is obtained, where $\boldsymbol{\mu}_{\tau,1} = (3/l)\sum_{k=k\tau}^{k\tau+l/3} \mathbf{x}[k]$. This method is below referred to as *stackmean3*.

Other combinations are also considered in this work including *stackmean5*, *stackmVar3* and *stackmVar5*.

IV. Experimental Study

In a first set of experiments, the influence of the texture window length and of the choice of the early integration function on the performance of our instrument classification system is assessed. The use of the sonic unit segmentation is then evaluated. Finally, in the last set of experiments, the potential of the late integration methods both on local and early integrated features is studied. In order to assess the efficiency of the sequence classifiers, they are compared to their "static" counterpart: GMM are evaluated against HMM and static kernels are compared to alignment kernels (in SVM classifiers).

A. Experimental Setup

1) Database and Pre-Processing: The database used in this work is a slightly enlarged version of Essid's database [32]. It is composed of solo recordings of eight different instruments, representing the main categories of instruments as detailed in Table II. Most of the sounds have been taken from commercial CDs of classical or contemporary music, jazz, and educational discs. Additional clarinet and trumpet samples have been recorded on purpose. Solo pieces of the RWC database [49], [50] have also been used. The class "clarinet" contains recordings of B flat and E flat clarinets. Note that the training and test databases are extracted from different recordings.

The input sound files are downsampled to a 32-kHz sampling rate, centered and normalized to get zero-mean, unit-variance signals.

B. Integration Parameters

The AR parameters of the *MAR* and *DAR* models are estimated using the *ARfit* toolbox [51]. For the spectral early integration methods, we set the number of frequency bins N to 64, in order to be able to compute a FFT of the feature signal over

[5]the soft-max of the real numbers $z_1, \ldots, z_n$ is defined as $\log\sum_{i=1}^{n} e^{z_i}$

TABLE II
DURATION OF THE TRAINING AND TEST SETS, FOR EACH CLASS

Instrument	Training set	Test set
Piano	22' 16"	14' 13"
Guitar	10' 43"	15' 58"
Oboe	14' 46"	14' 40"
Clarinet	8' 34"	13'38"
French Horn	3' 43"	3' 24"
Trumpet	10' 46"	11' 30"
Cello	15' 47"	12' 7"
Violin	34' 11"	24' 11"
Total	120' 26"	109' 41"

texture windows up to 64 frames, which represents about 1s of signal. Two texture window configurations are considered.

1) *Constant window length*, with a constant hop-size. Eight different lengths from the set $W_l = \{10, 20, 30, 40, 50, 60, 90, 120\}$ in number of frames (corresponding to lengths from 176 ms to 1.936 s) are evaluated. A constant ten-frame hop-size is used which enables us to keep a nearly constant number of texture windows whatever their length may be.
2) *Variable window length*, where a texture window is defined for each sonic unit (defined in Section II-A) with no overlap.

Note that in order to avoid noisy data, the frames detected as silence are not considered and that the integration is performed on successive non-silence frames only. Too long texture windows are split into several segments and an integrated observation vector is calculated over these segments.

The mean and standard deviation of the integrated observation feature vectors are estimated on the whole training database. The mean is subtracted from every observation vector, and the result is divided by the standard deviation. Thus, the classifiers run on normalized data.

C. Classification Parameters

The experiments are done using public implementations of the different classifiers[6].

For the SVM classifier, the parameter C is set to 1, based on the results of previous work [28] and the parameter σ^2 is searched for in the set $W_\sigma = \{0.25, 0.5, 1, 2, 4, 8, 16, 25, 32, 40, 49\}$.

To model the musical notes, a left-right three-state HMM is used, in order to capture, respectively, the attack, sustain and release of each sonic unit. The probability density in each state is modeled as an eight-component Gaussian mixture. Several numbers of components were tested for the GMM classifier, and we present here only the results of the best system, which uses eight components. The training of these models is performed with the expectation-maximization (EM) algorithm over the separated sonic units, after an initialization phase using the k-means algorithm. The parameters are chosen according to the conclusions of previous studies [23], [28] using the same type and volume of data, which showed that eight Gaussian components was a good choice for this problem, after varying this number in the set $\{1, 2, 8, 16, 32, 64, 128, 256\}$. The Gaussians have full covariance matrices.

