Mach Learn (2006) 65:457–471
DOI 10.1007/s10994-006-8458-5

Bootstrap learning for accurate onset detection

Ning Hu* · **Roger B. Dannenberg**

Received: 27 September 2005 / Revised: 17 February 2006 / Accepted: 20 March 2006 / Published online: 8 May 2006

Abstract Supervised learning models have been applied to create good onset detection systems for musical audio signals. However, this always requires a large set of labeled training examples, and hand-labeling is quite tedious and time consuming. In this paper, we present a bootstrap learning approach to train an accurate note onset detection model. Audio alignment techniques are first used to find the correspondence between a symbolic music representation (such as MIDI data) and an acoustic recording. This alignment provides an initial estimate of note boundaries which can be used to train an onset detector. Once trained, the detector can be used to refine the initial set of note boundaries and training can be repeated. This iterative training process eliminates the need for hand-labeled audio. Tests show that this training method can improve an onset detector initially trained on synthetic data.

Keywords Bootstrap learning · Onset detection · Audio-to-score alignment

1. Introduction

A note onset is simply the beginning of a musical note. In real music performances, a note onset can be very smoothly connected to the previous note, so onset detection is challenging.

* Major part of work was done while the first author was at Carnegie Mellon University.

Editor: Gerhard Widmer

N. Hu (✉)
Google Inc. New York Office
1440 Broadway, 21st Floor, New York,
NY 10018, United States
e-mail: ninghu@google.com

R. B. Dannenberg
Computer Science Department,
Carnegie Mellon University, 5000 Forbes Ave,
Pittsburgh, PA 15213, United States
e-mail: rbd@cs.cmu.edu

Onset detection in sound signals is an important topic in many audio processing areas such as speech recognition and audio coding. In the domain of computer music and Music Information Retrieval (MIR), it is often a crucial pre-processing step for further music processing, for example, segmenting sung melodies into note sequences for a Query-by-Humming system, or providing candidate beat locations for rhythmic analysis and tempo estimation. Because of its importance, audio onset detection has been one of the major tasks in the Music Information Retrieval Evaluation eXchange (MIREX) contests (Downie et al., 2005).

A common practice is to apply various machine learning techniques to the onset detection problem, and good results have been obtained. Some of the representative machine learning models used in this area are the Hidden Markov Model (HMM) (Raphael, 1999), Neural Network (Marolt, Kavcic & Privosnik, 2002), Support Vector Machine (SVM) (Lu, Lu & Zhang, 2001), and the Hierarchical Model (Kapanci & Pfeffer, 2004). However, as in many other machine learning applications, audio segmentation using machine learning schemes inevitably faces a problem: getting training data is difficult and tedious. Manually marking each note onset in a five-minute piece of music can take several hours of work. Since the quantity and quality of the training data directly affect the performance of the machine learning model, many designers have no choice but to label some training data by hand.

Meanwhile, audio-to-score alignment has become a popular MIR topic. Linking signal and symbolic representations of music can enable many interesting applications, such as polyphonic music retrieval (Hu, Dannenberg & Tzanetakis, 2003; Muller & Kurth, 2005), real-time score following (Raphael, 2004), and intelligent editors (Dannenberg & Hu, 2003).

In a sense, audio-to-score alignment and music note onset detection are closely related. Both of the operations are performed on acoustic features extracted from the audio, though alignment focuses on global correspondence while onset detection focuses on local changes. Given a *precise* alignment between the symbolic and corresponding acoustic data, desired note onsets can be located in the audio. Even if alignment is not that precise, it still provides valuable information for audio onset detection. Conversely, given *precise* note onset detection, alignment becomes almost trivial. This relationship between alignment and segmentation is exploited in our bootstrap learning method to improve onset detection.

We assume we are given a musical score in some machine-readable form that indicates nominal pitches and durations (e.g., a MIDI file). We are also given an audio recording (possibly played expressively) of that score, and we want to determine the onset times of each note in the audio recording. As a second problem, we want to find onset times in additional audio recordings for which we have no score information.

We propose a bootstrap learning method that uses automatic alignment information to help train the onset detector. The training process consists of two parts. One is an alignment process that finds the time correspondence between the symbolic and acoustic representations of a music piece. The other part is an onset detection process that marks each note onset in the acoustic recording. Alignment is accomplished by matching sequences of chromagram features using Dynamic Time Warping (DTW). The onset detection model is a feed-forward neural network, with several features extracted from audio as the inputs, and a real value between 0 and 1 as the output. The alignment results help train the onset detector iteratively. Our implementation and evaluation show that this training scheme is feasible, and that it can greatly improve the performance of audio onset detection without manually labeling any training data.

The bootstrap learning approach does not replace the commonly used machine learning models previously mentioned. On the contrary, it is meant to be a complementary part that should be used in conjunction with those models. Its essential purpose is to alleviate the problem of not having enough labeled data.

