Hindawi Publishing Corporation
EURASIP Journal on Advances in Signal Processing
Volume 2011, Article ID 650204, 10 pages
doi:10.1155/2011/650204

Research Article

Adaptive Linear Prediction Filtering in DWT Domain for Real-Time Musical Onset Detection

Leonardo Gabrielli, Francesco Piazza, and Stefano Squartini

3MediaLabs, Department of Bioengineering, Electronics and Telecommunications, Università Politecnica delle Marche, 60121 Ancona, Italy

Correspondence should be addressed to Stefano Squartini, s.squartini@univpm.it

Received 15 September 2010; Revised 30 December 2010; Accepted 11 February 2011

Academic Editor: Federico Fontana

Onset detection is a typical digital signal processing task in acoustic signal analysis, with many applications as in the musical field. Many techniques have been proposed so far, which are typically reliable in terms of performances but often not suitable to real-time computing, for example, they require knowledge of the whole piece to perform optimally, or they are too computationally intensive for most embedded processors. Up to the authors' knowledge, the real-time implementation problem for musical onset detection has been scarcely addressed within the literature, which has motivated them to propose a scalable and computationally efficient algorithm with good detection capabilities. Comparison with other techniques and porting to a real-time embedded processor are discussed as well: provided experimental results seem to confirm the effectiveness of the approach.

1. Introduction

Onset detection is a fundamental task in many music DSP scenarios. Some of these have no critical time constraints, such as database queries, content retrieval, data mining, and so forth. Other applications have critical real-time constraints and ask for implementation on embedded processors (from specialized floating-point DSP to generic RISC architectures) for consumer electronics, performing arts, or musical equipment. Thus, it can be of interest to scale the most recent techniques for onset detection to fit into such platforms.

Early onset detection techniques required low computational effort [1], but not as much performing as the more recent techniques. On the other hand, the latest onset detection approaches yield remarkable performances, but they are not well-suited for real-time embedded platforms. For example, a recent contribution from Bruni et al. [2] makes use of the wavelet transform to construct a complex scheme for onset detection, making it unpractical for a real-time implementation. Other methods exist (see, e.g., [3, 4]), which make use of neural networks or similar machine-learning structures to analyze the signal features or to combine the extracted features to obtain the final decision. Other papers, such as [5], although making use of more efficient DSP techniques, may combine information coming from different onset detection algorithms, requiring an increase in computational cost and a complex decision taking mechanism at the end of the signal processing. Finally, some onset detection methods exist (see, e.g., [6]) which are targeted for the retrieval of soft onsets (i.e., very slowly occurring transient), which require a very long time knowledge, and cannot, hence, provide results without a sensible latency, if run in real-time.

In fact, beside the mere computational efficiency problem, real-time processing means to have only a small portion of the signal to process, hence missing long-term information on the piece. A problem often encountered in onset detection algorithms is that they necessarily require a long portion of audio especially when mixed with other musical information retrieval tasks (see, e.g., [7]). This long-term knowledge is often needed to gather information on the dynamics of the piece, but also to apply heuristic rules for removal of false negatives or for beat and tempo analysis.

For real-time onset detection the latency must be kept to the minimum allowed by the application under study, reducing frames and buffers length to low values, making the use of certain techniques impracticable.

Last but not least, two other remarkable issues have to be considered during the algorithm porting to a DSP processor: the finite-precision arithmetic and the meaninglessness of fixed thresholds. The former can often be addressed by using double-precision floating-point arithmetic, the latter may require the use of slowly-adaptive gains to suitably match the incoming signal amplitude.

To obtain a good trade-off between the real-time constraints and the detection performances, we propose an algorithm which yields good detection performances in real-time, also allowing to be scaled down to be implemented on low-cost processors, still ensuring good performances. The proposed algorithm is an improved version (and even more flexible in terms of real-time implementation) of an approach already appeared in the literature [8, 9], based on the evaluation of linear prediction error from a filter bank. This approach has been found to be effective and has been chosen as a starting point to develop our algorithm further. The main differences with the original papers, are in the use of the discrete wavelet transform (DWT) for the signal subband decomposition, and in the combination of the subband error signals with the use of a MSP-fashioned (multiscale product) [10, 11] merge. The peak picking procedure is different as well and made fast by the use of two low-order linear filters and a maximum search.

The algorithm is divided into two sections, the first one aimed at the extraction of a proper Transient Detection signal (from now on, shortly, T_D), and the second one providing a peak picking functionality, in order to obtain exact time instants of the onsets. The first section can be executed frame-wise or sample-by-sample, whereas the second one can be executed periodically, according to the application requirements and the implementation trade-offs.

Computer simulations (using the publicly available music database [12]) have been performed for two different scaled versions of the algorithm, a higher-end implementation and a lower-end one, showing remarkable performances for the first one and still acceptable performances for the second. As final outcome of the scalable algorithm here developed, a performing implementation has been also carried out on real-time embedded target (the TMS320C6747 floating-point processor powered by Texas Instruments), confirming the real-time versatility of the approach.

Real-time applications of an onset detection algorithm may include many different scenarios, not only in the audio field. Real-time onset detection may be, for instance, applied to gesture controllers for gaming, live music performances, or control of media devices such as smartphones. Nevertheless, this algorithm has been designed to deal with musical signals and in this field we can find the most relevant applications. Offline onset detection algorithms are more oriented to database-centric information retrieval, whereas a real-time onset detection system may be used for creating new sonic interaction design scenarios, innovative live performance instruments, tempo detectors, beat correction, and, combined with a pitch detection algorithm, it may give origin to advanced arrangement systems for electronic keyboard instruments. And more, due to its inherent real-time capabilities, we believe that what is proposed can represent a relevant tool and reference to explore new applicative contexts.