The alignment kernels presented in Section III-B.3 use a form of the DTW algorithm. The chosen weighting coefficients are equal to 1 for a "horizontal" or "vertical" step and 2 for a unitary "diagonal" step, so that the normalization coefficient M_π is independent of the alignment path.

D. Decision Process

It is important to emphasize that the decisions are made on the same time segments which guarantees to have meaningful comparisons when different texture window lengths are used[7].

The decision horizon is chosen to be an integer number of texture windows. The lengths used are chosen from the set W_l (that is from 10 to 120 frames). In the case of sonic units, the decision horizon used is the number of sonic units whose cumulated lengths are the closest to the chosen length.

It must be noted that only "continuous" decision horizons are considered. When the interval between two silence segments is shorter than the chosen length, the decision horizon used is reduced to this interval.

Since in our database, the amount of data differs from one instrument to another, the score used to compare the classification systems is the average recognition rate, i.e., the average of all the classes recognition rates. With the figures (all expressed in percents) is also specified the upper bound of the 95% confidence interval (corresponding to the worst case), which is referred to as *confidence*.

E. Experiment 1: Early Integration

In this section, we focus on the early integration functions presented in Section III-A. The aim of this study is to determine the best parameters for the early integration in terms of classification performance. The parameters to set are the following:

- the integration function (and the order in the case of the auto-regressive models);
- the texture window lengths;
- the decision horizon.

The different experiments conducted are summed up in Table III. Since CAR models turned out to be more effective than DAR, they are tested on a wider range of orders (up to order 9). In the case of the *MAR* model, the shortest texture window is 656 ms, because the estimation of the parameters demands a large number of observations. A total of 622 experiments have been run. Our reference system (k_0-SVM+DF) is the one presented in [28] which exploits a SVM classifier with the Gaussian RBF kernel k_0 (see Section II-C) with a fixed value for the parameter σ^2 ($\sigma^2 = 1$). It does not use early temporal features integration, but performs late fusion of decisions as seen is Section III-B.1. The parameters of the tested SVM classifiers are set to the same values as the ones used in the reference system.

For the sake of clarity, Table IV only presents the results for the best early integration functions. The systems that are not

[6]"SVM-Light" implementation by Joachims [52] for SVMs, Murphy's MATLAB toolbox [53] for GMMs and HMMs.

[7]Note however that when the texture windows are shorter than the chosen decision horizon, an additional fusion of the decisions is performed according to the method seen in Section III-B.1, e.g., based on the product of the classes posterior probabilities.

TABLE III
PARAMETERS OF THE EXPERIMENTS. SU STANDS FOR SONIC UNIT. THE TEXTURE WINDOWS AND DECISION HORIZONS LENGTHS ARE SELECTED FROM THE SET $\{176, 336, 496, 656, 816, 976, 1456, 7936\}$ ms

Integration Function	Order	Texture Windows (ms)	Decision horizons (ms)
k_0-*SVM+DF*	–	(32)	176 to 1936
mean	–	176 to 1936 / 1 SU	176 to 1936
mVar	–	176 to 1936 / 1 SU	176 to 1936
mCov	–	176 to 1936	176 to 1936
stack	–	176 to 1936	176 to 1936
MAR	1	656 to 1936	656 to 1936
DAR	1-3	176 to 1936	176 to 1936
CAR	1-9	176 to 1936 / 1 SU	176 to 1936
spec	–	176 to 1936 / 1 SU	176 to 1936
specMom	–	176 to 1936 / 1 SU	176 to 1936
specShape	–	176 to 1936 / 1 SU	176 to 1936

TABLE IV
AVERAGE RECOGNITION RATES OBTAINED USING THE "BEST" TEXTURE WINDOWS LENGTHS (A SELECTION OF RESULTS). DECISIONS HORIZON: 1.936 s (120 FRAMES). CONFIDENCE ON CONSTANT TEXTURE WINDOWS: $\pm 0.7\%$; CONFIDENCE ON SONIC UNITS: $\pm 1.0\%$

Texture Window	Constant	Sonic Unit
k_0-*SVM+DF*	82.3	81.5
mean	82.0 [336]	80.2
mVar	81.5 [176]	81.6
mCov	79.3 [496]	77.4
stack	81.2 [176]	–
CAR(1)	82.1 [336]	79.5
DAR(1)	76.9 [176]	75.3
MAR(1)	56.8 [816]	–
spec	82.0 [336]	78.1
specMom	82.3 [496]	69.6
specShape	82.3 [336]	70.8

presented here all obtain lower scores than the reference system.

