Real-Time Detection of Musical Onsets with Linear Prediction and Sinusoidal Modelling

John Glover, Victor Lazzarini and Joseph Timoney

The Sound and Digital Music Research Group
National University of Ireland, Maynooth
Ireland

John.C.Glover@nuim.ie
Victor.Lazzarini@nuim.ie
JTimoney@cs.nuim.ie

Abstract

Real-time musical note onset detection plays a vital role in many audio analysis processes, such as score following, beat detection and various sound synthesis by analysis methods. This paper provides a review of some of the most commonly used techniques for real-time onset detection. We suggest ways to improve these techniques by incorporating linear prediction, as well as presenting a novel algorithm for real-time onset detection using sinusoidal modelling. We provide comprehensive results for both the detection accuracy and the computational performance of all of the described techniques, evaluated using *Modal*, our new open source library for musical onset detection, which comes with a free database of samples with hand-labelled note onsets.

1 Introduction

Many real-time musical signal processing applications depend on the temporal segmentation of the audio signal into discrete note events. Systems such as score followers [1] may use detected note events to interact directly with a live performer. Beat-synchronous analysis systems [2, 3] group detected notes into beats, where a beat is the dominant time unit or metric pulse of the music, then use this knowledge to improve an underlying analysis process.

In sound synthesis by analysis, the choice of processing algorithm will often depend on the characteristics of the sound source. Spectral processing tools such as the Phase Vocoder [4] are a well established means of time-stretching and pitch-shifting harmonic musical notes, but they have well documented weaknesses in dealing with noisy or transient signals [5]. For real-time applications of tools such as the Phase Vocoder, it may not be possible to depend on any prior knowledge of the signal to select the processing algorithm, so we must be able to identify transient regions on-the-fly in order to reduce synthesis artifacts. It is within this context that onset detection will be studied in this paper.

While there have been several recent studies that examine musical note onset detection [6, 7, 8], there have been few that analyse the real-time performance of the published techniques. One of the aims of this paper is to provide such an overview. In Section 2 some of the common onset detection techniques from the literature are described. In Section 3.1 we suggest a way to improve on these techniques by incorporating linear prediction [9]. In Section 4.1, we present a novel onset detection method that uses sinusoidal modelling [10]. Section 5.1 introduces *Modal*, our new open source library for musical onset detection. This is then used to evaluate all of the previously described algorithms, with the results given in Sections 5.2 and 5.3, and then discussed in Section 5.4. This evaluation includes details of the performance of all of the algorithms, in terms of both accuracy and computational requirements.

2 Real-Time Onset Detection

2.1 Definitions

This paper distinguishes between the terms *audio buffer* and *audio frame* as follows:

Audio Buffer: A group of consecutive audio samples taken from the input signal. The algorithms in this paper all use a fixed buffer size of 512 samples.

Audio Frame: A group of consecutive audio buffers. The algorithms described here all operate on overlapping, fixed-sized frames of audio. These frames are 4 audio buffers (2048 samples) in duration, consisting of the most recent audio buffer which is passed directly to the algorithm, combined with the previous 3 buffers which are saved in memory. The start of each frame is separated by a fixed number of samples that is equal to the buffer size.

In order to say that an onset-detection system runs in *real-time*, we require two characteristics:

1. **Low Latency:** The time between an onset occurring in the input audio stream and the system correctly registering an onset occurrence must be no more than 50 ms. This value was chosen in order to allow for the difficulty in specifying reference onsets, which is described in more detail in Section 2.1.1. All of the onset detection schemes that are described in this paper have latency of 1024 samples (the size of two audio buffers), except for the peak amplitude difference method (given in Section 4.3) which has an additional latency of 512 samples, or 1536 samples of latency in total. This corresponds to latency times of 23.2 ms and 34.8 ms respectively, at a sampling rate of 44.1 kHz. The reason for the 1024 sample delay on all onset detection systems is explained in Section 2.2.2, while the cause of the additional latency for the peak amplitude difference method is given in Section 4.3.

2. **Low Processing Time:** The time taken by the algorithm to process one frame of audio must be less than the duration of audio that is held in each buffer. As the buffer size is fixed at 512 samples, the algorithm must be able to process a frame in 11.6 ms or less when operating at a sampling rate of 44.1 kHz.

It is also important to draw a distinction between the terms *onset*, *transient* and *attack* in relation to musical notes. This paper follows the definitions given in [6], summarised as follows:

Attack: The time interval during which the amplitude envelope increases.

Transient: A short interval during which the signal evolves in a relatively unpredictable way. It often corresponds to the time during which the excitation is applied then dampened.

Onset: A single instant marking the beginning of a transient.

2.1.1 The Detection Window

The process of verifying that an onset has been correctly detected is not straightforward. The ideal situation would be to compare the *detected onsets* produced by an onset detection system with a list of *reference onsets*. An onset could then be said to be correctly detected if it lies within a chosen time interval around the reference onset, referred to here as the *detection window*. In reality, it is difficult to give exact values for reference onsets, particularly in the case of instruments with a soft attack such as the flute or bowed violin. Finding reference onsets from natural sounds generally involves human annotation of audio samples. This inevitably leads to inconsistencies, and it was shown in [11] that the annotation process is dependent on the listener, the software used to label the onsets and the type of music being labelled. In [12] Vos and Rasch make a distinction between the *Physical Onset Time* and the *Perceptual Onset Time* of a musical note, which again can lead to differences between the values selected as reference onsets, particularly if there is a mixture of natural and synthetic sounds. To compensate for these limitations of the annotation process, we follow the decision made in a number of recent studies [6, 7, 8] to use a detection window that is 50 ms in duration.

2.2 The General Form Of Onset Detection Algorithms

As onset locations are typically defined as being the start of a transient, the problem of finding their position is linked to the problem of detecting transient intervals in the signal. Another way to phrase this is to say that onset detection is the process of identifying which parts of a signal are relatively unpredictable.

2.2.1 Onset Detection Functions

The majority of the algorithms described in the literature involve an initial data reduction step, transforming the audio signal into an *Onset Detection Function* (ODF), which is a representation of the audio signal at a much lower sampling rate. The ODF usually consists of one value for every frame of audio, and should give a good indication as to the measure of the unpredictability of that frame. Higher values correspond to greater unpredictability. Figure 1 gives an example of a percussive audio sample together with an ODF calculated using the spectral difference method (see Section 2.3.2 for more on this technique).

2.2.2 Peak Detection

The next stage in the onset detection process is to identify local maxima, also called *peaks*, in the ODF. The location of each peak is recorded as an onset location if the peak value is above a certain threshold. While peak picking and thresholding are described elsewhere in the literature [13], both require special treatment in order to operate with the limitations of strict real-time operation (defined in Section 2.1). As this paper focuses on the evaluation of different ODFs in real-time, the peak picking and thresholding processes are identical for each ODF.