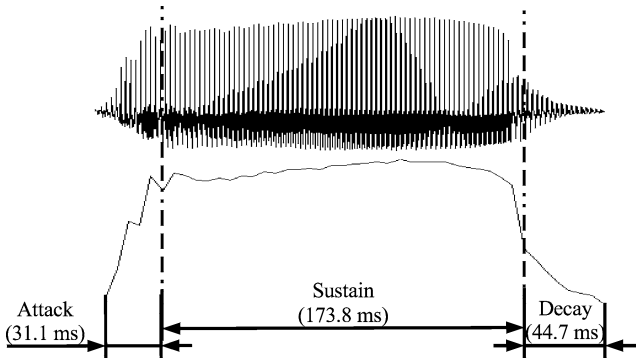

Fig. 1 A typical trumpet slurred note (a mezzo forte C4 from an ascending scale of slurred quarter notes), displayed in waveform along with the amplitude envelope. Attack, sustain and decay parts are indicated in the figure

Onset detection can be viewed as a special case of a more general problem, which is to split audio into segments that have homogeneous properties. A typical example is dividing acoustic recordings into segments with and without singing. Though this paper mainly aims at detecting note onsets accurately, we believe that the bootstrap learning scheme combined with automatic alignment can also be used for other kinds of audio segmentation and classification.

The initial purpose of this project is to aid the research of creating high-quality music synthesis by training on data from real music performances. Our system learns how amplitude and frequency vary, not only with time, but as a function of information in the score such as pitch, duration, and neighboring notes. Thus, the training data consists of notes (as audio) paired with symbolic data from the music score. This requires an audio-to-score alignment process.

We previously developed a polyphonic audio alignment system and effectively deployed it in several applications (Hu, Dannenberg & Tzanetakis, 2003; Dannenberg & Hu, 2003). However, the alignment is not precise enough to isolate the initial "attack" portion of tones. For example, the attack of a typical trumpet tone lasts only about 30 ms (see Fig. 1). But due to limits imposed by the acoustic features used for alignment, the size of the analysis window is usually 50 to 250 ms, which is too large to accurately pinpoint note onsets. If note boundaries are not detected precisely, the learning system will not receive good examples of attacks which are important for high quality synthesis.

Moreover, we want our system to learn from acoustic examples and corresponding scores without much human intervention, thus no manual data labeling should be involved. Therefore, we pursue *accurate* audio alignment with a resolution of several milliseconds. Thus, our study is motivated and characterized by three goals: (1) very high precision, on the order of milliseconds, (2) very high accuracy, which can only be obtained through the use of a symbolic score and score alignment, and (3) no need for manual onset labeling to provide training examples. While our previous work addresses some of these issues, we believe this new approach is superior in terms of satisfying all three requirements. As an added benefit, the onset detector works very well without the aid of a symbolic score, once trained by this bootstrap learning method.

Because our note detector was initially developed for music synthesis data, our initial experiments use monophonic audio.We have extended this work to deal with polyphonic music, just to demonstrate that this approach can also work well in polyphonic cases.

The audio-to-score alignment process is closely related to that of Orio and Schwarz (2001), who also use dynamic time warping to align polyphonic music to scores. While we use the chromagram (described in a later section), they use a measure called Peak Structure Distance, which is derived from the spectrum of audio and from synthetic spectra computed from score data. They also intend to use their system to generate training examples for music synthesis (Schwarz, 2004), and they obtain accurate alignment using small (5.8 ms) analysis windows. The average error reported is about 23 ms (Soulez, Rodet & Schwarz, 2003), which makes it possible to directly generate training data for an onset detection system. However, such a small analysis window also greatly affects the efficiency of the alignment process. They report that even with optimization measures, their system is running "2 hours for 5 minutes of music, and occupying 400 MB memory". In contrast, our system uses larger analysis windows and aligns 5 minutes of music in less than 5 minutes. Although we use larger analysis windows for alignment, we use small analysis windows (and different features) for note segmentation, and this allows us to obtain high accuracy.

Note that the best features for score alignment, which typically relies on melody and harmony, may not be the best features for detecting segment boundaries, which are also characterized by rapid changes in energy and noise content. Our approach recognizes this difference and uses different features for onset detection verses alignment.

In the following sections, we describe our system in detail. We introduce the audio-to-score alignment process in Section 2, and the note detection model in Section 3. Section 4 describes the bootstrap learning method in detail. Section 5 evaluates the system and presents some experimental results. We conclude and summarize this paper in the last section.

2. Audio-to-score alignment

2.1. The chroma representation

As mentioned above, the alignment is performed on two sequences of features extracted from both the symbolic and audio data. As described in a previous study (Hu, Dannenberg & Tzanetakis, 2003), we looked at the ability of different features to discriminate matching music from non-matching music when aligning scores to recorded audio. The chroma representation is clearly the winner for this task compared to several other features including spectrograms and cepstral coefficients.

Thus, our first step is to convert audio data into *discrete chromagrams*: sequences of chroma vectors. The chroma vector representation is a 12-element vector, where each element represents the spectral energy corresponding to one pitch class (i.e., C, C#, D, D#, etc.). To compute a chroma vector from a magnitude spectrum, we assign each bin of the FFT to the pitch class of the nearest step in the chromatic equal-tempered scale. Then, given a pitch class, we average the magnitude of the corresponding bins. This results in a 12-value chroma vector. Each chroma vector in this work represents 50 ms of audio data (non-overlapping).