In accordance with one's natural intuition, the recognition rate always increases with the length of the decision horizon, whatever the integration function. For decisions made over longer length (about one minute), the score can be greatly improved (we obtain 89.7% without early integration). Thus, the length of the decision horizon is not a critical parameter, and a real improvement of a classification system should be measured using a constant decision horizon length. To that purpose, all scores below are given for a fixed decision horizon of 2 s.

When the decision is made on a single texture window, the classification results of early integration systems are rather poor compared to the reference system. In the case of the *specMom* function with 336-ms texture windows, the score is 74.6% for a decision horizon of 336 ms while the reference system average recognition rate is 77.0%. However, when the decision takes into account several texture windows, it seems to be more reliable. In the same example, the score becomes 82.1% versus 82.3% for the reference system. When appropriate, the texture windows size that leads to the best results is given into square brackets (in ms).

In order to enable fair comparisons, the reference system is used with both decision horizon types, corresponding respectively to 2-s and to the closest number of sonic unit to this length. The difference between the obtained scores (0.8%) can be explained by the decision horizon lengths. Indeed, for a chosen length of 2-s (120 frames), about 20% of the decision horizons are shorter than 110 frames when considering the sonic units, versus less than 3.5% for the other strategy. Nevertheless, we use these figures to compare the systems using constant texture windows and the systems using sonic units as texture windows.

The results obtained are somewhat mixed. On the one hand, the experiments show that none of the early integration functions tested induce an improvement of the classification score compared to the reference system. On the other hand, some of the integration functions may lead to equivalent performance at a much lower complexity. In fact, the *mean* function can lead to an equivalent score to the reference (respectively 82.0% and 82.3%) with a 10-frame hop-size between texture windows, and thus a complexity reduced by a factor of ten.

The performances of the autoregressive functions decrease when the order of the model increases. This tends to indicate that they are not really appropriate for the modeling of the features' temporal evolution.

A disappointing observation is that none of the early integration functions performs significantly better than either the *mean* or the *mVar* functions, although they cannot render the sequential evolution of the features. This tends to indicate that the tested early integration functions do not truly capture the temporal aspects which are relevant to our problem.

In general, the score does not change considerably when the texture window length varies in the range [176-500 ms]. However, for longer texture windows, the recognition rate is significantly lower than the reference system score, for all the tested integration functions. For most of these functions, the best texture window length is 336 ms (20 frames). In fact, for nearly all of them, this length induces scores that are equivalent to the highest score obtained with the same function. This may be explained by the fact that such a duration should often correspond to a typical note duration. Indeed, it corresponds to an eighth note at a tempo of 90 beats per minute, which is a very common tempo. Furthermore, the average sonic unit length in our database is 387 ms (23.4 frames) and the median is 272 ms (16 frames), which tends to confirm this claim.

The sonic unit segmentation does not improve the performances either and does not seem to be appropriate for early integration since none of the experiments succeeded to obtain a function of lower complexity with the same performance as the reference system. Note though that the function *mean* may represent a satisfactory tradeoff between complexity and performance.

In summary, whereas early integration can allow for a reduction of a classification system complexity, it does not improve the performance of the system. Therefore, the mean/variance techniques are the most advantageous early integration functions, as they are the simplest ones and they do not affect the classification results.

F. Experiment 2 : Late Integration

In this section, the late integration methods presented in Section III-B are assessed first without early integration. However, it will then be shown that it is only when the late

TABLE V
RECOGNITION RATES WITH THE "BEST" VALUES OF σ^2. DECISION HORIZON: 120 FRAMES (ABOUT 2 s). CONFIDENCE: $\pm 0.8\%$)

Late integration Window	k_0-SVM+DF	GMM	HMM	GDTW-SVM	DTAK-SVM	$\mathcal{K}_\xi$-SVM	$\mathcal{K}_\chi$-SVM
Value of σ^2	1	–	–	0.5	0.5	32	1
20 frames (Confidence: ± 0.8)	82.3	78.0	80.8	78.8	78.9	78.4	82.0
Sonic Unit (Confidence: ± 1.1)	81.5	76.7	81.3	72.1	76.1	–	–

integration is combined with early integration that a clear increase of performance can be obtained.