When processing a real-time stream of ODF values, the first stage in the peak detection algorithm is to see if the current value is a local maxima. In order to make this assessment, the current ODF value must be compared to the two

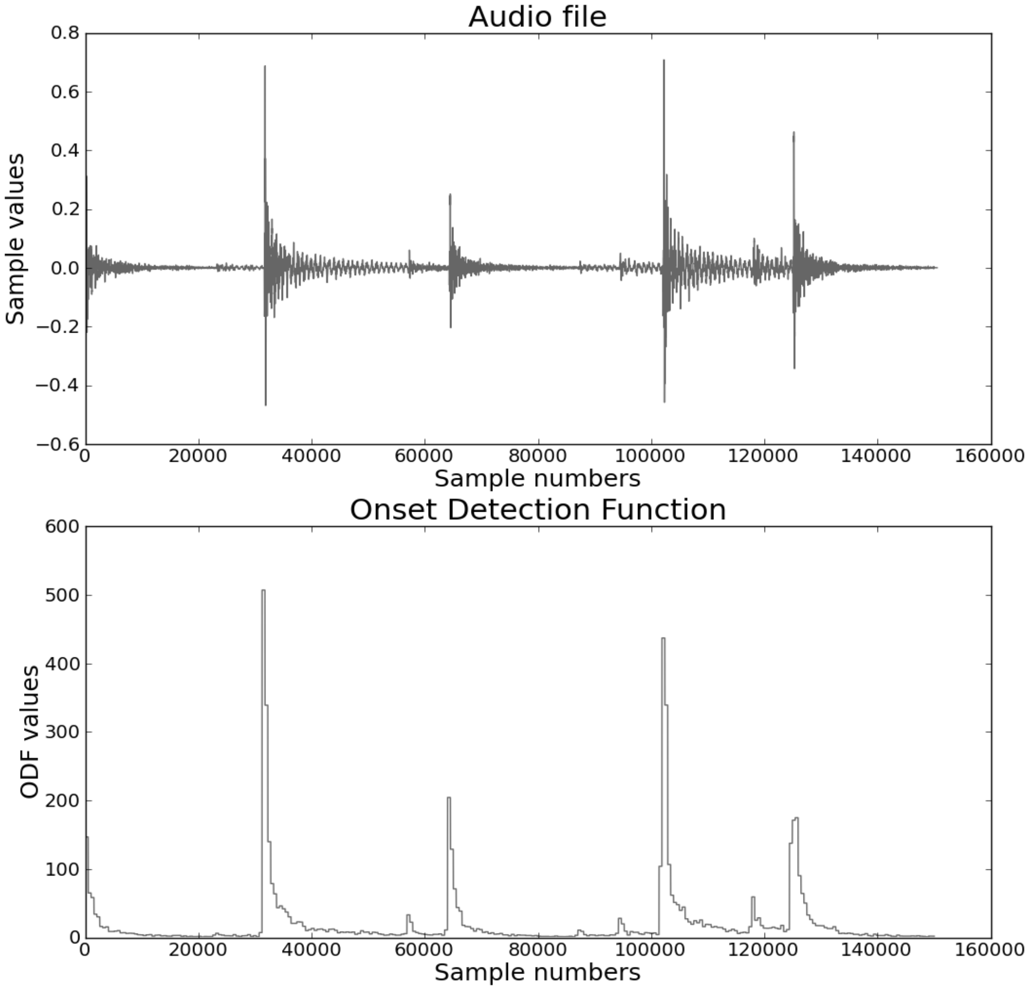

Figure 1: Percussive audio sample with ODF generated using the spectral difference method.

neighbouring values. As we cannot "look ahead" to get the next ODF value, it is necessary to save both the previous and current ODF values and wait until the next value has been computed in order to make the comparison. This means that there must always be some additional latency in the peak picking process, in this case equal to the buffer size which is fixed at 512 samples. When working with a sampling rate of 44.1 kHz, this results in a total algorithm latency of two buffer sizes or approximately 23.2 ms. The process is summarised in Algorithm 1.

```
Input: ODF value
Output: Whether or not previous ODF value represents a peak (Boolean)
$IsOnset \longleftarrow False$
if PreviousValue $>$ CurrentValue and PreviousValue $>$ TwoValuesAgo then
    if PreviousValue $>$ CalculateThreshold() then
        $IsOnset \longleftarrow True$
    end
end
$UpdatePreviousValues()$
return IsOnset
```

Algorithm 1: Real-time peak picking (one buffer delay)

2.2.3 Threshold Calculation

Thresholds are calculated using a slight variation of the median/mean function described in [14] and given by (1), where σ_n is the threshold value at frame n, $O[n_m]$ is the previous m values of the ODF at frame n, λ is a positive median weighting value and α is a positive mean weighting value.

$$\sigma_n = \lambda \times \mathit{median}(O[n_m]) + \alpha \times \mathit{mean}(O[n_m]) + N \tag{1}$$

The difference between (1) and the formula in [14] is the addition of the term N, which is defined as

$$N = w \times v \tag{2}$$

where v is the value of the largest peak detected so far and w is a weighting value. For indefinite real-time use it is advisable to either set $w = 0$ or to update w at regular intervals in order to account for changes in dynamic level. Figure 2 shows the values of the dynamic threshold (green dashes) of the ODF given in Figure 1, computed using $m = 7$, $\lambda = 1.0$, $\alpha = 2.0$ and $w = 0.05$. Every ODF peak that is above this threshold (highlighted in Figure 2 with red circles) is taken to be a note onset location.

2.3 Onset Detection Functions

This section reviews several existing approaches to creating ODFs that can be used in a real-time situation. Each technique operates on frames of *N* samples, with the start of each frame being separated by a fixed buffer size of *h* samples. The ODFs return one value for every frame, corresponding to the likelihood of

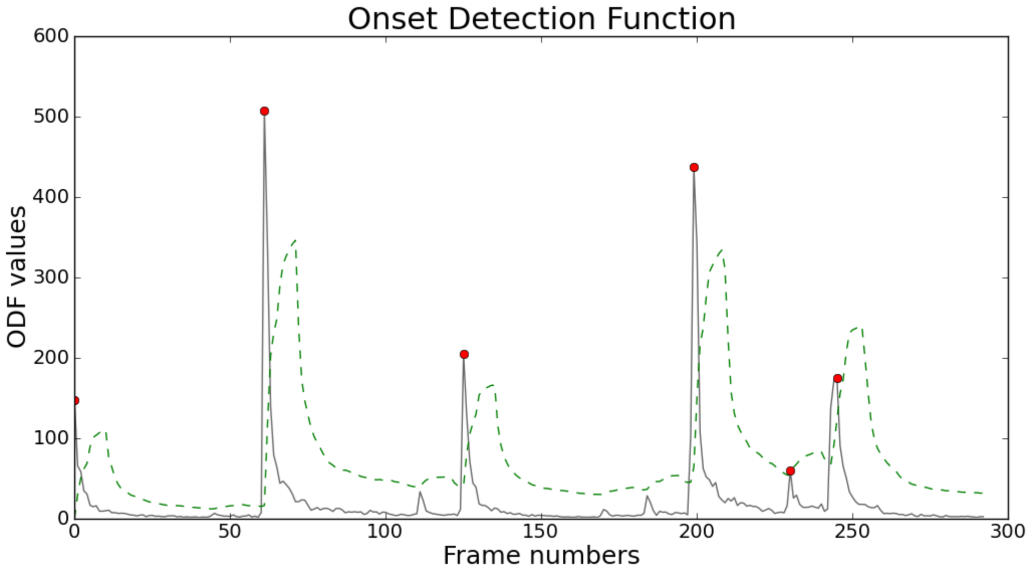

Figure 2: ODF peaks detected (circled) and threshold (dashes) during real-time peak picking

that frame containing a note onset. A full analysis of the detection accuracy and computational efficiency of each algorithm is given in Section 5.