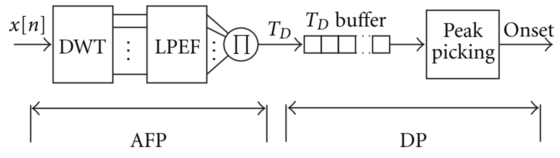

Figure 1: Overview of the proposed algorithm.

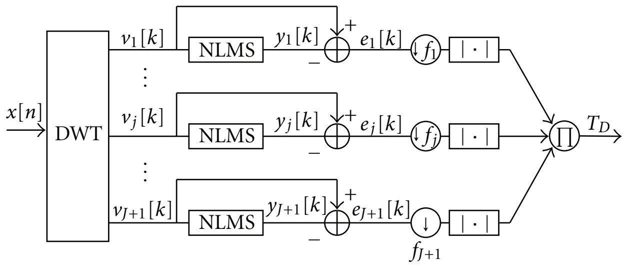

Figure 2: Block-scheme of the adaptive filtering process (AFP). The acronym NLMS stands for normalized LMS. The downsampling factors f_j are detailed in (9).

Here, the outline of the paper is given. In Section 2, the proposed onset detection algorithm is described in all its parts, also highlighting the differences with the algorithms analyzed in [8, 9]. Section 3 is devoted to the ways adopted to scale the original algorithm version down, whereas 4 discusses performed computer simulations and real-time capabilities of the proposed technique, for both addressed implementation schemes. In Section 5, conclusions are drawn and some future work ideas suggested.

2. The Onset Detection Algorithm

This section describes the high-end version of the algorithm. Simpler schemes are described in Section 3. The general scheme, adopted by all the versions of the algorithm, is depicted in Figure 1. The algorithm consists of two processes operating in cascade, namely, the adaptive filtering process (AFP) and the detection process (DP). These two are detailed in Figures 2 and 3.

The first one aims at the reduction of the original signal into the T_D signal which is a highly subsampled signal which manifests the occurrence of transients in the original signal. This part can be processed sample-wise or frame-wise, depending on the implementation requirements and must be executed in real-time. The output can be stored in a buffer for the DP to take place afterwards. This second processing aims to reveal onsets in the signal by a peak-picking procedure over the T_D signal.

The linear prediction error filters were first used in [8], where the filter bank was made of bandpass filters, with slightly different bandwidths, and all the subband signals were uniformly decimated. These were fed to the LPEFs

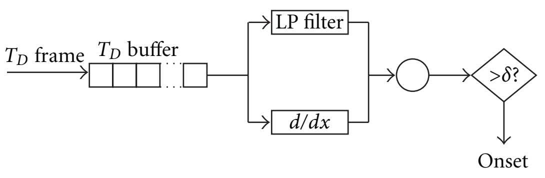

Figure 3: Block-scheme of the detection process (DP).

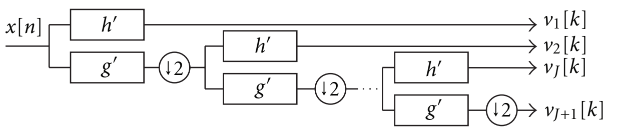

Figure 4: Flow graph representation of a J-level discrete wavelet Transform (analysis part).

(linear prediction error filters) and then summed together to obtain the T_D signal. In the proposed implementation, the AFP consists in a dyadic filter bank [13] based on wavelet filter coefficients and linear prediction error filters, one per channel. The output of every channel is then subsampled to one of the lowest channels sampling frequency and rectified. Finally the channels are multiplied together to form a single T_D signal. This last operation may be regarded as a form of MSP product [10, 11].

The DP collects many frames of the T_D signal in a buffer. When the buffer is full, it is smoothed and differentiated by filtering. Finally the difference signal and the smoothed signal are multiplied together. This operation has the meaning of weighting the difference signal, containing sharp peaks, with the amplitude of the original T_D signal. Previous works, such as [8] and later ones, suggested the use of median filtering to smoothen the T_D signal. This technique proves helpful, but is very time consuming on mathematical processors because it requires a sorting mechanism, which, in turn, requires branch instructions and can be hardly optimized.

2.1. The Adaptive Filtering Process.

The input signal is assumed to be single-channel. For common applications, a stereophonic signal may be available. Merging channels together into a single signal may result in a loss of spatial information, the worst case occurring when every channel contains different musical instruments. In this case, it is suggested to process the channels separately, if there are enough processing resources.

Once the signal has been gathered from input, in frames or sample-by-sample, it is fed to a dyadic filter bank [13], shown in Figure 4, which performs a multiresolution DWT analysis. The choice of a dyadic bank is motivated by the way it equally partitions wide-band signal energy into subbands. It can be shown experimentally that in a uniform filter bank higher-frequency bands gather fewer energy than lower-frequency bands, hence making senseless the prediction error over a signal which nearly contains no musical content.