1) Late Integration Only: The late integration systems proposed in this paper are compared to three reference systems, namely an SVM (the previous k_0-SVM+DF), a GMM and an HMM classifier.

For all these systems, the late integration is performed on constant 20-frame (336-ms) integration windows and on sonic units. Then an additional fusion of the decisions is run in order to take the decision over 2-s decision horizons, or the number of sonic units which is the closest to this length. The results are summarized in Table V. A SVM system using a kernel k is referred to as k-SVM and the decision fusion is denoted by DF. For each SVM classifier of the late integration experiments, the given results are obtained with the parameter σ^2 inducing the best performance, from the set W_σ.

The results of the SVMs using the alignment kernels $\mathcal{K}_\xi$ and $\mathcal{K}_\chi$ over the sonic units are not presented due to convergence problems of the SVM optimization. An explanation of this phenomenon may be the absence of DTW normalization in the computation of these kernels (see (4)). Consequently, the kernel values depend on the sequence length and the obtained gram matrices may be ill-conditioned in the case of highly variable lengths, resulting in a bad convergence of the optimization algorithm.

These figures show that the HMM system is significantly better than the GMM system, which suggests that taking into account the temporal dependency of the feature vectors does improve the classification performance. Furthermore, the HMM classifier obtains a higher score with the so-called sonic units than with constant integration windows, and reaches the same performance as the reference SVM system.

However, on the overall, the results obtained are not conclusive about the potential advantage of the late integration when used on raw features since the different approaches are in any case more complex than the reference "static" system and only equal its performance in the best case ($\mathcal{K}_\chi$).

Nevertheless, it will be shown that late integration becomes interesting when it operates on "early integrated" features.

G. Combined Integration: Summary Sequence Classification

It is proposed in this section to study the potential of combined integration. The principle of this new approach is sketched in Fig. 4. The integration windows are split in a small number of subsegments of the same length. Then, an early integration is performed over each of the subsegments. Finally, the summary sequences (the sequences of these integrated feature vectors), are classified so that a decision is made over every integration window.

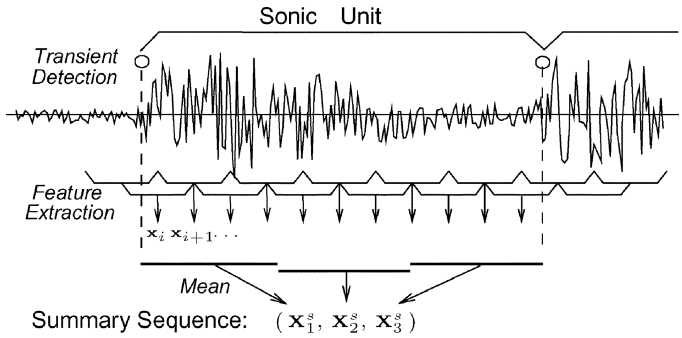

Fig. 4. Sonic unit segmentation and summary sequence. Here is represented a three-subsegment summary sequence of a sonic unit using *mean*: the sonic unit is divided in three equal-length segments; the mean of the feature vectors $\mathbf{x}_i$ is computed over each of the subsegments; the summary sequence is then the sequence of these means $(\mathbf{X}_1^S, \mathbf{X}_2^S, \mathbf{X}_3^S)$.

The results in Table IV showed that no early integration function performs significantly better than the simplest ones. Therefore, the early integration functions tested in these experiments are *mean* and *mVar*. Two integration windows are used: uniform 20-frame windows and sonic units. Finally, tests are run using three and five subsequences for each integration window. We choose these small numbers because the sonic units can be as short as five-frame length. Besides, the complexity of the kernel computations is much smaller than in the previous tests, allowing us to use the SVM classifiers with the alignment kernels $\mathcal{K}_\xi$ and $\mathcal{K}_\chi$.