The symbolic data, i.e. MIDI file, is also converted into chromagrams. The obvious way is to synthesize the MIDI data and convert the synthetic audio into chromagrams. However, we have found a simple alternative that directly maps from MIDI events to chroma vectors (Hu, Dannenberg & Tzanetakis, 2003). To compute the chromagram directly from MIDI data, we first associate each pitch class with an independent unit chroma vector—the chroma vector with only one element value as 1 and the rest as 0; then, where there is polyphony in the MIDI data, the unit chroma vectors are simply multiplied by the MIDI loudness of notes, added and normalized.

The direct mapping scheme speeds up the system by skipping the synthesis procedure, and it rarely sacrifices any quality in the alignment results. In most cases we have tried, the results are generally better when using this alternative approach. Furthermore, when audio is rendered from symbolic (MIDI) data, there can be small timing variations in practice. Since we are aiming for highly accurate results, it is desirable to bypass this source of error.

2.2. Matching MIDI to audio

After obtaining two sequences of chroma vectors from the audio recording and MIDI data, we need to find a time correspondence between the two sequences such that corresponding vectors are similar. Before comparing the chroma vectors, we normalize the vectors, as obviously the amplitude levels vary throughout the acoustic recordings and MIDI files. We experimented with different normalization methods, and normalizing the vectors to have a mean of zero and a variance of one seems to be the best one. But this can cause trouble when dealing with *silence*. Thus, if the average amplitude of an audio frame is lower than a predefined threshold, we define it as a silence frame. We then calculate the Euclidean distance between the vectors. The distance is zero if there is perfect agreement. If exactly one of the two compared chroma vectors is a silence frame, we assign the distance a predefined value d_{max}. Choosing an adequate value of d_{max} is important as it may affect the alignment quality. This will be discussed in more detail after the alignment algorithm is described.

Figure 2 shows a similarity matrix where the horizontal axis is a time index into the acoustic recording, and the vertical axis is a time index into the MIDI data. The intensity of each point is the distance between the corresponding vectors, where black represents a distance of zero. A *path* in this matrix is a sequence of neighboring cells (including diagonal neighbors) from the lower left to the upper right corner. The *cost* of a path is the sum of the between-vector distances of all the cells in the path.

We use the Dynamic Time Warping (DTW) algorithm to find the lowest cost path, which we refer to as the optimal alignment. DTW computes a matrix (in addition to the similarity matrix) where each matrix cell (i, j) represents the sum of distances along the best path from

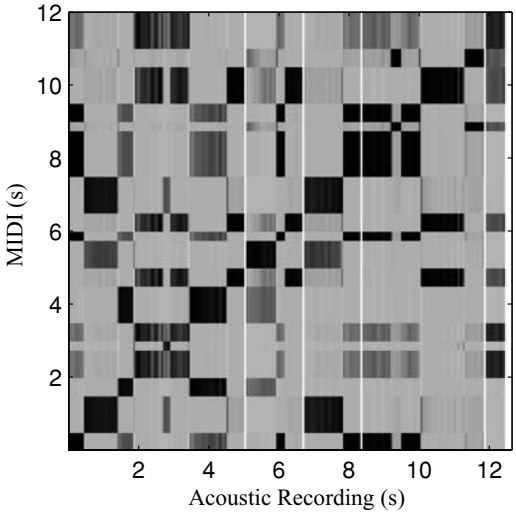

Fig. 2 Similarity Matrix for the first part in the third movement of English Suite composed by R. Bernard Fitzgerald. The acoustic recording is the trumpet performance by the second author

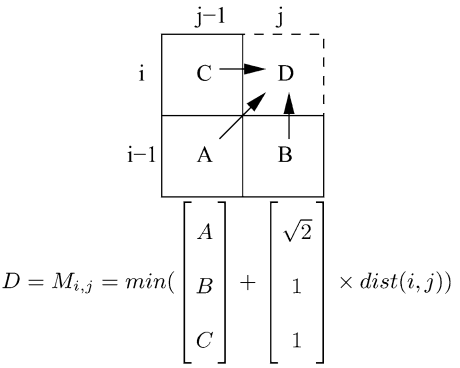

Fig. 3 Calculation pattern for cell (i, j)

(0, 0) to (i, j). We use the calculation pattern shown in Fig. 3 for each cell. The best path up to location (i, j) in the matrix (labeled in the figure) depends only on the adjacent cells (A, B, and C) and the weighted distance between the vectors corresponding to row i and column j. Note that the horizontal step from C and the vertical step from B allow for the skipping of silence in either sequence. We also weight the distance value in the step from cell A by $\sqrt{2}$ so as not to favor the diagonal direction. This calculation pattern is the one we feel most comfortable with, but the resulting differences from various formulations of DTW (Hu & Dannenberg, 2002) are often subtle. The DTW algorithm requires a single pass through the matrix to compute the cost of the best path. Then, a back-tracing step is used to identify the actual path.