As said, the employed multiband decomposition technique is the discrete wavelet decomposition, which can be seen as an octave band filter bank, with its analysis section (DWT properly) and its synthesis section (IDWT, inverse DWT) [14]. Moving from the following notation equalities:

$$\begin{aligned}
&\text{Upsampling:}\quad (\uparrow x)[2n] = x[n],\ (\uparrow x)[2n+1] = 0,\\
&\text{Downsampling:}\quad (\downarrow x)[n] = x[2n],\\
&\mathbf{G}\ (g_n \text{ low-pass filter})\quad (\mathbf{G}x)[n] = \sum_k x[k]\cdot g[n-k],\\
&\mathbf{H}\ (h_n \text{ high-pass filter})\quad (\mathbf{H}x)[n] = \sum_k x[k]\cdot h[n-k],
\end{aligned} \tag{1}$$

where $\tilde{g}[n] = g[-n]^*$ is the paraconjugate operator, and looking at the decomposition part, it can be observed that the original signal $x[n]$ is processed through filtering and down-sampling operations resulting in different sequences:

$$\begin{aligned}
v_j[k] &= \left\langle x[n], \tilde{h}'_j\left[n-2^j k\right]\right\rangle, \quad j = 1,\ldots,J,\\
v_{J+1}[k] &= \left\langle x[n], \tilde{g}'_J\left[n-2^J k\right]\right\rangle,
\end{aligned} \tag{2}$$

where $\tilde{g}'_j = (\tilde{\mathbf{G}}'\uparrow)_{j-1}\tilde{g}'$, $\tilde{h}'_j = (\tilde{\mathbf{G}}'\uparrow)_{j-1}\tilde{h}'$. Each of these sequences is relative to a precise part of spectrum of the signal and has different length than the others; in more specific words, it means that their scale and resolution values are divided by 2 at each decomposition level, consequently reducing of the same factor the sequence length and the part of spectrum they represent. This fact is directly related to the coverage of time/frequency plane existing in continuous wavelet transform (CWT). The signal can be reassembled from the coefficients through filtering and up-sampling operation:

$$x[n] = \sum_{j=1}^{J}\sum_k v_j[k]h_j\left[n-2^j k\right] + \sum_k v_{J+1}[k]g_J\left[n-2^J k\right], \tag{3}$$

where $g_j = (\mathbf{G}\uparrow)_{j-1}g$, $h_j = (\tilde{\mathbf{G}}'\uparrow)_{j-1}h$. However, since this filter bank is critically sampled, used filters are constrained to satisfy the following condition to achieve perfect reconstruction (without delay), here valid in case of simple two-band filter bank:

$$x[n] - \mathbf{G}\uparrow\downarrow\mathbf{G}'x[n] = \mathbf{H}\uparrow\downarrow\mathbf{H}'x[n]. \tag{4}$$

This can be easily extended to J-level decomposition case. Note that IDWT is not performed in our algorithmic scheme, but it is here described for the sake of completeness.

DWT has been also chosen for the fast convergence speed of LMS adaptive filters in the transformed domain [15] and for the short length of its filters impulse responses, which makes the dyadic bank filtering feasible with very short FIR filters. Time-domain FIR filters would instead require very long impulse responses for the aliasing to be negligible. It has been shown by our early experiments that the LMS prediction filters in the transformed domain converge, as expected, faster than the ones in the time domain with the same order.

The choice of the wavelet impulse response is not obvious, given the number of possibilities. In our case, the Coiflets functions [15] have been selected due to their properties, and also compared to other wavelet families such as the Daubechies' [15]. Biorthogonal wavelets have the advantage over orthogonal wavelets of being linear phase or nearly linear phase. This property is desired, that is why the Coiflets have been chosen. This allows for a close to perfect synchronization of the subband signals. The choice of Coiflets over other biorthogonal wavelets is their higher number of vanishing points for a given order which increases convergence properties [16]. As previously stated, the DWT filter bank adds different delays to each subsignal because of its asymmetric tree structure. Furthermore, if the wavelet filters are not symmetric, that is, not linear phase (or nearly linear phase), there is a spread of the group delay over frequency. To avoid this, we have chosen Coiflets which are known for their nearly linear phase property, that is,

$$\phi(\omega) = k\omega + o\left(\omega^N\right) \quad \text{for } \omega \longrightarrow 0. \tag{5}$$

Under this condition, the group delay can be approximated to the one of a linear phase filter, that is, $(N-1)/2$ samples, where N is the length of the FIR impulse response [15]. Once the filter delay is known, a simple synchronization mechanism can be designed to align all the subsignals by simply applying different delays for every channel. Since the delay increases while descending the decomposition tree and the sampling frequency decreases, the upper subsignals must be compensated with higher delays, according to the following:

$$d(j) = 2^{J+1-j} \sum_{i=1}^{J+1-j} \frac{N}{2^i}, \tag{6}$$

where $d(j)$ is the delay to be added to channel j. By adding such a delay to each channel, after the DWT filter bank, synchronization is granted in case the subchannels are all resampled to the same sampling frequency, as it is done at the end of the AFP process.

Instead of Coiflets, other wavelet families, with different properties, can be used. Although, in principle, the aforementioned mechanism works only for linear phase or nearly linear phase filters, it is shown by experimental results that the subsignals after a DWT filter bank with, for example, Daubechies' wavelets, which are minimum phase, can achieve a close to perfect synchronization because of the low group delay spread over frequency. With Daubechies' wavelets, the trade off schemes discussed in Section 3 can be more easily achieved as the filter order is proportional to a factor 2 instead of 6 as with Coiflets.