The experiments are run with the same classifiers as above and with the system ξ-SVM+DF. For each SVM system, the kernel parameter σ^2 is chosen in the set W_σ. Table VI sums up the recognition rates obtained with the most effective values of σ^2.

These results show some significant improvements of the recognition rates in comparison with the best reference system (up to $+2.4\%$), indicating that the use of summary sequences is an efficient way of combining early and late integration. Using summary sequences with the function *mVar* can improve the classification score with the ξ-SVM+DF system (up to $+1.1\%$), even though it uses a "static" classifier.

However, the scores of the HMM classifier decrease with the use of summary sequences. An explanation to this may be that the obtained sequences are too short to take advantage of this classifier. Indeed, it leads to more scarce training data, and the learning of the model is probably less accurate.

The figures also show that, whereas the kernels GDTW and DTAK still offer no benefit, SVMs using the alignment kernels $\mathcal{K}_\xi$ and $\mathcal{K}_\chi$ are efficient for sequence classification. Almost all the systems using these kernels obtain scores which are similar to or better than the reference. The combination of these setups is successful: the systems using alignment kernel $\mathcal{K}_\xi$ or

TABLE VI
RECOGNITION RATES USING SUMMARY SEQUENCES. DECISION HORIZON: 120 FRAMES (ABOUT 2 s). IN BOLDFACE ARE THE SCORES WHICH ARE SIGNIFICANTLY HIGHER THAN THE REFERENCE SYSTEM

Integration Window	20 frames (Confidence: $\pm 0.8\%$) Reference Score: 82.3%				Sonic Unit (Confidence: $\pm 1.0\%$) Reference Score: 81.6%			
Summary Sequence	$3 \times$ *mean*	$5 \times$ *mean*	$3 \times$ *mVar*	$5 \times$ *mVar*	$3 \times$ *mean*	$5 \times$ *mean*	$3 \times$ *mVar*	$5 \times$ *mVar*
GMM	79.3	80.7	80.1	79.7	79.2	78.7	80.4	79.5
HMM	78.9	81.1	80.3	79.3	79.5	78.8	78.9	79.2
k_0-SVM+DF	82.1	82.0	82.6	81.6	81.9	81.0	81.9	77.8
ξ-SVM+DF	81.8	82.0	**83.4**	82.4	81.8	80.8	**83.1**	81.8
DTAK-SVM	80.1	79.5	79.2	76.8	81.3	80.2	79.8	77.4
GDTW-SVM	79.9	79.5	77.1	78.7	79.7	78.3	78.4	75.4
$\mathcal{K}_\xi$-SVM	81.3	79.2	81.4	82.6	**84.1**	**84.7**	**83.8**	82.0
$\mathcal{K}_\chi$-SVM	81.8	82.3	**83.6**	82.9	**83.6**	**83.4**	**84.0**	**84.5**

$\mathcal{K}_\chi$ to classify summary sequences of sonic units obtain significantly higher recognition rates than the reference system. The best system appears to be $\mathcal{K}_\xi$-SVM with summary sequences using the function *mean* on five subsegments of the sonic units.

Combined early and late integration leads to better results than single integration scheme. This is due to the complementary role of the integration strategies. Indeed, the *mean* and *mVar* functions reduce the influence of outliers, while late integration successfully models the temporal evolution of these "robust" integrated features.

V. CONCLUSION

In this paper, a large number of experiments was conducted to assess the impact of temporal integration (both at an early stage, i.e., at the feature level and at a late stage, i.e., at the classifier level) on the performances of state of the art automatic musical instrument recognition systems. The results obtained show that temporal integration can significantly improve the performance of a classification system, at the cost of a possible increase of the problem dimensionality. This change does not inevitably lead to an augmentation of the complexity, as the number of observation vectors is decreased.

Our tests show significant improvements of the recognition rate, compared to our references, when early and late integration are combined. Another interesting outcome of these experiments is the contribution of the semantically rich temporal segmentation into sonic units. In fact, the best system combines early integration (i.e., mean of successive features on short segments) and late integration on these sonic units. Finally, it was shown that the dynamic kernels (that were not previously used in this context) are very efficient, especially when the temporal integration is conducted on sonic units.