2.3.1 Energy ODF

This approach, described in [5], is the most simple conceptually and is the most computationally efficient. It is based on the premise that musical note onsets often have more energy than the steady-state component of the note, as for many instruments this is when the excitation is applied. Larger changes in the amplitude envelope of the signal should therefore coincide with onset locations. For each frame, the energy is given by

$$E(n) = \sum_{m=0}^{N} x(m)^2 \tag{3}$$

where $E(n)$ is the energy of frame n and $x(m)$ is the value of the m^{th} sample in the frame. The value of the energy ODF (ODF_E) for frame *n* is the absolute value of the difference in energy values between consecutive frames.

$$ODF_E(n) = |E(n) - E(n-1)| \tag{4}$$

2.3.2 Spectral Difference ODF

Many recent techniques for creating onset detection functions have tended towards identifying time-varying changes in a frequency domain representation of an audio signal. These approaches have proven to be successful in a number of areas, such as in detecting onsets in polyphonic signals [15] and in detecting "soft" onsets created by instruments such as the bowed violin which do not have a percussive attack [16]. The spectral difference ODF (ODF_{SD}) is calculated by examining frame-to-frame changes in the Short-Time Fourier Transform [17] of an audio signal and so falls into this category.

The Fourier transform of the n^{th} frame, windowed using a Hanning window *w(m)* of size *N* is given by

$$X(k,n) = \sum_{m=0}^{N-1} x(m)w(m)e^{-\frac{2j\pi mk}{N}} \tag{5}$$

where $X(k,n)$ is the k^{th} frequency bin of the n^{th} frame.

The spectral difference [16] is the absolute value of the change in magnitude between corresponding bins in consecutive frames. As a new musical onset will often result in a sudden change in the frequency content in an audio signal, large changes in the average spectral difference of a frame will often correspond with note onsets. The spectral difference ODF is thus created by summing the spectral difference across all bins in a frame. This is given by Equation 6.

$$ODF_{SD}(n) = \sum_{k=0}^{\frac{N}{2}} ||X(k,n)| - |X(k,n-1)|| \tag{6}$$

2.3.3 Complex Domain ODF

Another way to view the construction of an ODF is in terms of *predictions* and *deviations* from predicted values. For every spectral bin in the Fourier transform of a frame of audio samples, the spectral difference ODF predicts that the next magnitude value will be the same as the current one. In the steady-state of a musical note, changes in the magnitude of a given bin between consecutive frames should be relatively low and so this prediction should be accurate. In transient regions, these variations should be more pronounced and so the average deviation from the predicted value should be higher, resulting in peaks in the ODF.

Instead of making predictions using only the bin magnitudes, the complex domain ODF [18] attempts to improve the prediction for the next value of a given bin by using combined magnitude and phase information. The magnitude prediction is the magnitude value from the corresponding bin in the previous frame. In polar form we can write this predicted value as

$$\hat{R}(k,n) = |X(k,n-1)| \tag{7}$$

The phase prediction is formed by assuming a constant rate of phase change between frames.

$$\hat{\phi}(k,n) = princarg[2\varphi(k,n-1) - \varphi(k,n-2)] \tag{8}$$

where $princarg$ maps the phase to the $[-\pi, \pi]$ range, and $\varphi(k,n)$ is the phase of the k^{th} bin in the n^{th} frame. If $R(k,n)$ and $\phi(k,n)$ are the actual values of the magnitude and phase of bin *k* in frame *n*, the deviation between the prediction and the actual measurement is the Euclidean distance between the two complex phasors, which can be written as

$$\Gamma(k,n) = \sqrt{R(k,n)^2 + \hat{R}(k,n)^2 - 2R(k,n)\hat{R}(k,n)cos(\phi(k,n) - \hat{\phi}(k,n))} \tag{9}$$

The complex domain ODF (ODF_{CD}) is the sum of these deviations across all bins in a frame, as given in Equation 10.

$$ODF_{CD}(n) = \sum_{k=0}^{\frac{N}{2}} \Gamma(k,n) \tag{10}$$

3 Measuring Signal Predictability

The ODFs that are described in Section 2.3, and the majority of those found elsewhere in the literature [6], are trying to distinguish between the steady-state and transient regions of an audio signal by making predictions based on information about the most recent frame of audio and one or two preceding frames. In this section we present methods that use the same basic signal information to the approaches described in Section 2.3, but instead of making predictions based on just one or two frames of this data, we use an arbitrary number of previous values combined with linear prediction (LP) to improve the accuracy of the estimate. The ODF is then the absolute value of the differences between the actual frame measurements and the LP predictions. The ODF values are low when the LP prediction is accurate, but larger in regions of the signal that are more unpredictable, which should correspond with note onset locations.

This is not the first time that linear prediction errors have been used to create an onset detection function. The authors in [19] describe a somewhat similar system in which an audio signal is first filtered into 6 non-overlapping sub-bands.

The first 5 bands are then decimated by a factor of 20:1 before being passed to a LP error filter, while just the amplitude envelope is taken from the 6^{th} band (everything above the note B7 which is 3951 kHz). Their ODF is the sum of the 5 LP error signals and the amplitude envelope from the 6^{th} band.

Our approach differs in a number of ways. In this paper we show that LP can be used to improve the detection accuracy of the three ODFs described in Section 2.3 (detection results are given in Section 5). As this approach involves predicting the time-varying changes in signal features (energy, spectral difference and complex phasor positions) rather than in the signal itself, the same technique could be applied to many existing ODFs from the literature, and so it can be viewed as an additional post-processing step that can potentially improve the detection accuracy of existing ODFs. Our algorithms are suitable for real-time use and the results were compiled from real-time data. In contrast, the results given in [19] are based on off-line processing, and include an initial pre-processing step to normalise the input audio files, so it is not clear how well this method performs in a real-time situation.

The LP process that is used in this paper is described in Section 3.1. In Sections 3.2, 3.3 and 3.4 we show this can be used to create new ODFs based on the energy, spectral difference and complex domain ODFs respectively.

3.1 Linear Prediction

In the linear prediction model, also known as the autoregressive model, the current input sample $x(n)$ is estimated by a weighted combination of the past values of the signal. The predicted value, $\hat{x}(n)$, is computed by FIR filtering according to

$$\hat{x}(n) = \sum_{k=1}^{p} a_k x(n-k) \tag{11}$$

where p is the order of the LP model and a_k are the prediction coefficients.

The challenge is then to calculate the LP coefficients. There are number of methods given in the literature, among the most widespread are the autocorrelation method [20], covariance method [9] and the Burg method [21]. All three methods were evaluated, but the Burg method was selected as it produced the most accurate and consistent results. Like the autocorrelation method, it is minimum phase, and like the covariance method it estimates the coefficients on a finite support [21]. It can also be efficiently implemented in real-time [20].

3.1.1 The Burg Algorithm

The LP error is the difference between the predicted value and the actual value.

$$e(n) = x(n) - \hat{x}(n) \tag{12}$$

The Burg algorithm minimises average of the forward prediction error $f_m(n)$ and the backward prediction error $b_m(n)$. The initial (order 0) forward and backward errors are given by

$$f_0(n) = x(n) \tag{13}$$

$$b_0(n) = x(n) \tag{14}$$

over the interval $n = 0, \ldots N-1$, where N is the block length. For the remaining $m = 1, \ldots, p$ the m^{th} coefficient is calculated from

$$k_m = \frac{-2\sum_{n=m}^{N-1}[f_{m-1}(n)b_{m-1}(n-1)]}{\sum_{n=m}^{N-1}[f_{m-1}^2(n) + b_{m-1}^2(n-1)]} \tag{15}$$

then the forward and backward prediction errors are recursively calculated from

$$f_m(n) = f_{m-1}(n) - k_m b_{m-1}(n-1) \tag{16}$$

for $n = m+1, \ldots, N-1$, and

$$b_m(n) = b_{m-1}(n-1) - k_m f_{m-1}(n) \tag{17}$$

for $n = m, \ldots, N-1$ respectively. Pseudocode for this process is given in Algorithm 2, taken from [21].

```
$f \longleftarrow x$
$b \longleftarrow x$
$a \longleftarrow x$

for $m \leftarrow 0$ to $p-1$ do
    $fp \longleftarrow f$ without its first element
    $bp \longleftarrow b$ without its last element
    $k \longleftarrow -2bp \cdot fp/(fp \cdot fp + fp \cdot fp)$
    $f \longleftarrow fp + k \cdot bp$
    $b \longleftarrow bp + k \cdot fp$
    $a \longleftarrow (a[0], a[1], \ldots, a[m], 0) + k(0, a[m], a[m-1], \ldots, a[0])$
end
```

Algorithm 2: The Burg method.

3.2 Energy With Linear Prediction

The energy ODF (given in Section 2.3.1) is derived from the absolute value of the energy difference between two frames. This can be viewed as using the energy value of the first frame as a prediction of the energy of the second, with the difference being the prediction error. Here we try to improve this estimate by using linear prediction. Energy values from the past p frames are taken, resulting in the sequence

$$E(n-1), E(n-2), \ldots, E(n-p)$$

Using (13) - (17), p coefficients are calculated based on this sequence, then a one sample prediction is made using (11). So for each frame the energy with linear prediction ODF (ODF_{ELP}) is given by

$$ODF_{ELP}(n) = |E(n) - P_E(n)| \qquad (18)$$

where $P_E(n)$ is the predicted energy value for frame n.

3.3 Spectral Difference With Linear Prediction

Similar techniques can be applied to the spectral difference and complex domain ODFs. The spectral difference ODF is formed from the absolute value of the magnitude differences between corresponding bins in adjacent frames. Similarly to the process described in Section 3.2, this can be viewed as a prediction that the magnitude in a given bin will remain constant between adjacent frames, with the magnitude difference being the prediction error. In the spectral difference with linear prediction ODF (ODF_{SDLP}), the predicted magnitude value for each of the k bins in frame n is calculated by taking the magnitude values from the corresponding bins in the previous p frames, using them to find p LP coefficients then filtering the result with (11). So, for each k in n, the magnitude prediction coefficients are formed by using (13) - (17) on the sequence

$$|X(k, n-1)|, |X(k, n-2)|, \ldots, |X(k, n-p)|$$

If $P_{SD}(k, n)$ is the predicted spectral difference for bin k in n, then

$$ODF_{SDLP}(n) = \sum_{k=0}^{\frac{N}{2}} ||X(k,n)| - P_{SD}(k,n)| \qquad (19)$$

As is shown in Section 5.3, this is a significant amount of extra computation per frame compared with the ODF_{SD} given by Equation (6). However, it is still capable of real-time performance, depending on the chosen LP model order. We found that an order of 5 was enough to significantly improve the detection accuracy while still comfortably meeting the real-time processing requirements. Detailed results are given in Section 5.

3.4 Complex Domain With Linear Prediction

The complex domain method described in Section 2.3.3 is based on measuring the Euclidean distance between the predicted and actual complex phasors for a given bin. There are a number of different ways that linear prediction could be applied in an attempt to improve this estimate. The bin magnitudes and phases could be predicted separately, based on their values over the previous p frames, and then combined to form an estimated phasor value for the current frame. Another possibility would be to only apply linear prediction to one of either the magnitude or phase parameters.

However, we found that the biggest improvement came from using linear prediction to estimate the value of the Euclidean distance that separates the complex phasors for a given bin between consecutive frames. So for each bin k in frame n, the complex distances between the k^{th} bin in each of the last p frames are used to calculate the LP coefficients. If $R(k, n)$ is the magnitude of the k^{th} bin in frame n, and $\phi(k, n)$ is the phase of the bin, then the distance between the k^{th} bins in frames n and $n - 1$ is

$$\Gamma(k,n) = \sqrt{R(k,n)^2 + R(k,n-1)^2 - 2R(k,n)R(k,n-1)cos(\phi(k,n) - \phi(k,n-1))}$$

LP coefficients are formed from the values

$$\Gamma(k,n-1), \Gamma(k,n-2), \ldots, \Gamma(k,n-p)$$

using (13) - (17) and predictions $P_{CD}(k, n)$ are calculated using (11). The complex domain with linear prediction ODF (ODF_{CDLP}) is then given by

$$ODF_{CDLP}(n) = \sum_{k=0}^{\frac{N}{2}} |\Gamma(k,n) - P_{CD}(k,n)| \qquad (20)$$

4 Real-Time Onset Detection Using Sinusoidal Modelling

In Section 3, we describe a way to improve the detection accuracy of several ODFs from the literature by using linear prediction to enhance their estimates of the frame-by-frame evolution of an audio signal. This improvement in detection accuracy comes at the expense of much great computational cost however (see Section 5 for detection accuracy and performance results).

In this section we present a novel ODF that has significantly better real-time performance than the LP-based spectral methods. It uses sinusoidal modelling, and so it is particularly useful in areas that include some sort of harmonic analysis. We begin with an overview of sinusoidal modelling in Section 4.1, followed by a review of previous work that uses sinusoidal modelling for onset detection in Section 4.2 and then concludes with a description of the new ODF in Section 4.3.

4.1 Sinusoidal Modelling

Sinusoidal modelling [10] is based on Fourier's theorem, which states that any periodic waveform can be modelled as the sum of sinusoids at various amplitudes and harmonic frequencies. For stationary pseudo-periodic sounds, these amplitudes and frequencies evolve slowly with time. They can be used as parameters to control pseudo-sinusoidal oscillators, commonly referred to as partials. The audio signals can be calculated from the sum of the partials using

$$s(t) = \sum_{p=1}^{N_p} A_p(t) cos(\theta_p(t)) \tag{21}$$

$$\theta_p(t) = \theta_p(0) + 2\pi \int_0^t f_p(u)du \tag{22}$$

where N_p is the number of partials and A_p, f_p and θ_p are the amplitude, frequency and phase of the p^{th} partial respectively. Typically, the parameters are measured for every

$$t = nh/F_s$$

where n is the sample number, h is the buffer size and F_s is the sampling rate. To calculate the audio signal, the parameters must then be interpolated between measurements. Calculating these parameters for each frame is referred to in this paper as *peak detection*, while the process of connecting these peaks between frames is called *partial tracking*.

4.2 Sinusoidal Modelling And Onset Detection

The sinusoidal modelling process can be extended, creating models of sound based on the separation of the audio signal into a combination of sinusoids and noise [22], and further into combinations of sinusoids, noise and transients [23]. Although primarily intended to model transient components from musical signals, the system described in [23] could also be used to detect note onsets. The authors show that transient signals in the time domain can be mapped onto sinusoidal signals in a frequency domain, in this case by using the Discrete Cosine Transform (DCT) [24]. Roughly speaking, the DCT of a transient time-domain signal produces a signal with a frequency that depends only on the time shift of the transient. This information could then be used to identify when the onset occurred. However, it is not suitable for real-time applications as it requires a DCT frame size that makes the transients appear as a small entity, with a frame duration of about one second recommended. This is far too much latency to meet the real-time requirements that were specified in Section 2.1.

Another system that combines sinusoidal modelling and onset detection is presented in [25]. It creates an ODF that is a combination of two energy measurements. The first is simply the energy in the audio signal over a 512 sample frame. If the energy of the current frame is larger than that of a given number of previous frames, then the current frame is a candidate for being an onset location. A multi-resolution sinusoidal model is then applied to the signal in order to isolate the harmonic component of the sound. This differs from the sinusoidal modelling implementation described above in that the audio signal is first split into 5 octave spaced frequency bands. Currently only the lower 3 are used, the upper 2 (frequencies above about 5 kHz) are discarded. Each band is then analysed using different window lengths, allowing for more frequency resolution in the lower band at the expense of worse time resolution. Sinusoidal amplitude, frequency and phase parameters are estimated separately for each band, and linked together to form partials. An additional post-processing step is applied to the partials, removing any that have an average amplitude that is less than an adaptive psychoacoustic masking threshold, and removing any that are less than 46 ms in duration.

As it stands, it is unclear whether or not the system described in [25] is suitable for use as a real-time onset detector. The stipulation that all sinusoidal partials must be at least 46 ms in duration implies that there must be a minimum latency of 46 ms in the sinusoidal modelling process, putting it very close to our 50 ms limit. If used purely as an ODF in the onset detection system described in Section 2.3, the additional 11.6 ms of latency incurred by the peak detection stage would put the total latency outside this 50 ms window. However, their method uses a rising edge detector instead of looking for peaks so it may still meet our real-time requirements, although as it was designed as part of a larger system that is primarily intended to encode audio for compression, no onset detection accuracy or performance results are given by the authors.

In contrast, the ODF that is presented in Section 4.3 was designed specifically as a real-time onset detector and so has a latency of just two buffer sizes (23.2 ms in our implementation). As we show in Section 5, it compares favourably to leading approaches from the literature in terms of computational efficiency and it is also more accurate than the reviewed methods.

4.3 Peak Amplitude Difference ODF

This ODF is based on the same underlying premise as sinusoidal models, namely that during the steady-state of a musical note, the harmonic signal component can be well modelled as a sum of sinusoids. These sinusoids should evolve slowly in time, and should therefore be well represented by the partials detected by the sinusoidal modelling process. It follows then that during the steady-state, the absolute values of the frame-to-frame differences in the sinusoidal peak amplitudes and frequencies should be quite low. In comparison, transient regions at note onset locations should show considerably more frame-by-frame variation in both peak frequency and amplitude values. This is due to two main factors:

1. Many musical notes have an increase in signal energy during their attack regions that corresponds to a physical excitation being applied, which increases the amplitude of the detected sinusoidal components.

2. As transients are by definition less predictable and less harmonic, the basic premise of the sinusoidal model breaks down in these regions. This can result in peaks existing in these regions that are really noise and not part of any underlying harmonic component. Often they will remain unmatched and so do not form long-duration partials. Alternatively, if they are incorrectly matched it can result in relatively large amplitude and/or frequency deviations in the resulting partial. In either case, the difference between the parameters of the noisy peak and the parameters of any peaks before and after it will often differ significantly.

Both of these factors should lead to larger frame-to-frame sinusoidal peak amplitude differences in transient regions than in steady-state regions. We can therefore create an ODF by analysing the differences in peak amplitude values over consecutive frames.

The sinusoidal modelling algorithm that we used is very close to the one described in [26], with a couple of changes to the peak detection process. Firstly, the number of peaks per frame can be limited to M_p, reducing the computation required for the partial tracking stage [27, 28]. If the number of detected peaks $N_p > M_p$, the M_p largest amplitude peaks will be selected. Also, in order to allow for consistent evaluation with the other frequency domain ODFs described in this paper, the frame size is kept constant during analysis (2048 samples). The partial tracking process is identical to the one given in [26]. As this partial tracking algorithm has a delay of one buffer size, this ODF has an additional latency of 512 samples, bringing the total detection latency (including the peak picking phase) to 1536 samples or 34.8 ms when sampled at 44.1 kHz.

For a given frame n, let $P_k(n)$ be the peak amplitude of the k^{th} partial. The peak amplitude difference ODF (ODF_{PAD}) is given by

$$ODF_{PAD}(n) = \sum_{k=0}^{M_p} |P_k(n) - P_k(n-1)| \tag{23}$$

In the steady-state, frame-to-frame peak amplitude differences for matched peaks should be relatively low, and as the matching process here is significantly easier than in transient regions, less matching errors are expected. At note onsets, matched peaks should have larger amplitude deviations due to more energy in the signal, and there should also be more unmatched or incorrectly matched noisy peaks, increasing the ODF value. As specified in [26], unmatched peaks for a frame are taken to be the start of a partial, and so the amplitude difference is equal to the amplitude of the peak, $P_k(n)$.

5 Evaluation Of Real-Time Onset Detection Functions

This section provides evaluations of all of the ODFs described in this paper. Section 5.1 describes a new library of onset detection software, which includes a database of hand-annotated musical note onsets, that was created as part of this work. This database was used to assess the performance of the different algorithms. Section 5.2 evaluates the detection accuracy of each ODF, with their computational complexities described in Section 5.3. Section 5.4 concludes with a discussion of the evaluation results.

5.1 Musical Onset Database And Library (Modal)

In order to evaluate the different ODFs described in Sections 2.3, 3 and 4.3, it was necessary to access a set of audio files with reference onset locations. To the best of our knowledge, the Sound Onset Labellizer [11] was the only freely available reference collection, but unfortunately it was not available at the time of publication. Their reference set also made use of files from the RWC database [29], which although publicly available is not free and does not allow free redistribution.

These issues lead to the creation of Modal, which contains a free collection of samples, all with creative commons licensing allowing for free reuse and redistribution, and including hand-annotated onsets for each file. Modal is also a new open source (GPL), cross-platform library for musical onset detection written in C++ and Python, and contains implementations of all of the ODFs discussed in this paper in both programming languages. Additionally, from Python there is onset detection and plotting functionality, as well as code for generating our analysis data and results. It also includes an application that allows for the labelling of onset locations in audio files, which can then be added to the database. Modal is available now at http://github.com/johnglover/modal.

5.2 Detection Results

The detection accuracy of the ODFs was measured by comparing the onsets detected using each method with the reference samples in the Modal database. To be marked as "correctly detected", the onset must be located within 50 ms of a reference onset. Merged or double onsets were not penalised. The database currently contains 501 onsets from annotated sounds that are mainly monophonic, so this must be taken under consideration when viewing the results. The annotations were also all made by one person, and while it has been shown in [11] that this is not ideal, the chosen detection window of 50 ms should compensate for some of the inevitable inconsistencies.

The results are summarised by three measurements that are common in the field of Information Retrieval [15]: the precision (P), the recall (R), and the F-measure (F), defined here as follows:

$$P = \frac{C}{C + f_p} \tag{24}$$

$$R = \frac{C}{C + f_n} \tag{25}$$

$$F = \frac{2PR}{P + R} \tag{26}$$

where C is the number of correctly detected onsets, f_p is the number of false positives (detected onsets with no matching reference onset) and f_n is the number of false negatives (reference onsets with no matching detected onset).

Every reference sample in the database was streamed one buffer at a time to each ODF, with ODF values for each buffer passed immediately to a real-time peak picking system, as described in Algorithm 1. Dynamic thresholding was applied according to (1), with $\lambda = 1.0$, $\alpha = 2.0$, and w in (2) set to 0.05. A median window of 7 previous values was used. These parameters were kept constant for each ODF. Our novel methods that use linear prediction (described in Sections 3.2, 3.3 and 3.4) each used a model order of 5, while our peak amplitude difference method described in Section 4.3 was limited to a maximum of 20 peaks per frame.

The precision, recall and F-measure results for each ODF are given by Figure 3, Figure 4 and Figure 5 respectively. In each figure the blue bars give the results for the ODFs from the literature (described in Section 2.3), the brown bars give the results for our linear prediction methods, and the green bar gives the results for our peak amplitude difference method.

Figure 3 shows that the precision values for all of our methods are higher than the methods from the literature. The addition of linear prediction noticeably improves each ODF that it is applied to. The precision values for the peak amplitude difference method is better than the literature methods and the energy with linear prediction method, but worse than the two spectral-based linear prediction methods.

The recall results for each ODF are given in Figure 4. Here we can see that linear prediction has improved the energy method, but made the spectral difference and complex domain methods slightly worse. The peak amplitude difference method has a greater recall than all of the literature methods and is only second to the energy with linear prediction ODF.

Figure 5 gives the F-measure for each ODF. All of our proposed methods are shown to perform better than the methods from the literature. The spectral difference with linear prediction ODF has the best detection accuracy, while the energy with linear prediction, complex domain with linear prediction and peak amplitude difference methods are all closely matched.

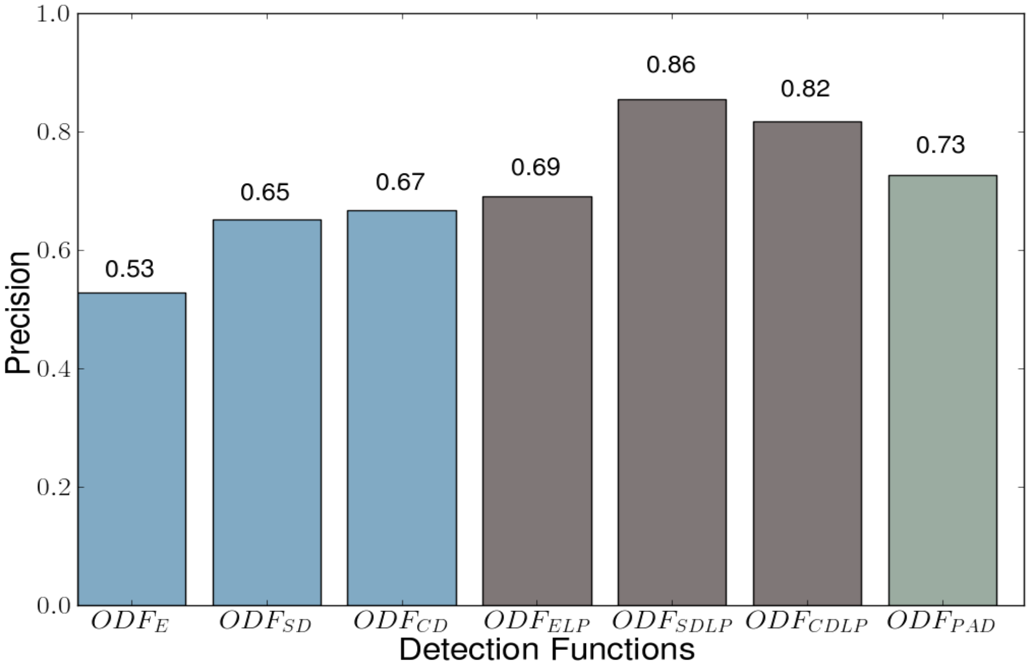

Figure 3: Precision values for each ODF

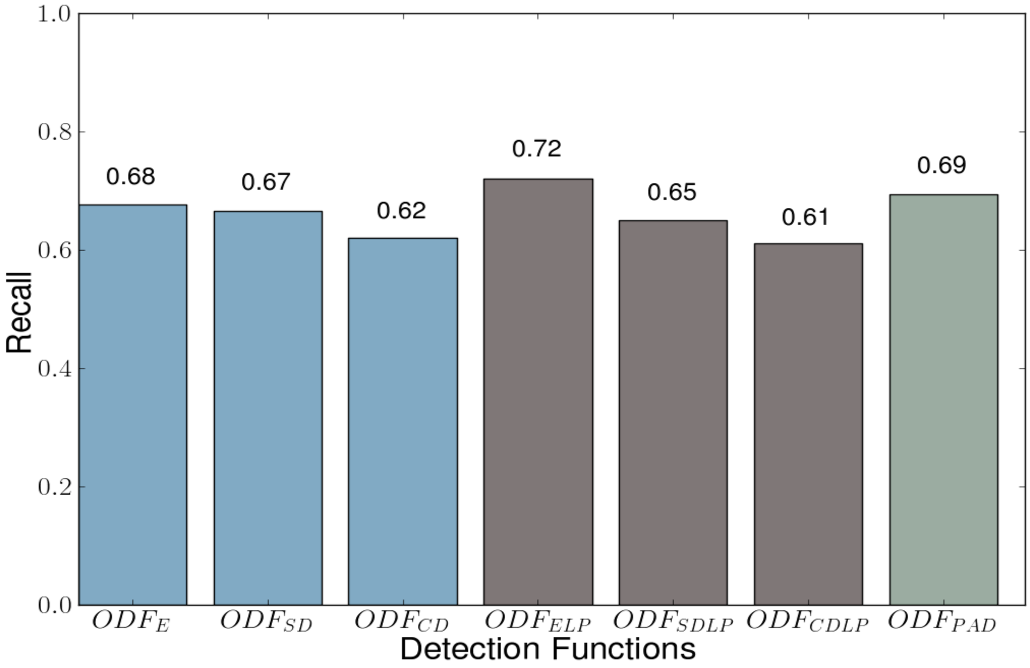

Figure 4: Recall values for each ODF

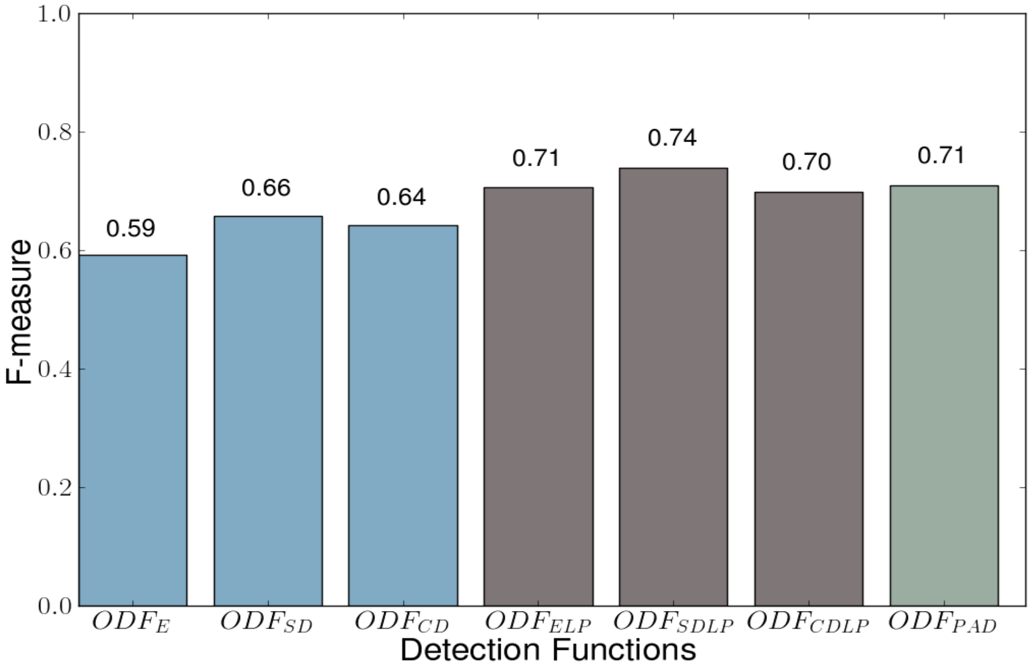

Figure 5: F-measure values for each ODF

5.3 Performance Results

In Table 1 we give the worst-case number of floating-point operations per second (FLOPS) required by each ODF in order to process real-time audio streams, based on our implementations in the Modal library. This analysis does not include data from the setup/initialisation periods of any of the algorithms, or data from the peak detection stage of the onset detection system. As specified in Section 2.1, the audio frame size is 2048 samples, the buffer size is 512 samples and the sampling rate is 44.1 kHz. The linear prediction methods all use a model order of 5. The number of peaks in the ODF_{PAD} is limited to 20.

These totals were calculated by counting the number of floating-point operations required by each ODF in order to process 1 frame of audio, where we define a floating-point operation to be an addition, subtraction, multiplication, division or assignment involving a float-point number. As we have a buffer size of 512 samples measured at 44.1 kHz, we have 86.133 frames of audio per second, so the number of operations required by each ODF per frame of audio was multiplied by 86.133 to get the FLOPS total for that ODF.

To simplify the calculations, the following assumptions were made when calculating the totals:

- As we are using the real Fast Fourier Transform (FFT) computed using the FFTW3 library [30], the processing time required for a FFT is $2.5Nlog_2(N)$ where N is the FFT size, as given in [31].
- The complexity of basic arithmetic functions in the C++ standard library such as $sqrt$, cos, sin, and log is $O(M)$, where M is the number of digits of precision at which the function is to be evaluated.
- All integer operations can be ignored.
- All function call overhead can be ignored.

As Table 1 shows, the energy-based methods (ODF_E and ODF_{ELP}) require far less computation than any of the others. The spectral difference ODF is the third fastest, needing about half the number of operations that are required by the complex domain method. The worst case requirements for the peak amplitude difference method are still relatively close to the spectral difference ODF and noticeably quicker than the complex domain ODF. As expected, the addition of linear prediction to the spectral difference and complex domain methods makes them significantly more expensive computationally than any other technique.

To give a more intuitive view of the algorithmic complexity, in Table 2 we also give the estimated real-time CPU usage for each ODF given as a percentage of the maximum number of FLOPS that can be achieved by two different processors:

	FLOPS
ODF_E	529,718
ODF_{SD}	7,587,542
ODF_{CD}	14,473,789
ODF_{ELP}	734,370
ODF_{SDLP}	217,179,364
ODF_{CDLP}	217,709,168
ODF_{PAD}	9,555,940

Table 1: Number of floating-point operations per second (FLOPS) required by each ODF to process real-time audio streams, with a buffer size of 512 samples, a frame size of 2048 samples, a linear prediction model order of 5, and a maximum of 20 peaks per frame for ODF_{PAD}.

an Intel Core 2 Duo and an Analog Devices ADSP-TS201S (TigerSHARC). The Core 2 Duo has a clock speed of 2.8 GHz, a 6 MB L2 cache and a bus speed of 1.07 GHz, providing a theoretical best case performance of 22.4 GFLOPS [32]. The ADSP-TS201S has a clock speed of 600 MHz and a best case performance of 3.6 GFLOPS [33], and scores relatively well on the BDTI DSP Kernel Benchmarks [34]. Any value less than 100% here shows that the ODF can be calculated in real-time on this processor.

	Core 2 Duo (%)	ADSP-TS201S (%)
ODF_E	0.002	0.015
ODF_{SD}	0.034	0.211
ODF_{CD}	0.065	0.402
ODF_{ELP}	0.003	0.020
ODF_{SDLP}	0.970	6.033
ODF_{CDLP}	0.972	6.047
ODF_{PAD}	0.043	0.265

Table 2: Estimated real-time CPU usage for each ODF, shown as a percentage of the maximum number of FLOPS that can be achieved on two processors: an Intel Core 2 Duo and an Analog Devices ADSP-TS201S (TigerSHARC).

5.4 Discussion

The F-measure results (shown in Figure 5) for the methods described in Section 2.3 are lower than those given elsewhere in the literature, but this was expected as real-time performance is significantly more challenging at the peak picking and thresholding stages. The nature of the sample set must also be taken into account, as evidently the heavy bias towards monophonic sounds is reflected by the surprisingly strong performance of the energy-based methods. As noted in [8], the various parameter settings can have a large impact on overall performance. We tried to select a parameter set that gave a fair reflection on each algorithm, but it must be noted that every method can probably be improved by some parameter adjustments, especially if prior knowledge of the sound source is available.

In terms of performance, the linear prediction methods are all significantly slower than their counterparts. However, even the most computationally expensive algorithm can run with an estimated real-time CPU usage of just over 6% on the ADSP-TS201S (TigerSHARC) processor, so they are still more than capable of real-time performance. The energy with linear prediction ODF in particular is extremely cheap computationally, and yet has relatively good detection accuracy for this sample set.

The peak amplitude difference method is also notable as it is computationally cheaper than the complex domain ODF and compares favourably with the spectral difference ODF, while giving better accuracy than both for our sample set. For applications such as real-time sound synthesis that may already include a sinusoidal modelling process, this becomes an extremely quick method of onset detection. One significant difference between the peak amplitude difference ODF and the others is that the computation time is not fixed, but depends on the sound source. Harmonic material will have well defined partials, potentially requiring more processing time for the partial tracking process than noisy sound sources, for this sinusoidal modelling implementation at least.

6 Conclusions

In this paper we have described two new approaches to real-time musical onset detection, one using linear prediction and the other using sinusoidal modelling. We compared these approaches to some of the leading real-time musical onset detection algorithms from the literature, and found that they can offer either improved accuracy, computational efficiency, or both. It is recognised that onset detection results are very context sensitive, so without a more extensive sample set it is hard to make completely conclusive comparisons to other methods. However, our software and our sample database are both released under open source licenses and are freely redistributable, so hopefully other researchers in the field will contribute.

Choosing a real-time onset detection function remains a complex issue and depends on the nature of the input sound, the available processing power and the penalties that will be experienced for producing false negatives and false positives. However, some recommendations can be made based on the results in this paper. For our sample set, the spectral difference with linear prediction method produced the most accurate results, so if computational complexity is not an issue then this would be a good choice. On the other hand, if low complexity is an important requirement then the energy with linear prediction ODF is an attractive option. It produced accurate results at a fraction of the computational cost of some of the established methods.

The peak amplitude difference ODF is also noteworthy and should prove to be useful in areas such as real-time sound synthesis by analysis. Spectral processing techniques such as the Phase Vocoder or sinusoidal models work well during the steady-state regions of musical notes, but have problems in transient areas which follow note onsets [5, 23]. One solution to this problem is to identify these regions and process them differently, which requires accurate onset detection in order to avoid synthesis artifacts. It is in this context that the peak amplitude difference ODF is particularly useful. It was shown to provide more accurate results than the well established complex domain method with noticeably lower computation requirements, and as it integrates seamlessly with the sinusoidal modelling process, it can be added to existing sinusoidal modelling systems at very little cost.

7 Acknowledgements

The authors would like to acknowledge the generous support of An Foras Feasa, who funded this research.

References

[1] N. Orio, S. Lemouton, and D. Schwarz, “Score following: State of the art and new developments,” in *Proceedings of the 2003 Conference on New Interfaces for Musical Expression (NIME-03)*, (Montreal, Canada), 2003.

[2] A. Stark, D. Matthew, and M. Plumbley, “Real-time beat-synchronous analysis of musical audio,” in *Proceedings of the 12th International Conference on Digital Audio Effects (DAFx-09)*, (Como, Italy), September 2009.

[3] N. Schnell, D. Schwarz, and R. Muller, “X-micks - interactive content based real-time audio processing,” in *Proceedings of the 9th International Conference on Digital Audio Effects (DAFx-06)*, (Montreal, Canada), 2006.

[4] M. Dolson, “The phase vocoder: A tutorial,” *Computer Music Journal*, vol. 10, pp. 14–27, Winter 1986.

[5] C. Duxbury, M. Davies, and M. Sandler, “Improved time-scaling of musical audio using phase locking at transients,” in *112th Audio Engineering Society Convention*, (Munich, Germany), May 2002.

[6] J. P. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies, and M. Sandler, “A Tutorial on Onset Detection in Music Signals,” *IEEE Transactions on Speech and Audio Processing*, vol. 13, pp. 1035–1047, Sept. 2005.

[7] D. Stowell and M. Plumbley, “Adaptive whitening for improved real-time audio onset detection,” in *Proceedings of the International Computer Music Conference (ICMC’07)*, (Copenhagen, Denmark), pp. 312–319, August 2007.

[8] S. Dixon, “Onset detection revisited,” in *Proceedings of the 9th International Conference on Digital Audio Effects (DAFx-06)*, (Montreal, Canada), September 2006.

[9] J. Makhoul, “Linear prediction: A tutorial review,” *Proceedings of the IEEE*, vol. 63, no. 4, pp. 561–580, 1975.

[10] X. Amatriain, J. Bonada, A. Loscos, and X. Serra, *DAFx - Digital Audio Effects*, ch. Spectral Processing, pp. 373–438. John Wiley and Sons, 2002.

[11] P. Leveau, L. Daudet, and G. Richard, “Methodology and tools for the evaluation of automatic onset detection algorithms in music,” in *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR)*, (Barcelona, Spain), October 2004.

[12] J. Vos and R. Rasch, “The perceptual onset of musical tones,” *Perception and Psychophysics*, vol. 29, no. 4, pp. 323–335, 1981.

[13] I. Kauppinen, “Methods for detecting impulsive noise in speech and audio signals,” in *Proceedings of the 14th International Conference on Digital Signal Processing (DSP 2002)*, vol. 2, pp. 967–970, 2002.

[14] P. Brossier, J. P. Bello, and M. Plumbley, “Real-time temporal segmentation of note objects in music signals,” in *Proceedings of the International Computer Music Conference (ICMC’04)*, pp. 458–461, 2004.

[15] “Mirex 2009 audio onset detection results.” http://www.music-ir.org/mirex/wiki/2009:Audio_Onset_Detection_Results (last accessed 05-10-2010).

[16] C. Duxbury, M. Sandler, and M. Davies, “A hybrid approach to musical note onset detection,” in *Proceedings of the 5th International Conference*

on Digital Audio Effects (DAFx-02), (Hamburg, Germany), pp. 33–38, September 2002.

[17] J. Allen and L. Rabiner, "A unified approach to short-time Fourier analysis and synthesis," *Proceedings of the IEEE*, vol. 65, November 1977.

[18] 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, pp. 553–556, June 2004.

[19] W.-C. Lee and C.-C. J. Kuo, "Musical onset detection based on adaptive linear prediction," in *Proceedings of the 2006 IEEE Conference on Multimedia and Expo, ICME 2006*, (Ontario, Canada), pp. 957–960, July 2006.

[20] F. Keiler, D. Arfib, and U. Zolzer, "Efficient linear prediction for digital audio effects," in *Proceedings of the COST G-6 Conference on Digital Audio Effects (DAFX-00)*, (Verona, Italy), December 200.

[21] M. Lagrange, S. Marchand, M. Raspaud, and J.-B. Rault, "Enhanced partial tracking using linear prediction," in *Proceedings of the 6th International Conference on Digital Audio Effects (DAFx-03)*, (London, UK), September 2003.

[22] X. Serra and J. Smith, "Spectral modeling synthesis: A sound analysis/synthesis system based on a deterministic plus stochastic decompostion," *Computer Music Journal*, vol. 14, pp. 12–24, Winter 1990.

[23] T. S. Verma and T. H. Y. Meng, "Extending spectral modeling synthesis with transient modeling synthesis," *Computer Music Journal*, vol. 24, pp. 47–59, Summer 2000.

[24] N. Ahmed, T. Natarajan, and K. Rao, "Discrete cosine transfom," *IEEE Transactions on Computers*, vol. C-23, pp. 90–93, January 1974.

[25] S. Levine, *Audio Representations for Data Compression and Compressed Domain Processing*. PhD thesis, Stanford University, 1998.

[26] R. McAulay and T. Quatieri, "Speech analysis/synthesis based on a sinusoidal representation," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. ASSP-34, August 1986.

[27] V. Lazzarini, J. Timoney, and T. Lysaght, "Alternative analysis-synthesis approaches for timescale, frequency and other transformations of musical signals," in *Proceedings of the 8th International Conference on Digital Audio Effects (DAFx-05)*, (Madrid, Spain), pp. 18–23, 2005.

[28] V. Lazzarini, J. Timoney, and T. Lysaght, "Time-stretching using the instantaneous frequency distribution and partial tracking," in *Proceedings of the International Computer Music Conference (ICMC'05)*, (Barcelona, Spain), 2005.

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

[30] M. Frigo and S. G. Johnson, "Fftw3 library." http://www.fftw.org (last accessed 29-01-2011).

[31] M. Frigo and S. G. Johnson, "The design and implementation of fftw3," *Proceedings of the IEEE*, vol. 93, no. 2, pp. 216–231, 2005.

[32] Intel Corporation, "Intel microprocessor export compliance metrics." http://www.intel.com/support/processors/sb/cs-023143.htm (last accessed 13-04-2011).

[33] Analog Devices, "ADSP-TS201S data sheet." http://www.analog.com/static/imported-files/data_sheets/ADSP_TS201S.pdf (last accessed 13-04-2011).

[34] Berkeley Design Technology, Inc., "BDTI DSP kernel benchmarks (BDTIMark200) certified results." http://www.bdti.com/Resources/BenchmarkResults/BDTIMark2000 (last accessed 13-04-2011).