Once the signal has been filtered and decimated, the output of every channel is fed into LPEFs. The DWT-domain LPEF adaptation algorithm can be drawn from analogy with time-domain LMS-like algorithms. When computational power is not an issue, a common NLMS technique [16] can be applied to the DWT-domain case with a varying step size $\mu_j[k]$ according to the following:

$$\mu_j[k] = \frac{\mu'}{\left|\mathbf{u}_j[k]\right|^2 + c}, \tag{7}$$

where $0 < \mu' < 2$, $\mathbf{u}_j[k] = (v_j[k-1], v_j[k-2] \cdots v_j[k-L_o])$, L_o is the order of the LPEF, c is a small constant to avoid division by zero, and $|\cdot|$ stands for vector norm. The vector norm has the meaning of the estimate of the signal energy, which varies in time, making the step size varying as well. The NLMS technique is preferred to the Widrow-Hoff LMS algorithm for its suitability to signals with large energy variations, such as music. Such a variability occurs also in the DWT domain [15, 17], motivating the adoption of a stepsize normalization strategy within the adaptive filtering weight updating rule, which for jth channel filter coefficient vector $\mathbf{L}_j$ can be written as:

$$\begin{aligned} y_j[k] &= \mathbf{L}_j^T[k] \cdot \mathbf{u}_j[k], \\ e_j[k] &= v_j[k] - y_j[k], \\ \mathbf{L}_j[k+1] &= \mathbf{L}_j[k] + \mu_j[k] \cdot e_j[k] \cdot \mathbf{u}_j[k]. \end{aligned} \tag{8}$$

Since the DWT bank outputs have different sampling frequencies, the LPEF error signals $e_j[k]$ will have different sampling frequency as well. The LPEFs error signals $e_j[k]$ are suitable for heavy downsampling, hence they can all be subsampled, before rectification takes place as shown in Figure 2, to reach the sampling frequency of one of the highest-decimated channels. Smoothing before downsampling, can be avoided in most practical case, as $e_j[k]$ signals contain only peaks, which can be seen as a very low frequency content, on a very low noise floor. In the case the channels are decimated to the sampling frequency of the highest-decimated channel, that is, $(J+1)$th channel, the downsampling factor for the jth channel is:

$$f_j = 2^{J+1-j}. \tag{9}$$

In case the sampling frequency for the channels is chosen to be higher than that of channel $(J+1)$, (9) can be easily generalized accordingly. The subsampled signals $e'_j[l]$ can now be multiplied sample-by-sample together in a MSP-like way, according to the following formula:

$$T_D[l] = \prod e'_j[l] \tag{10}$$

obtaining a sharp T_D signal. To avoid multiplying by zero or near-zero, the least energetic channels are discarded. The resulting T_D signal is then stored in a buffer for the peak extraction by the DP.

2.2. The Detection Process.

The DP works on a number of T_D frames stored in a buffer. The size of this buffer must be as long as possible while allowing for real-time performances. For many applications, the real-time constraint may be the temporal masking phenomenon, which lasts approximately

20 ms [18]. Also, musical instruments involving user feedback or any kind of device with haptic interfaces may require slightly lower delays. The major time constraint of the whole system is the DP buffer length. Only when the buffer is full, the DP can take place. The DP consists in a linear processing, and a maximum search, which is the only heuristic peak extraction decision used by the proposed algorithm.

The goal consists in obtaining a signal that clearly shows remarkable peaks where strong transients in the T_D signal rise, so to allow the use of a single threshold, which is the simplest decision scheme possible. A complex heuristic scheme would otherwise badly affect the algorithm efficiency on mathematic processors. Two thresholds would be needed for good detection capabilities: one on the steepness of the transient (slow transients are unlikely to be note onsets), and one on the energy of the signal (transients under a low energy threshold are unlikely to be onsets). The sole differentiation operation is not robust enough to eliminate noise peaks or bursts, hence the signal must be weighted with a sort of estimate of the energy of the signal near the localization of these peaks. The most efficient way to do this, though not much accurate, is the weighting of the difference signal with a lowpass version of the T_D signal. Hence the first stage of the DP is made of a very low cut-off frequency lowpass filter and a differentiator filter both applied to the incoming T_D signal. The former can have a low order if designed as an IIR filter, while the latter consists in a single delay tap and a subtraction. The output signals from these two filters are multiplied together:

$$T_P[l] = T'_D[l] \cdot T''_D[k], \qquad (11)$$

where $T'_D[l]$ is the differentiated version of $T_D[l]$ and $T''_D[l]$ is the low pass filtered version of $T_D[l]$. The heuristic process finds the maximum of the function and compares it with a threshold: if the maximum is higher, an onset has been found and the DP flags the discovery and sends its timestamp to other tasks for further processing,

$$\text{onset-flag} = \text{MAX}(T_P) > \delta. \qquad (12)$$

This last stage is very simple, and most DSPs have hand-optimized assembly functions for vector maximum value search. There is no need to look for near onsets or keep record of recently discovered onsets since it is assumed that the length of the T_D buffer used by the DP is approximately as long as the audible temporal masking threshold. Any other onset with a lower energy than the found one will be discarded.

3. Scaling Down of the Algorithm

A detailed description of the general detection framework has been thus provided in the previous section, with details given for a high-end version of the algorithm. Scaled-down versions may be necessary when low-power DSPs or RISC processors are used. In Section 4, we discuss the implementation of two versions of the algorithm, namely Scheme1 and Scheme2, the former being a high-end version targeted for maximum detection performances, the latter being a scaled-down version aiming to a good detection with lower computational cost. These two are detailed in Table 1. Practical implementation of the algorithm may make a compromise between Scheme1 and Scheme2. We provide here some useful options to scale down the high-end algorithm provided in Section 2, which brings to reduced computational cost, indicating separately for each option the decrease in computational cost and detection performances.

Table 1: Processing schemes details.