However, the calculation of these efficient alignment kernels ($\mathcal{K}_\xi$ and $\mathcal{K}_\chi$) is more complex than the classic Gaussian kernel. Although the complexity remains in the same order of magnitude, it would be interesting to study in the future new ways or paradigms to overcome the complexity of the resulting SVM classifiers, e.g., by reducing the number of kernel evaluations. Another critical point to be investigated is the normalization of these kernels, which seems to be necessary for the classification of segments of variable lengths. Studying the influence of the weighting coefficients m_π of the alignment algorithm (Section III-B.III) would also be an interesting perspective, as they may affect the behavior of the alignment kernels.

It is also worth mentioning that in the current framework, the feature selection is completely independent of the subsequent integration. In a sense, the selected features are the best considering that there is no integration and may therefore not be the best choices for a system with integration. This may also explain to a certain extent that several setups with integration have difficulties to outperform the reference system. An interesting direction which is being considered will then consist in taking into account the temporal integration in the feature selection process.

REFERENCES

[1] M. Ryynänen and A. Klapuri, "Automatic bass line transcription from streaming polyphonic audio," in *Proc. IEEE ICASSP*, Honolulu, HI, Apr. 2007, pp. IV-1437–IV-1440.

[2] J. Marques and P. Moreno, "A study of musical instrument classification using Gaussian mixture models and support vector machines," Tech. Rep. Compaq Computer Corp., Cambridge, MA, 1999.

[3] A. Eronen, "Comparison of features for musical instrument recognition," in *Proc. IEEE Workshop Applicat. Signal Process. Audio Acoust.*, 2001, pp. 19–22.

[4] M. A. Loureiro, H. B. de Paula, and H. C. Yehia, "Timbre classification of a single musical instrument," in *Proc. Int. Symp. Music Inf. Retrieval*, 2004, pp. 546–549.

[5] A. Verma, T. Faruquie, C. Neti, S. Basu, and A. Senior, "Late integration in audio-visual continuous speech recognition," in *Proc. Automatic Speech Recognition and Understanding Workshop (ASRU)*, 1999.

[6] A. Meng, "Temporal feature integration for music organisation," Ph.D. dissertation, Technical Univ. of Denmark, Lyngby, Denmark, 2006.

[7] S. Dubnov and X. Rodet, "Timbre recognition with combined stationary and temporal features," in *Proc. Int. Comput. Music Conf.*, 1998.

[8] L. G. Martins, J. J. Burred, G. Tzanetakis, and M. Lagrange, "Polyphonic instrument recognition using spectral clustering," in *Proc. ISMIR*, 2007, pp. 213–218.

[9] M. Eichner, M. Wolff, and R. Hoffmann, "Instrument classification using hidden Markov models," in *Proc. ISMIR*, 2006, pp. 349–350.

[10] G. Tzanetakis, G. Essl, and P. Cook, "Automatic musical genre classification of audio signals," in *Proc. ISMIR*, 2001.

[11] M. Mandel and D. Ellis, "Song-level features and SVMs for music classification," in *Proc. ISMIR*, 2005, pp. 594–599.

[12] N. Scaringella and G. Zoia, "On the modeling of time information for automatic genre recognition systems in audio signals," in *Proc. ISMIR*, 2005, pp. 666–671.

[13] M. Slaney, "Semantic-audio retrieval," in *IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2002, vol. 4, pp. IV-1408–IV-1411.

[14] M. F. McKinney and J. Breebart, "Features for audio and music classification," in *Proc. Int. Symp. Music Inf. Retrieval*, 2003, pp. 151–158.

[15] M. Athineos, H. Hermansky, and D. Ellis, "Lp-trap: Linear predictive temporal patterns," in *Proc. Int. Conf. Spoken Lang. Process.*, 2004, pp. 1154–1157.

[16] A. Meng, P. Ahrendt, J. Larsen, and L. K. Hansen, "Temporal feature integration for music genre classification," *IEEE Trans. Audio, Speech, Lang. Process.*, pp. 1654–1664, Jul. 2006.

[17] A. Eronen, "Musical instrument recognition using ica-based transform of features and discriminatively trained hmms," in *Proc. 7th Int. Symp. Signal Process. and its Applicat.*, 2003, pp. 133–136.