As we mentioned earlier, when any of the compared chroma vectors is a silence frame, the distance between the two is defined as d_{max}. d_{max} should have the value that neither penalizes horizontal or vertical direction of the alignment path, nor makes matching silence to non-silence frames impossible. Otherwise, the detection of those note onsets immediately after silence frames can be slightly delayed, as DTW may be prone to local minima. In our experiments, we set d_{max} to 16, which is suitable for this kind of data.

The time complexity of the automatic alignment is $O(mn)$, where m and n are respectively the lengths of the two compared feature sequences. Assuming the expected optimal alignment path is near the diagonal, we can optimize the process by running DTW on just a part of the similarity matrix, which is basically a diagonal band representing the allowable range of misalignment between the two sequences. However, if the width of the diagonal band of interest is a constant fraction of its length, the time complexity is still $O(mn)$.

After computing the optimal path found by DTW, we get the time points of the note onsets in the MIDI file and map them to the acoustic recording according to the path (see Fig. 4).

The analysis window used for alignment is $W_a = 50$ ms. A smaller window actually makes the alignment worse because of the way chroma vectors are computed. Thus the alignment result is really not that *accurate*, considering the alignment resolution is on the same scale as the analysis window size. Nevertheless, the alignment path still indicates roughly where the note onsets should be in the audio. In fact, the statistical error between the actual note onsets and the estimated ones found by the alignment path appears to be approximately Gaussian. (Figure 5 shows the histogram of such an error distribution from 154 synthetic note onset samples.) In other words, the probability of observing an actual note onset around an estimated one given by the alignment can be roughly estimated by a Gaussian distribution. This is valuable information that can help train the segmenter.

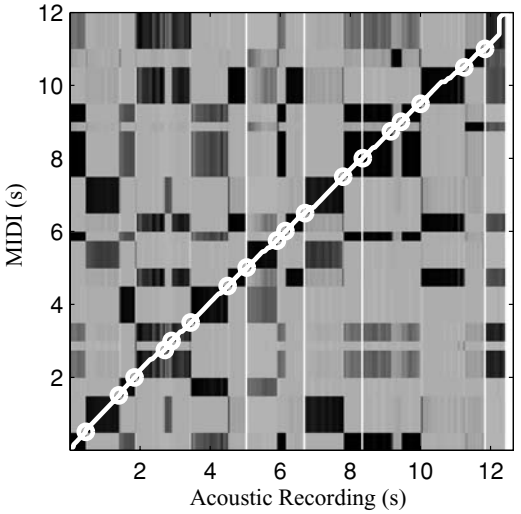

Fig. 4 The optimal alignment path is shown in white over the similarity matrix of Fig. 2; the little circles on the path denote the mapping of note onsets

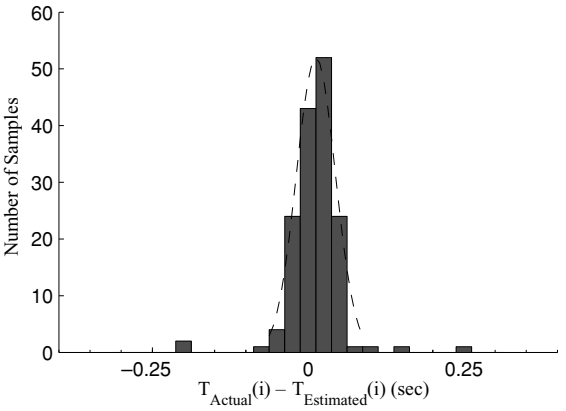

Fig. 5 Histogram of estimated onset error distribution. $T_{estimated}(i)$ denotes an estimated note onset given by the alignment, while $T_{actual}(i)$ denotes the corresponding actual note onset manually labeled in acoustic recordings. Here $1 \leq i \leq 154$. The dotted curve represents a Gaussian window function with a standard deviation about half the size of the alignment analysis window $(0.5 \times 0.05s = 0.025s)$

3. Onset detection

3.1. Acoustic features

Several features are extracted from the acoustic signals. The basic ones are listed below:

– Logarithmic energy of the frame (measured in decibels), distinguishing silent frames,

$$LogEng = 10log_{10}Energy.$$

– Fundamental frequency $F0$. Fundamental frequency and harmonics are computed using the McAulay-Quatieri model (McAulay & Quatieri, 1986) provided by the SNDAN package (Beauchamp, 1993).

– Relative strengths of first three harmonics

$$RelAmp_i = \frac{Amplitude_i}{Amplitude_{overall}},$$

where i denotes the ith harmonic.
– Relative frequency deviations of first three harmonics

$$RelDFr_i = \frac{f_i - i \times F0}{f_i},$$

where f_i is the frequency of the ith harmonic
– Zero-crossing rate (ZCR), serving as an indicator of the noisiness of the signal.

Furthermore, the derivatives of these features are also included, as derivatives are good indicators of fluctuations in the audio such as note attacks or fricatives.