	Scheme1	Scheme2
Filter bank levels	8	6
Wavelet filters	coif4 (order = 24)	coif2 (order = 12)
LPEF Adapt. Algo	NLMS	Sign-Error LMS
LPEFs order	$L_j = 10 + 2(j - 1)$	$L_j = 10$
Computational cost	$3W_o + {}+(4L_8 + 2L_7 + L_6 \cdots)$	$3/4W_o + 2L_o$
MFLOPS	13.8	1.3

The subband decomposition scheme seen in Figure 2 presents several degrees of freedom for the developer:

(i) the DWT FIR filter length can be shortened with smoothly degrading performances. As with FIR filters, the operations required are $O[N]$, and the lower sampling frequency channels have lesser impact on the operations per sample; the computational cost has a linear decrease with the filter order decrease. The detection performances are slightly affected, for example, by halving the filter order, the loss is of a maximum of 4% F-measure points (the detection performances are evaluated through the F-measure index, see (14) in Section 4 for further details),

(ii) the number of analysis levels may be reduced with respect to Scheme1. This may largely contribute to a decrease in detection performances while the increase in processing speed is low, as the lower channels have lower sampling frequencies, hence they do not impact heavily on the operations required,

(iii) the filter bank may be applied to a decimated version of the signal, assuming that the musical content at higher frequencies is negligible, compared to lower frequencies. This of course may greatly speed up the processing, without a fair decrease in detection performances. Experimental tests show that a 2× decimation on the input signal allows to save at least 1/4 of the processing needed for the DWT bank (the theoretic saving of 1/2 is not achieved because of the processing overhead and the need for a high-order IIR antialiasing filter), while detection performances decrease is unnoticeable.

LPEFs can be scaled down as well:

(i) the order of the filter can be reduced. Experimental tests show that when increasing the filter order over a suggested value of 10, the increase in detection performances is very low (close to 1%). However for the higher-end implementation we decided to use higher order LPEFs, with the order proportional to the band index so that wider bandwidth subsignals have a higher-order LPEF. Since NLMS filters require $O[N^2]$ operations per sample, a slight decrease of the order allow for high computational savings,

(ii) the updating rule, which is the most expensive part of the LMS algorithm, can be changed to an LMS-like technique which allow for a lower computational complexity at the expenses of the convergence speed. In our experiments, the Sign-Error LMS [19] proved accurate enough to guarantee good convergence speed and accuracy. The Sign-Error LMS algorithm differs from the NLMS algorithm only for the coefficients update rule, which varies according to the following:

$$\mathbf{L}_j[k+1] = \mathbf{L}_j[k] + \mu_j[k] \cdot \operatorname{sign}\Big(e_j[k]\Big)\mathbf{u}_j[k]. \qquad (13)$$

This may be a good solution on RISC processors where a multiplication is substituted by other instructions such as bit manipulations (depending on the processor instruction set and data format). On a total $4 \cdot L_o$ operations needed by the NLMS per sample, this solution allows to avoid L_o multiplications. Regarding the detection performances decrease, the Sign-Error LMS can cause a loss of 3% to 4%F-measure points.

Note that other modifications on adaptive filtering operations might be applied for scaling purposes: for instance the vector norm in (7) could be replaced by a 1st-order low-pass filtering. We can observe that, for our implementation target, the presence of optimized DSP library makes the vector norm not computationally demanding (especially if performed on a low number of samples, like in our case study), leading us to prefer investigating the Sign-Error LMS scaling approach and leave the other to future developments.

Furthermore, if a mixed DSP-RISC platform is available—this is often the case for consumer electronics, musical equipment, and multimedia devices—the parallelism of the platform may be exploited so that the DP can take place onto the RISC core, while the DSP core processes the signal. It is required, hence, that the DP completes only once every time the T_D buffer is full. This provides headroom for more processing on the DSP core.

4. Experimental Tests

Two schemes of the algorithm were implemented: Scheme1, allowing for maximum performances, and Scheme2, aimed at the lowest computational effort. Both have been implemented using Matlab, to give upper and lower bounds of the onset detection performances and on a Texas Instruments TMS320C6000 DSP, for a proof-of-concept design. Some profiling results will be given in Section 4.3.

4.1. The TI Target Platform. In order to practically evaluate the efficiency of the proposed method, a porting of the algorithm has been performed, from the original Matlab code to a DSP evaluation board. The board used for this purpose is an OMAPL137, which combines a RISC core with the floating-point C6747 processor. Only the floating-point core was used for the experiments, although further work will include embedding part of the algorithm on the RISC core. The C6747 core runs at 300 MHz, and has two separate data paths, as shown in Figure 5, each one including four different instruction units (M, S, L, D units, each one responsible for one of the following: multiply, arithmetic, logical/branch, and load instructions) which can efficiently execute in parallel if the code has been optimized. This allows, at best, to employ eight operations per clock cycle, two of which are MAC instructions used to implement digital filters.

The operating system used for the current implementation is the DSP/BIOS 5.41 from Texas Instruments, which provides user-level APIs, low-level drivers, and other common features such as tasks, semaphores, queues, and so forth. The operating system allows the C6747 core itself to manage onboard peripherals, so that the RISC core is not strictly required and thus the prototyping of a DSP-only application is made faster.

4.2. The Two Proposed Schemes and Computational Cost. The implementation schemes proposed hereby are summarized in Table 1. The theoretical cost in MFLOPS (Million FLoating point OPerations per Second) of the two schemes is reported as well. The LPEF order in Scheme1 depends on the channel index j, so that the channels at higher sampling frequencies have higher order LPEFs and the lowest channel has order 10. All the LPEFs in Scheme2 have a fixed order of 10. This lower bound is considered safe after experimental evaluation. In Scheme1, the LPEFs output error subsignals are resampled at the sampling frequency of channel 7, namely 689 Hz, while for Scheme2 they are subsampled to the lowest channel sampling frequency, which is approximatively 1378 Hz. The heavy subsampling performed after the LPEFs allows to neglect the computational cost of the rest of the algorithm. From Table 1, it can be noted that there is an order of magnitude between the two proposed schemes in terms of computational cost. This difference between the two, however, is reduced in a practical implementation due to overheads, and additional procedures for managing buffers, data, and so forth. Some relevant aspects about Table 1 are here detailed: lowest and highest levels discarded in Scheme1; 4 decimation factor preceding the bank in Scheme2; j is the channel index (higher frequencies channels have lower index j); W_o is the wavelet filter order, L_o is the LPEF order; filter bank polyphase implementation has been used for each approximation-details filter pair separately; FLOPS are calculated at sampling frequency equal to 44.1 kHz.