[18] T. Kitahara, M. Goto, K. Komatani, T. Ogata, and H. G. Okuno, "Musical instrument recognizer "instrogram" and its application to music retrieval based on instrumentation similarity," in *Proc. IEEE Int. Symp. Multimedia*, 2006, pp. 265–274.
[19] C. Bahlmann, B. Haasdonk, and H. Burkhardt, "On-line handwriting recognition with support vector machines—A kernel approach," in *Proc. 8th IWFHR*, 2002, pp. 49–54.
[20] H. Shimodaira, K. Noma, M. Nakai, and S. Sagayama, "Dynamic time-alignment kernel in support vector machine," in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 2002.
[21] M. Cuturi, J.-P. Vert, O. Birkenes, and T. Matsui, "A kernel for time series based on global alignments," in *Proc. IEEE ICASSP*, Apr. 2007, vol. 2, pp. 413–416.
[22] J. Grey, "Multidimensional perceptual scaling of musical timbres," *J. Acoust. Soc. Amer.*, pp. 1270–1277, 1977.
[23] S. Essid, G. Richard, and B. David, "Instrument recognition in polyphonic music based on automatic taxonomies," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 14, no. 1, pp. 68–80, Jan. 2006.
[24] I. Kaminskyj and A. Materka, "Automatic source identification of monophonic musical instrument sounds," in *Proc. IEEE Int. Conf. Neural Netw.*, 1995, pp. 189–194.
[25] K. D. Martin, "Sound-source recognition: A theory and computational model," Ph.D. dissertation, Mass. Inst. Technol., Cambridge, 1999.
[26] A. Eronen, "Automatic Musical Instrument Recognition," M.S. thesis, Tampere Univer. of Technol., Tampere, Finland, 2001.
[27] T. Kitahara, M. Goto, K. Komatani, T. Ogata, and H. Okuno, "Instrument identification in polyphonic music: Feature weighting with mixed sounds, pitch-dependent timber modeling, and use of musical context," in *Proc. ISMIR*, 2005, pp. 558–563.
[28] S. Essid, "Classification automatique des signaux audio-fréquences : Reconnaissance des instruments de musique," Ph.D. dissertation, École Nationale Supérieure des Télécommunications, Paris, France, 2005.
[29] P. Leveau, L. Daudet, and G. Richard, "Methodology and tools for the evaluation of automatic onset detection algorithms in music," in *Proc. ISMIR*, 2004, pp. 72–75.
[30] J. Bello, C. Duxburry, M. Davies, and M. Sandler, "On the use of phase and energy for musical onset detection in the complex domain," *IEEE Signal Process. Lett.*, vol. 11, no. 6, pp. 553–556, Jun. 2004.
[31] J. C. Brown, O. Houix, and S. McAdams, "Feature dependence in the automatic identification of musical woodwind instruments," *J. Acoust. Soc. Amer.*, pp. 1064–1072, 2000.
[32] S. Essid, G. Richard, and B. David, "Musical instrument recognition based on class pairwise feature selection," in *Proc. ISMIR*, 2004, pp. 560–567.
[33] G. Peeters, "Automatic classification of large musical instrument databases using hierarchical classifiers with inertia ratio maximization," in *Proc. 115th AES Convention*, 2003.
[34] G. Peeters, "A Large Set of Audio Features for Sound Description (Similarity and Classification) in the Cuidado Project," Tech. Rep. IRCAM, 2004.
[35] *ISO/IEC, "Information Technology – Multimedia Content Description Interface – Part 4: Audio"*, , Jun. 2001, Int. Standard ISO/IEC FDIS 15938–4: 2001(E).
[36] E. Scheirer and M. Slaney, "Construction and evaluation of a robust multifeature speech/music discriminator," in *Proc. ICASSP '97*, Munich, Germany, 1997, pp. 1331–1334.
[37] T. Li and M. Ogihara, "Music genre classification with taxonomy," in *Proc. Int. Conf. Spoken Lang. Process.*, 2005, pp. 197–200.
[38] I. Guyon and A. Elisseeff, "An introduction to feature and variable selection," *J. Mach. Learn. Res.*, pp. 1157–1182, 2003.
[39] R. Duda, P. Hart, and D. E. Stork, *Pattern Classification*. New York: Wiley-Interscience, 2001.
[40] B. Scholkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA: MIT Press, 2001.
[41] J. Platt, "Probabilistic outputs for support vector machines and comparison to regularized likelihood methods," in *Advances in Large Margin Classifiers*, A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, Eds. Cambridge, MA: MIT Press, 1999.
[42] P. Delsarte and Y. V. Genin, "The split Levinson algorithm," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-34, no. 3, pp. 470–478, Jun. 1986.
[43] A. Rabaoui, M. Davy, S. Rossignol, Z. Lachiri, and N. Ellouze, "Using one-class SVMs and wavelets for audio surveillance systems," *IEEE Trans. Inf. Forensic Security*, 2007, submitted for publication.
[44] A. Meng and J. Shawe-Taylor, "An investigation of feature models for music genre classification using the support vector classifier," in *Proc. ISMIR*, 2005, pp. 604–609.
[45] J. Kittler, M. Hatef, R. P. Duin, and J. Matas, "On combining classifiers," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 3, pp. 226–239, Mar. 1998.
[46] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. IEEE*, vol. 77, no. 2, pp. 257–286, Feb. 1989.
[47] J. A. Bilmes, "What hmms can do," *IEICE – Trans. Inf. Syst.*, vol. E89-D, no. 3, pp. 869–891, 2006.
[48] J.-P. Vert, H. Saigo, and T. Akutsu, *Local Alignment Kernels for Biological Sequences*. Cambridge, MA: MIT Press, 2004, pp. 131–154.
[49] M. Goto, *RWC Music Database: Popular, Classical, and Jazz Music Databases*, 2002.
[50] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, *RWC Music Database: Music Genre Database and Musical Instrument Sound Database*, 2003.
[51] T. Schneider and A. Neumaier, Arfit [Online]. Available: http://www.gps.caltech.edu/ tapio/arfit/
[52] T. Joachims, SVM-Light Toolbox [Online]. Available: http://svmlight.joachims.org/
[53] K. Murphy, HMM Matlab Toolbox [Online]. Available: http://www.cs.ubc.ca/ murphyk/Software/HMM/hmm.html.