Features are computed using a sliding non-overlapping analysis window W_s with a size of 5.8 ms. If the audio to be processed has a sampling rate of 44.1 kHz, every analysis window contains 256 samples. Other features that have proved to be useful in audio segmentation (Plumbley, Brossier & Bello, 2004) could be added to this set and might improve performance.

3.2. Onset detection model

We use a multi-layer perceptron (MLP) Neural Network as the onset detection model (see Fig. 6). It is a feed-forward network with two hidden layers, and is fully connected between each layer. The first hidden layer contains 6 neurons and the second 4 neurons. Each neuron (perceptron) is a Sigmoid unit, which is defined as $f(s) = \frac{1}{1+e^{-s}}$, where s is the input of the neuron, and $f(s)$ is the output. The input units accept the features extracted from the acoustic signals. The output is a single real value ranging from 0 to 1, indicating the likelihood that the current audio frame is a segmentation point. In other words, the output is the model's estimate of the certainty of a note onset. When using the model to detect onsets in the audio file, an audio frame is classified as a note onset if the output of the detector is more than 0.5.

Neural networks offer a standard approach for supervised learning. Labeled data are required to train a network. Training is accomplished by adjusting weights within the network to minimize the expected output error. We use a conventional back-propagation learning method to train the model.

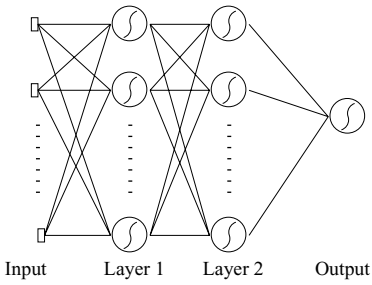

Fig. 6 Neural network for onset detection

It should be noted that the emphasis of this work is to demonstrate that the alignment information can help train the onset detector and improve its performance via bootstrap learning. We chose a small neural network with two hidden layers based on the proof by the Kolmogorov Theorem (Kurková, 1992) that an MLP with two hidden layers is theoretically sufficient to model any problem. This model does work well, so it is certainly adequate for our study. We would expect a neural network with a single hidden layer or some other supervised learning approach to also work well.

4. Bootstrap learning

Making use of unlabeled data has been a hot topic in machine learning. Standard approaches are usually classified as unsupervised (use only unlabeled data) or semi-supervised (use both unlabeled and labeled data) learning. If we can find some way to transform an unsupervised learning problem into a supervised learning problem, we might gain huge benefits, as supervised learning is better understood and offers many well-studied techniques.

In our study, the amount of labeled data is either very limited or synthetically generated. We overcome this problem by utilizing the global correspondence information provided by the alignment, which provides approximate note boundaries. Our approach uses a supervised learning model, such as a neural network, to iteratively improve its own training data. Adopting terminology from related work (Kuipers & Beeson, 2002), we call this *bootstrap learning*.

Our bootstrap learning system starts by creating a function representing the probability of a note starting at a particular time point in the acoustic recording, based on estimated note onsets computed from the alignment path. As shown in Fig. 7, the function is generated by summing up a set of Gaussian window functions. Each window function is centered at the estimated note onsets given by the alignment path, and its standard deviation is half the size of the alignment analysis window as in Fig. 5 ($\sigma = 0.5 \times 0.05s = 0.025s$). For those points

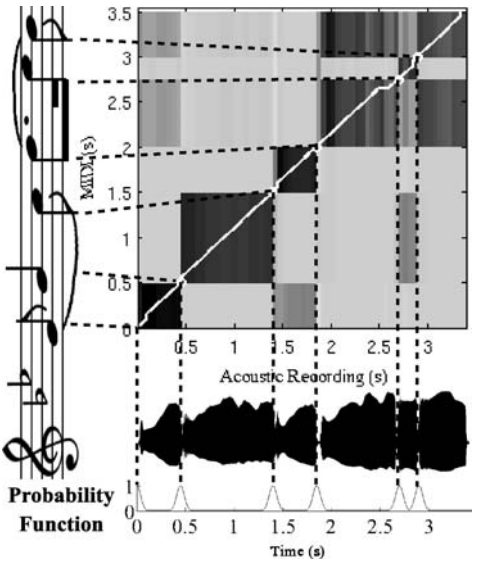

Fig. 7 Probability function generated from the alignment of a snippet, which is a phrase from the piece shown in Fig. 2

outside any Gaussian function window, the value is assigned to a small value slightly bigger than 0 (e.g., 0.04).

The creation of the probability function is based on the assumption that note onsets are independent of each other. We should also point out that this is just a pseudo-probability function, and the function integral is not necessarily equal to 1.

We also need to initialize the onset detection model. The neural network needs to be trained first to get initial weights. The training data for initialization are generated automatically by synthesizing onsets at known times. Synthesized audio is *not* equivalent to recorded audio from acoustic instruments, but this simplifies the initialization and avoids the need to label recordings by hand.

Then we run the following steps iteratively until the outputs of the trained neural network are unchanged by a round of training or testing error of the validation dataset starts to increase. Usually the training process will stop after 6 to 10 iterations.

1. Execute onset detection process on the acoustic audio.
2. Multiply the sequence of real values v output by the segmenter with the note onset probability function computed from the alignment. The result is a new sequence of values denoted as v_{new}.
3. For each note onset estimated by alignment, find a time point that has the biggest value v_{new} within a window W_p, and mark it as the adjusted note onset. The window is defined as follows:

$$W_p(i) = \left[\max\left(\frac{T_i + T_{i-1}}{2}, T_i - W_a\right),\ \min\left(\frac{T_i + T_{i+1}}{2}, T_i + W_a\right)\right],$$

   where T_i is the estimated onset time of the ith note in the acoustic recording given by alignment, and W_a is the size of the analysis window for alignment.
4. Use the audio frames to re-train the neural network. The adjusted note onset points are labeled as 1, and the rest are labeled as 0. Because the number of positive examples is far less than the negative ones, we increase the penalty by a factor of 300 when false negatives occur. This factor is chosen to approximate the ratio of non-onset frames to onset frames.

To prevent data over-fitting, we use 5-fold cross-validation. The iterative training process will stop when the local minimum of the validation set error is found.

As the onset detection model has a finer resolution than the alignment model, the trained onset detector can detect note boundaries in audio signals more precisely, as demonstrated in Fig. 8.

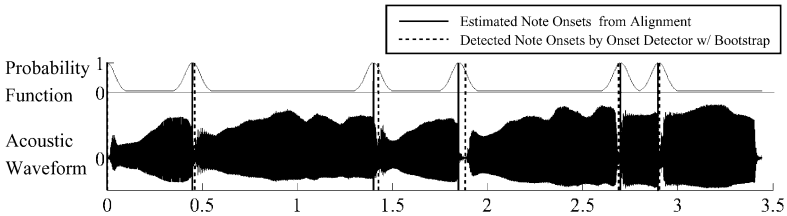

Fig. 8 Onset detection results after 8 iterations of bootstrap learning on the same music content as in Fig. 7. Note that the note onsets marked by the detector with bootstrapping are not exactly the same as the ones estimated from alignment. This is best illustrated on the note boundary around 1.9 seconds, where the detected onset is 32 ms later than the one estimated from alignment

5. Evaluations

5.1. Monophonic experiments

As mentioned in previous sections, the system was initially designed for helping with music synthesis research, which uses monophonic recordings as training examples. Thus, we were mainly focusing on monophony. The experimental data for monophonic music is the English Suite transcribed by R. Bernard Fitzgerald (1955) for Bb Trumpet. This is a set of 5 English folk tunes artfully arranged into one work. Each of the 5 movements (recorded without accompaniment) is essentially a monophonic melody, and the whole suite contains a total of 673 notes. We have several formats of this particular music piece, including MIDI files created using a digital piano, the real acoustic recordings performed by the second author, and synthetic audio generated from the MIDI files.

We conducted experiments to measure the impact of bootstrap learning. The baseline is just a neural network previously described as the onset detection model, and it weights are initialized by training on a set of labeled data: an arbitrarily picked MIDI file (monophonic melody of the Beatles' song "Let It Be") and an audio recording synthesized from it. The baseline detector represents the model *before* bootstrap learning. The onset detector with bootstrapping has the same initial configuration as the baseline, and then it is iteratively trained on the alignment information. Naturally this represents the model *after* bootstrap learning.

To measure the performance of two onset detectors, we run the baseline onset detector through all the audio files in the data set and compare its detected note onsets with the actual ones. For the onset detector with bootstrapping, we use cross-evaluation. In every evaluation pass, 4 MIDI files and the corresponding audio files are used to train the segmenter through the bootstrap method, and the remaining MIDI-audio files pair is used as the test set. This process is repeated so that the data of all 5 movements have been used once for evaluation, and the results on the test sets are combined to evaluate the model performance. It is important to note that *when testing the segmenter with bootstrap learning, score alignment is not used.* If the score were used (and this is in fact how we derive data for our synthesis project), the miss rate and spurious rate would normally be zero because the one-to-one alignment of audio notes to symbolic notes leaves little opportunity for these kinds of errors.

We do not need to use a similar cross-validation scheme on the baseline onset detector because it uses the dataset merely for testing. One may ask why the data is not used for training. The reason is that the data is deemed as unlabeled when used to train the onset detector with bootstrapping, but it would need to be fully labeled to train the baseline onset detector. Furthermore, if we hand-labeled the data, any performance differences between the two models would be due to the quality of the hand-labeled data verses the quality of data generated by the bootstrap learning process. While this might be theoretically interesting, the hand-labeled data is simply unavailable in a realistic scenario, and that is exactly why we introduce bootstrapping.

We calculate several values to measure the performance of the systems. Miss rate is defined as the ratio of missed note onsets among all the labeled data—a labeled note onset is deemed as *missed* when there is no detected onset within the window W_p around it. Spurious rate is the ratio between spurious onsets detected by the system and all the labeled note onsets—spurious note onsets are those detected that do not correspond to any labeled onset. Average error and standard deviation (STD) are obtained by measuring the distance between each labeled note onset and its corresponding detected onset, if the note onset is neither missed nor spurious.

Table 1 Model comparison on synthetic monophonic audio

Model	Miss rate	Spurious rate	Average error	STD
Baseline Onset detector	8.8%	10.3%	21 ms	29 ms
Onset detector w/ Bootstrap	0.0%	0.3%	10 ms	14 ms

Table 2 Model comparison on real monophonic recordings

Model	Miss rate	Spurious rate	Average error	STD
Baseline Onset detector	15.0%	25.0%	35 ms	48 ms
Onset detector w/ Bootstrap	2.0%	4.0%	8 ms	12 ms

We first use the synthetic audio from MIDI files as the data set, and the experimental results are shown in Table 1. Since the baseline detector is trained with data similar to that used for the bootstrapped detector, the differences in performance are mainly due to the greater quantity of training data available to the bootstrapped detector.

We also tried the two onset detectors on the acoustic recordings. However, it is very difficult to evaluate results because labeling all the note onsets in acoustic recordings is too time consuming. Therefore, we randomly picked a set of 100 note onsets throughout the music (20 in each movement), and labeled their onsets manually. The results are shown in Table 2. Some of the timing error could actually be due to error in the labels, but the clear trends in the data indicate that the bootstrap detector is better than the baseline detector. In this comparison, the differences in performance have two sources: The bootstrapped detector has more training data because it can use unlabeled data, and the data for the bootstrapped detector includes trumpet recordings similar to the data used for evaluation. If we could label all this data and use it to train the baseline detector, we would expect essentially equal performance from both systems. However, as explained earlier, labeling is expensive. The bootstrap detector offers good performance without the need for manual labeling.

5.2. Polyphonic experiments

We also tried the bootstrap learning method on polyphonic music. The experimental data are 3 piano pieces. They are F. Chopin, Nocturne Op. 9, No. 2; L. van Beethoven, Adagio "Moonlight" Sonata; and W.A. Mozart, Turkish March. As in the monophonic experiments, we used several formats of the pieces, including MIDI files, synthetic audio generated from MIDI files, and recordings of acoustic performances by the first author.

The polyphonic experiments are very similar to the monophonic experiments, and we count multiple simultaneous note onsets as one. Tables 3 and 4 show the experimental results on synthetic audio and real recordings respectively. Compared to experimental results on monophonic music, the polyphonic results are slightly worse with both onset detectors. Piano onsets are generally considered more prominent and easier to detect than trumpet onsets, but on the other hand, polyphony is generally considered more difficult to analyze than monophony. With polyphony, note onsets can be masked by other sounding notes. Also, our features may not be well-chosen for polyphonic music. Nevertheless, the onset detector

Table 3 Model comparison on synthetic polyphonic audio

Model	Miss rate	Spurious rate	Average error	STD
Baseline Onset detector	10.8%	12.1%	28 ms	61 ms
Onset detector w/ Bootstrap	1.2%	0.4%	12 ms	20 ms

Table 4 Model comparison on real polyphonic recordings

Model	Miss rate	Spurious rate	Average error	STD
Baseline Onset detector	16.0%	17.0%	27 ms	55 ms
Onset detector w/ Bootstrap	5.0%	3.0%	11 ms	23 ms

with bootstrapping significantly outperforms the baseline one, and its results are comparable to other published works (Marolt, Kavcic & Privosnik, 2002). This indicates that our bootstrap learning method works well with polyphony.

In all the experiment results presented here, bootstrap learning shows significant improvement over the original onset detector. However, this approach still has its limitations. First of all, it depends on the accuracy of the alignment path found by DTW. In our experiments, it is not a problem when a few onsets estimated by alignment are not very close to the actual ones. But if the alignment path makes too many mistakes or the MIDI file and real recordings do not basically correspond to each other, it is unlikely that bootstrapping can improve on the labels it is given. Secondly, there might be problems with over-fitting even though cross-validation is being used. A solution to that is to increase the size of the alignment data set, thus both the training set and the test set contain more data.

6. Conclusions

Onset detection is an important step in many music processing tasks, including beat tracking, tempo analysis, music transcription, and music alignment. However, onset detection at note boundaries is rather difficult. In real recordings, the end of one note often overlaps the beginning of the next due to resonance in acoustic instruments and reverberation in the performance space. Even humans have difficulty deciding exactly where note transitions occur. One promising approach to good onset detection is machine learning. With good training data, supervised learning systems frequently outperform those created in an ad hoc fashion. Unfortunately, we do not have very good training data for onset detection, and labeling acoustic recordings by hand is very tedious and time consuming.

Our work offers a solution to the problem of obtaining good training data. We use music alignment to tell us (approximately) where to find note boundaries. This information is used to improve the note segmentation, and the segmentation can then be used as labeled training data to improve the onset detector. This bootstrapping process is iterated until it either converges or minimizes testing error.

Our tests show that note onset detection can be dramatically improved using this approach. When a symbolic score is available (which must be the case at least for the bootstrap training data), the bootstrap process gives more accurate onset times than either the baseline

non-bootstrapped onset detector or the alignment process working alone, and alignment guarantees zero missing or spurious detection errors. Once trained, the bootstrapped onset detector also shows improved performance on new unlabeled data. One key advantage of the bootstrapping process is that it can learn from unlabeled acoustic recordings, while a non-bootstrapped onset detector is either restricted to synthetic training data, where onset times are known, or to expensive and therefore limited data that is labeled by hand.

Supervised learning models such as Neural Networks can scale well with more feature inputs. In future work, we hope to improve further on onset detection by considering more signal features and using multiple window sizes to incorporate features at different time scales. This will require more training data, but our bootstrap learning method should make this feasible.

In summary, we have described a system for music note onset detection that uses alignment to provide an initial set of labeled training data. A bootstrap learning method is used to improve both the labels and the onset detector. Onset detectors trained in this manner show improved performance over a baseline detector. Our bootstrap learning approach can be generalized to incorporate additional signal features and other supervised learning algorithms. This method is already being used to segment acoustic recordings for a music synthesis application, and we believe many other applications can benefit from this new approach.

Acknowledgments This project greatly benefits from the helpful discussions and suggestions during the Computer Music group meetings held at Carnegie Mellon University. We would also like to thank Guanfeng Li for his valuable input and support.

References

Beauchamp, J. (1993). Unix workstation software for analysis, graphics, modifications, and synthesis of musical sounds. *AES Convention,* preprint 3479. New York: Audio Engineering Society.

Dannenberg, R. B. & Hu, N. (2003). Polyphonic audio matching for score following and intelligent audio editors. In *Proceedings of the 2003 International Computer Music Conference* (pp. 27–34). San Francisco: International computer music association.

Downie, J. S., West, K., Ehmann, A., & Vincent, E. (2005). *The 2005 Music Information Retrieval Evaluation Exchange (MIREX 2005): Preliminary Overview* (pp. 320–323). *ISMIR 2005: 6th International Conference on Music Information Retrieval Proceedings* (pp. 288–295). London: Queen Mary, University of London.

Fitzgerald, R. B. (1955). *English suite*: *For Bb trumpet or cornet and piano.* Bryn Mawr: Theodore Presser Co.

Hu, N. & Dannenberg, R. B. (2002). A comparison of melodic database retrieval techniques using sung queries. In *JCDL 2002: Proceedings of the Second ACM/IEEE-CS Joint Conference on Digital Libraries* (pp. 301–307). New York: ACM.

Hu, N., Dannenberg, R. B., & Tzanetakis, G. (2003). Polyphonic audio matching and alignment for music retrieval. *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics* (pp. 185–188). New York: IEEE.

Kapanci, E. and Pfeffer, A. (2004). A hierarchical approach to onset detection. In *Proceedings of the 2004 International Computer Music Conference* (pp. 438–441). San Francisco: International Computer Music Association.

Kuipers, B. & Beeson, P. (2002). Bootstrap learning for place recognition. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence* (pp. 174-180). Menlo Park, CA: AAAI Press.

Kurková, V. (1992). Kolmogorov's theorem and multilayer neural networks. *Neural Networks, 5*(3), 501–506.

Lu, L., Li, S. Z., & Zhang, H. J. (2001). Content-based audio segmentation using support vector machines. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME 2001)* (pp. 956–959). New York: IEEE.

Marolt, M., Kavcic, A., & Privosnik, M. (2002). Neural networks for note onset detection in piano music. http://lgm.fri.uni-lj.si/ matic/SONIC.html.

McAulay, R. J. & Quatieri, T. F. (1986) Speech analysis/synthesis based on a sinusoidal representation. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, *34*(4), 744–754.

Muller, M., Kurth, F., & Clausen, M. (2005). Audio matching via chroma-based statistical features. *ISMIR 2005: 6th International Conference on Music Information Retrieval Proceedings* (pp. 288–295). London: Queen Mary, University of London.

Orio, N. & Schwarz, D. (2001). Alignment of monophonic and polyphonic music to a score. In *Proceedings of the 2001 International Computer Music Conference* (pp. 155–158). San Francisco: International Computer Music Association.

Plumbley, Mark D., Brossier, P. M., & Bello, J. P. (2004). Fast labelling of notes in music signals. *ISMIR 2004 Fifth International Conference on Music Information Retrieval Proceedings* (pp. 331–336). Barcelona, Spain: Universitat Pompeu Fabra.

Raphael, C. (1999). Automatic segmentation of acoustic musical signals using hidden markov model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *21*(4), 360–370.

Raphael, C. (2004). A hybrid graphical model for aligning polyphonic audio with musical scores. In *Proceedings of the 5th International Conference on Musical Information Retrieval* (pp. 387–394). London: Queen Mary, University of London.

Schwarz, D. (2004). *Data-driven concatenative sound synthesis* (PhD thesis) Paris, France: Universit Paris 6-Pierre et Marie Curie.

Soulez, F., Rodet, X., & Schwarz, D. (2003). Improving polyphonic and poly-instrumental music to score alignment. *ISMIR 2003* In *Proceedings of the Fourth International Conference on Music Information Retrieval* (pp. 143–148). Baltimore, MD: Johns Hopkins University.