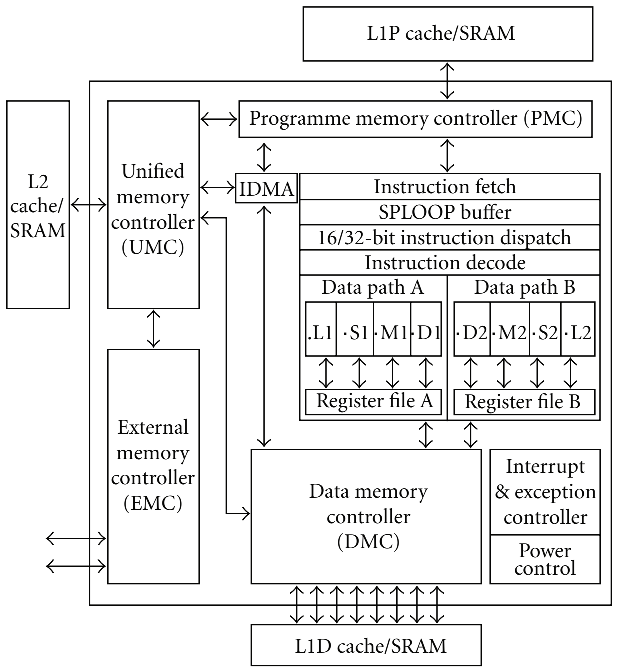

FIGURE 5: The TMS320C6747 DSP Megamodule architecture, featuring separate program and data buses, two separate data paths, with L, S, M, D instruction units. Adapted from [20]. Courtesy: Texas Instruments Incorporated. All rights reserved.

TABLE 2: Computer simulation results against the Leveau database.

	PNP	Complex mixtures	Others	Total
Proposed Scheme1	86.6%	90.6%	81.2%	85.5%
Proposed Scheme2	71.6%	73.7%	71.1%	72.3%
Linear prediction [8]	83.1%	86.4%	72.8%	78.9%
Improved linear prediction [9]	87.6%	87.1%	74.9%	82.0%
Complex domain [21]	71.9%	81.2%	63.1%	70.1%
Perceptual spectral flux [22]	46.1%	46.4%	46.7%	46.6%

TABLE 3: Detection results against the four drum tracks for both the computer simulations and the C6747 runs.

	Drums excerpts
Proposed Scheme1	92.9%
Proposed Scheme2	77.1%
C6747 Implementation Scheme1	90.5%
C6747 Implementation Scheme2	74.6%

4.3. Computer Evaluation. Computer simulations have been performed on a publicly available music database [12] to allow direct comparison of the proposed schemes to previous works. The public database contains 17 files, for a total of 744 hand-labelled onsets. Unfortunately, the Leveau database does not contain any NPP (nonpitched percussive) solo tracks (e.g., drums). We added 4 drums and percussions tracks for a total of 214 onsets to evaluate the performances of the proposed methods with NPP sounds. These results are reported in Table 3. In our experiments, automatically detected onsets are compared with hand-labeled reference onsets and they are considered correct if the distance between the detected and the reference onset is less than 50 ms. This slight margin allows for hand-labeling inaccuracy.

To evaluate the algorithm a metric based on CD (correct detections), FP (false positives), and FN (false negatives) is used, resulting in the following final parameter:

$$\text{F-measure} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}, \qquad (14)$$

where

$$\text{Precision} = \text{CD}/(\text{CD} + \text{FP}),$$
$$\text{Recall} = \frac{\text{CD}}{\text{CD} + \text{FN}}. \qquad (15)$$

Test results are provided in Table 2 for the Leveau database and in Table 3 for the NPP tracks. The former results are

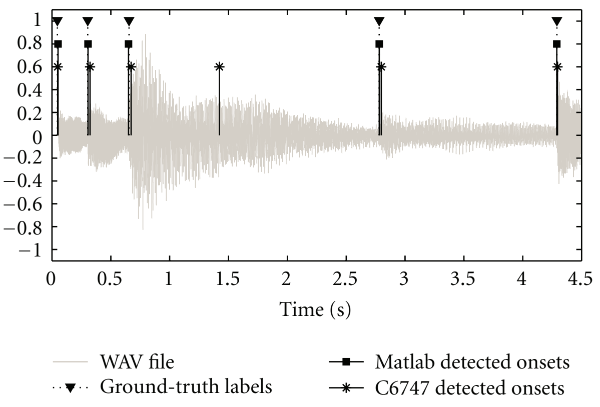

Figure 6: Comparison of the the onsets detected by the algorithm run on Matlab and on the C6747 platform for a portion of audio from Leveau's database against ground-truth labelled onsets.

divided in pitched non-percussive (PNP) tracks, complex mixtures, and others. These results are compared to the F-measure of several other techniques provided in [9] based on the same music database. Comparison of these results shows that the Scheme1 algorithm outperforms previous schemes, while Scheme2, together with a lower computational complexity, obtains results comparable or superior to other techniques taken here as reference. Tests have been performed also with the DWT filter bank based on the Daubechies' wavelets instead of the Coiflets. The Daubechies' wavelet filters used are of the type db32 for Scheme1 and db8 for Scheme2. Results are very close to the ones obtained with the Coiflets: 85.2% for Scheme1 and 72.4% for Scheme2 with Daubechies' wavelets.

4.4. Target Implementation Details and Results. Section 4.3 proves the good capabilities of the proposed schemes, and shows a graceful degradation of performances between the higher-end Scheme1 and the lower-end Scheme2. In this section, we provide experimental results on the detection capabilities and execution speed of the algorithms together with the computational cost reduction obtained by scaling Scheme1 down to Scheme2. It must be noted that computer simulation results stand as a higher bound reached by the proposed algorithm, whereas the embedded target detection results give merit to the algorithm implementation showing results which are close to the computer simulations although slightly lower (approximately 1.9% less in average for both Scheme1 and Scheme2), due to practical issues, such as the single-precision arithmetic (whereas the Matlab implementation makes use of double-precision arithmetic). These results are provided after a first implementation of the algorithms into a TMS320C6000 DSP and may improve in the future. Figure 6 shows a portion of a track from Leveau's database with labelled onsets, Matlab-detected onsets and the onset timestamps coming from the board.

The rationale behind the porting of the algorithm to an embedded device is to validate its design and principles, to assist the tuning of the algorithm parameters, and to gather an insight on problematics related to real-time.

Table 4: Detection results of the C6747-based implementation for Scheme1 and Scheme2 against the Leveau database.

	PNP	Complex mixtures	Others	Total
Scheme1	85.3%	88.7%	77.5%	83.2%
Scheme2	69.8%	75.2%	70.3%	70.8%

Table 5: Profiling for Scheme1 on the C6747 processor. Execution times and CPU load data are averaged over multiple observations.

Frame length	128	256	512
Execution time (AFP)—no opt	0.8 ms	1.42 ms	2.7 ms
Execution time (AFP)—opt	0.26 ms	0.44 ms	0.99 ms
Execution time (DP)—no opt	0.04 ms	0.06 ms	0.16 ms
Execution time (DP)—opt	0.02 ms	0.03 ms	0.04 ms
CPU load (AFP+DP)—no opt	34.5%	30.7%	27.9%
CPU load (AFP+DP)—opt	10.2%	9.5%	8.9%

Furthermore, execution times and processing overhead have been evaluated. To the theoretical computational cost, in fact, some overhead must be often added. This overhead may be introduced by buffers manipulation, memory read/write operations, and so forth. The number of all these operations can be severely increased when memory must be saved, a common situation in embedded processors programming. We are not taking into account the extra cost of drivers, operating system, and so forth, which is subject to considerable variations depending on software stack implementation and platforms. In the results, we evaluate below, we only consider the overhead strictly related to the algorithm itself.

The implementation has been performed with single-precision arithmetic, in order to preserve memory. However, a double-precision version of the code could be easily implemented. The code has been written in ANSI C language. The benefits of this choice are its faster prototype development, the possibility to use function libraries provided by TI, and a fair comparison with other algorithms in terms of speed. In fact, hand-written assembly code would obtain faster execution speed, but the results would be platform dependent, hence meaningless in this context.

Overall detection results are provided in Table 4 for both Scheme1 and Scheme2 implementations. As the Table shows, these are similar to Matlab results shown in Table 2. The tests were conducted with a fixed input volume, evaluated empirically through a training data set.

Execution time results are provided in Tables 5 and 6 for both Scheme1 and Scheme2 implementations. The results are specific for the AFP and the DP, showing absolute execution time and average CPU load percentage. As mentioned above, the processor is a Texas Instruments C6747, running at 300 MHz. Input signal is acquired at a 44.1 KHz sampling frequency in stereo, but only one channel is processed in order to evaluate the results against the same

Table 6: Profiling for Scheme2 on the C6747 processor. Execution times and CPU load data are averaged over multiple observations.

Frame length	128	256	512
Execution time (AFP)—no opt	0.5 ms	0.9 ms	1.71 ms
Execution time (AFP)—opt	0.13 ms	0.28 ms	0.57 ms
Execution time (DP)—no opt	0.04 ms	0.06 ms	0.16 ms
Execution time (DP)—opt	0.02 ms	0.03 ms	0.04 ms
CPU load (AFP+DP)—no opt	21.9%	17.1%	16.2%
CPU load (AFP+DP)—opt	6.0%	5.4%	5.1%

music database used for computer simulations. The signal is processed by frames, with a frame length of 256 being the best trade-off between reactivity of the system, overhead reduction and memory consumption; however, tests are conducted also on frames of size 128 and 512, to show how the execution time increases/reduces. As an example, if the frame length is 256 samples and the target latency is approximately 20 ms, four T_D frames can be stored for the DP task to handle them.

The results are provided for optimized and nonoptimized versions of the code for both Scheme1 and Scheme2. In the first case, no compiler optimization is used (in order to have an upper bound to the execution speed), whereas in the second one, the C code presents calls to the Texas Instruments DSPlib function library, some preprocessor pragmas to help the compiler optimize the code and function-level compiler optimization [23].

The results show a practical decrease of the execution times by nearly a factor two between Scheme1 and Scheme2 implementations, and a decrease of execution times of a factor 3 between the nonoptimized Scheme2 and the optimized Scheme2. Results on optimized code should be taken as a higher bound for speed, because in a production environment source code is always compiled with optimizations. However, these can largely vary according to a number of practical factors, including hardware specifications, hence nonoptimized results are given for scientific purposes.

A system such the one described hereby can be used with one-channel input signal to build a query-by-humming tool or instantaneous instrument transcription scenario, but in other cases, multiple channels may be used. This could be the case of more inventive sound interaction tools, that could even need a fast haptic feedback for the user. Given the light computational cost of the algorithm, it is easy to implement two separate instances of the algorithm to detect onsets on a stereo source. The detection capabilities of the algorithm should then increase due to the stereophonic source. By doubling the processing instances, the computational cost should double, but by rewriting a single instance to work on both channels, the computational cost could be reduced, by taking advantage of the two parallel processing units available on most DSPs. This will be object of further implementations.

5. Conclusions

In this paper, a novel musical onset detection algorithm with particular aim to real-time applications has been presented. This method has been especially designed for direct implementation on embedded-processors, and attention has been paid to keep the computational cost as low as possible. This method provides good results in both detection capabilities and speed performances, keeping the former as high as most complex algorithms and the latter as low as the simplest ones. A proof-of-concept design on a Texas Instruments TMS320C6000 DSP processor has been developed to assist the tuning of the algorithm parameters and to validate its design and principles. This allowed us to positively conclude about the flexibility of the proposed musical onset detection approach, a property extremely useful in real-time system deployment phase. Experimental results have been provided in both implementation case studies addressed: computer and embedded-processor based.

Future work will include the study of a prewhitening filter to improve the LPEFs convergence speed and a complex AGC (automatic gain control) system to allow robust detection in any environment condition. Parallel computing may be exploited to implement multichannel onset detection or to improve detection results by comparing the output of different onset detection methods. Frequency-domain adaptive filtering algorithms may be experimented as well in order to obtain faster implementations, at the expense of losing the logarithmically-spaced representation of the signal spectrum. We also plan to evaluate our methods against the MIREX database [24] for further comparison.

References

[1] 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, 1998.

[2] V. Bruni, S. Marconi, and D. Vitulano, "Time-scale atoms chains for transients detection in audio signals," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 18, no. 3, pp. 420–433, 2010.

[3] M. Marolt, A. Kavcic, and M. Privosnik, "Neural networks for note onset detection in piano music," in *Proceedings of the International Computer Music Conference*, Gothenberg, Sweden, 2002.

[4] A. Lacoste and D. Eck, "Onset detection with artificial neural networks for MIREX," in *Proceedings of the Music Information Retrieval Evaluation EXchange (MIREX '05)*, London, UK, 2005.

[5] W. Lee, Y. Shiu, and C. Kuo, "Musical onset detection with linear prediction and joint features," in *Proceedings of the Music Information Retrieval Evaluation EXchange (MIREX '07)*, Vienna, Austria, 2007.

[6] E. Kapanci and A. Pfeffer, "A hierarchical approach to onset detection," in *Proceedings of the International Computer Music Conference*, pp. 438–441, 2006.

[7] G. Tzanetakis, "MARSYAS submissions to MIREX 2009," in *Proceedings of the Music Information Retrieval Evaluation EXchange*, Kobe, Japan, 2009.

[8] W. C. Lee and C. C. J. Kuo, "Musical onset detection based on adaptive linear prediction," in *Proceedings of the IEEE*

International Conference on Multimedia and Expo (ICME '06), pp. 957–961, July 2006.

[9] W. C. Lee and C. C. J. Kuo, "Improved linear prediction technique for musical onset detection," in *Proceedings of the International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP '06)*, pp. 533–536, usa, December 2006.

[10] B. M. Sadler and A. Swami, "Analysis of multiscale products for step detection and estimation," *IEEE Transactions on Information Theory*, vol. 45, no. 3, pp. 1043–1051, 1999.

[11] M. A. B. Messaoud, A. Bouzid, and N. Ellouze, "Spectral multi-scale analysis for multi-pitch tracking," in *Proceedings of the IEEE 13th Digital Signal Processing Workshop and 5th IEEE Signal Processing Education Workshop (DSP/SPE '09)*, pp. 26–31, usa, January 2009.

[12] P. Leveau and L. Daudet, "Methodology and tools for the evaluation of automatic onset detection algorithms in music," in *Proceedings of the International Symposium on Music Information Retrieval (ISMIR '04)*, pp. 72–75, 2004.

[13] P. P. Vaidyanathan, *Multirate Systems and Filter Banks*, Prentice-Hall, Upper Saddle River, NJ, USA, 1993.

[14] O. Rioui, "Discrete-time multiresolution theory," *IEEE Transactions on Signal Processing*, vol. 41, no. 8, pp. 2591–2606, 1993.

[15] N. Erdol and F. Basbug, "Wavelet transform based adaptive filters: analysis and new results," *IEEE Transactions on Signal Processing*, vol. 44, no. 9, pp. 2163–2171, 1996.

[16] S. Haykin, *Adaptive Filter Theory*, Prentice Hall, New York, NY, USA, 4th edition, 2001.

[17] S. Attallah and M. Najim, "On the convergence enhancment of the wavelet transform based LMS," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP '95)*, vol. 1, pp. 973–976, 1995.

[18] E. Zwicker and H. Fastl, *Psychoacoustics: Facts and Models*, Springer, Berlin, Germany, 2nd edition, 1999.

[19] M. H. Hayes, *Statistical Digital Signal Processing and Modeling*, John Wiley & Sons, New York, NY, USA, 1996.

[20] "Tms320c674x dsp cpu and instruction set reference guide," October 2008.

[21] J. P. Bello, C. Duxbury, M. Davies, and M. Sandler, "On the use of phase and energy for musical onset detection in the complex domain," *IEEE Signal Processing Letters*, vol. 11, no. 6, pp. 553–556, 2004.

[22] K. Jensen, "Causal rhythm grouping," in *Proceedings of the International Symposium on Music Information Retrieval (ISMIR '04)*, Esbjerg, Denmark, May 2004.

[23] "TMS320C6000 optimizing compiler v 7.2 user's guide," January 2011, http://focus.ti.com/lit/ug/spru187s/spru187s.pdf.

[24] "Mirexdatabase," 2010, http://www.music-ir.org/mirex/.