Cyril Joder received the Engineering degree from the École Polytechnique and TELECOM ParisTech, Paris, France, and the M.Sc. degree in acoustics, signal processing, and computer science applied to music from the University Pierre et Marie Curie, Paris, in 2007. He is currently pursuing the Ph.D. degree in the Department of Signal and Image Processing, TELECOM ParisTech, in the field of music information retrieval.

His research mainly focuses on automatic music structure analysis and audio-to-score temporal alignment.

Slim Essid received the Electrical Engineering degree from the École Nationale d'Ingénieurs de Tunis, Tunis, Tunisia, in 2001, the M.Sc. (D.E.A.) degree in digital communication systems from the École Nationale Supérieure des Télécommunications (ENST), in 2002, and the Ph.D. degree from the Université Pierre et Marie Curie (Paris 6), in 2005 after completing a thesis on automatic audio classification in the Department of Signal and Image Processing, ENST.

Currently, he is a Lecturer and Research Engineer at TELECOM ParisTech. His research interests are in signal processing and machine learning applied to multimedia classification and indexing.

Gaël Richard (SM'06) received the State Engineering degree from the École Nationale Supérieure des Télécommunications, Paris, France, in 1990, the Ph.D. degree from LIMSI-CNRS, University of Paris-XI, in 1994 in speech synthesis, and the Habilitation à Diriger des Recherches degree from the University of Paris XI in September 2001.

After the Ph.D. degree , he spent two years at the CAIP Center, Rutgers University, Piscataway, NJ, in the Speech Processing Group of Prof. J. Flanagan, where he explored innovative approaches for speech production. From 1997 to 2001, he successively worked for Matra Nortel Communications, Bois d'Arcy, France, and for Philips Consumer Communications, Montrouge, France. In particular, he was the Project Manager of several large-scale European projects in the field of audio and multimodal signal processing. In September 2001, he joined the Department of Signal and Image Processing, TELECOM ParisTech (formerly ENST), where he is now a Full Professor in audio signal processing and Head of the Audio, Acoustics, and Waves research group. He is a coauthor of over 70 papers and inventor in a number of patents. He is also one of the experts of the European commission in the field of audio signal processing and man—machine interfaces.

Prof. Richard is an Associate Editor of the IEEE TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING.