Note onset detection for the transcription of polyphonic piano music

C. Gregor van den Boogaart, Rainer Lienhart

Note onset detection for the transcription of polyphonic piano music

C. G. v. d. Boogaart, R. Lienhart

Report 2009-11 Mai 2009

NOTE ONSET DETECTION FOR THE TRANSCRIPTION OF POLYPHONIC PIANO MUSIC

C. G. v. d. Boogaart, R. Lienhart

Multimedia Computing Lab
University of Augsburg, 86159 Augsburg, Germany
{boogaart,lienhart}@multimedia-computing.org

ABSTRACT

Transcription of music is the process of generating a symbolic representation such as a score sheet or a MIDI file from an audio recording of a piece of music. A statistical machine learning approach for detecting note onsets in polyphonic piano music is presented. An area from the spectrogram of the sound is concatenated into one feature vector. A cascade of boosted classifiers is used for dimensionality reduction and classification in an one-versus-all manner. The presented system achieves an accuracy of 87.4 % in onset detection outperforming the best comparison system by 25.1 %.

Index Terms— Acoustic signal detection, Spectral analysis, Feature extraction, Pattern classification

1. INTRODUCTION

A trained musician is able to write down the scores from listening to the recording of a song. In this paper a system is presented, which is able to perform a similar task, by creating a MIDI file from the recording of a song. The accurate symbolic representation of a song has many useful applications. It can easily be read by humans and machines and reveals the core essence of the music, which is more or less hidden in a recording. Examples of such information are: the key, the chord progression, and the melody line.

The main issue in music transcription is the superposition of the notes in the music recording. Any sort of detector for a note has to face the interference from an unknown combination of other notes sounding at the same time. The interference is hard or even impossible to model. Superposition therefore is the hard key issue in music transcription and other music understanding tasks.

Contributions: In this work a system is constructed, which produces a transcription in form of a MIDI file from the input of a music audio recording. The main contributions are: Detectors based on *statistical machine learning*, trained on example data are used instead of heuristic F0 estimation techniques. The detectors are trained for the detection of *note onsets* instead of a frame-wise detection of the presence of the note in its sustain phase. This has two advantages: First the onset is the most characteristic part of the note, which gives a very distinctive, instrument specific pattern. Second the note onset carries not only the pitch information, but also information about the rhythm, which makes it of great value for the transcription. As the onset is very characteristic, consequently not generic but *instrument specific classifiers* are employed. To enable the representation of the characteristic onset pattern, a rich feature set is necessary. Therefore high dimensional *spectrogram features* are used instead of heuristically designed features. *Several time frame vectors are concatenated* to form one feature vector, representing not only one time frame, but an area of the spectrogram. This allows profiting from the development of the sound over time during the onset as information source.

Related Work: State of the art systems for music transcription use a wide range of different methods and are designed for different application scenarios. The first type of system relies on typical properties of notes, such as the structure of fundamental frequency and harmonics or a sudden increase in energy at the onset, which are directly built into the system manually (e.g. [1, 2]). The second type of system uses statistical methods, such as classifiers [3] or distribution models (e.g. [4]). There also exist some approaches for onset detection independent of the current note (e.g. [5]).

2. SYSTEM OVERVIEW

The presented system consists of an array of classifiers, one for each note of the piano (see figure 1). The same feature vector is provided to all classifiers. The classifiers are either trained for note onset detection or for frame-wise detection of the presence of a note.

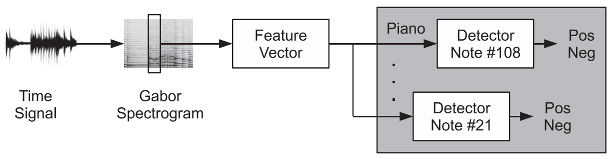

Fig. 1. From the time signal the spectrogram and finally the feature vector is generated. It is provided to all note detectors.

The *dimensionality* of the input feature set is quite high. This challenge is tackled by using Discrete AdaBoost [6] as classifier, which is capable of dealing with high dimensional input features, since it is not only a meta-classifier, but also a feature selection technique. Stumps are employed as weak classifiers. Boosting additionally has the advantages of being quite robust against overfitting [6] and of having similar discrimination performance as non-linear classifiers such as SVMs and Neural Networks (see [7]).

Note onsets are rare and short events, so the number of negative examples is much higher than the number of positive examples resulting in an *unbalanced learning problem.* This is tackled by creating a cascade of Discrete AdaBoost classifiers, see [7]. It instantly sorts out a lot of easy to detect false examples, saving time for cases harder to decide on.

The time-frequency representation of an audio signal such as a single note is sparse in nature (e.g. [8]). Thus different notes are not overlapping in all parts of the spectrogram. Therefore the rich set of spectrogram features together with the feature selection step in the classifier give rise to the hope of being able to tackle the challenge of superposition.

3. FEATURES

The features should carry valuable information about the task at hand. In the case of music transcription, the desired information is the pitch height or equivalently the MIDI note number of a note currently played, the attack time of the onset of the note and the timbre or instrument. All this is coded directly in the spectrogram of the sound. The harmonic signal parts, i.e. the fundamental frequency and overtones, encode the pitch and partly the timbre and form more or less horizontal lines in the spectrogram. Non harmonic, i.e. noise-like or non-deterministic signal parts encode the attack time in form of transients and the exact instrument or timbre. They form more texture-like than structured contributions in the spectrogram. Therefore we directly use the spectrogram of the signal as features.

Gabor Transform: The Gabor transform with Gaussian window [9] is used for generating the spectrogram features, because it perfectly localizes the information in time and frequency. The spectrogram is the logarithm of the modulus of the complex values of a Gabor transform. The modulus removes the phase, which is crucial as relative and absolute phases carry no important sound information, but can vary arbitrarily. The logarithm serves as nonlinear compression of the number range.

The Gabor transform is implemented as described in [9]. The sampling rate in the system is $f_s = 1/T = 48\,\text{kHz}$. The Gaussian window $g(t) = \frac{1}{\sqrt{2\pi\sigma_t^2}} \exp\left(-\frac{1}{2}\frac{t^2}{\sigma_t^2}\right)$ has a standard deviation of $\sigma_t = 14.4\,\text{ms}$. The window length is chosen with $(N = 1810)$ as $2N + 1 = 3621$ samples. The oversampling-factor of the Gabor lattice is $q = \frac{1}{\Delta t \Delta f} = 5$. The lattice constants are $\Delta t = 22.8\,\text{ms}$ and $\Delta f = 8.79\,\text{Hz}$ resulting in a hop size of 1092 samples. The signal $x(t)$ and the window $g(t)$ are sampled at times $t = kT$. With the normalization factor $L = \sqrt{q \sum_{k=-N}^{N-1} |g(kT)|^2}$ the Gabor transform values $X[n, m] = X(n\Delta t, 2\pi m \Delta f)$ are given as:

$$X[n, m] = \frac{1}{L} \sum_{k=-N+\frac{n\Delta t}{T}}^{N-1+\frac{n\Delta t}{T}} x(kT) g(kT - n\Delta t) e^{-2\pi j m b (kT - n\Delta t)}. \tag{1}$$

The logarithm of the modulus of the transform values are used as feature values, selecting only the values of the bins from 0 Hz up to 6 kHz ($0\,\text{Hz} \leq m\Delta f \leq 6\,\text{kHz}$). This is motivated by the fact that frequency content above about 6 kHz is only important for the sound quality, but not for music understanding and can therefore be skipped for better computational performance. With this setting the number of feature values per time frame is 683.

Concatenation of Feature Vectors: A single vector in time can be ambiguous e.g. due to interference of noise or due to more or less random parts of the signal. To overcome this, the information of several consecutive vectors should be evaluated together. In audio information retrieval Hidden Markov Models (HMM, [10]) are the common standard tool for the analysis of sequences, frequently used e.g. in speech recognition [10], chord recognition [11], and music transcription [3]. They model a fixed temporal order of a sequence, while the speed of the process generating the sequence can vary. This corresponds to a loose coupling of the time axis. The evolving of a single note over time is precisely defined. This results in a fixed coupling of the time axis. HMMs are therefore not suitable for onset detection.

Therefore we use a different, even simpler but computational demanding method, in order to use the temporal development as information source. By concatenating several consecutive feature vectors into one vector, a combined evaluation of the local time neighborhood is enabled. The concatenation is performed in overlapping and sliding manner, i.e. each time frame vector becomes the member of several concatenated feature vectors. Figure 2 illustrates this for concatenating 4 time frame vectors. A single square represents one time frame vector.

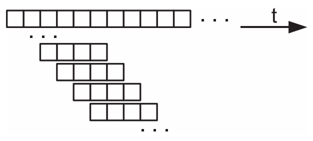

Fig. 2. Illustration of the overlapping and sliding concatenation of 4 time frame vectors into feature vectors. A single square represents one time frame vector.

For the typical setting of 4 time frame vectors forming one feature vector, the concatenation results in $683 \cdot 4 = 2732$ feature dimensions. The high dimensionality of spectrogram features, made worse by the concatenation is expected to be the main reason why spectrogram features and also concatenated feature vectors are not a common approach in music information retrieval. For comparison: Mel-Frequency-Cepstral-Coefficients (MFCC, [12]) and Chroma features ([13]), which are quite common audio features have a typical number of feature values of 13–30 (MFCCs) and 12 (Chroma) per time frame i.e. per feature vector.

4. CLASSIFIER

Note onsets are quite short and rare events in polyphonic music. Most of the time a given note is either not sounding or in the sustain or decay phase. Consequently the number of negative examples is much larger than the number of positive examples. A classifier for this task has to achieve not only a high recognition rate for positive examples, but also a very high rejection rate for negative examples. This can be addressed by building a cascade of Discrete AdaBoost classifiers. This has been introduced and successfully used for face detection in images, see [7]. Face detection similarly has to cope with an asymmetry in the frequency of class occurrences as a lot of positions and scales of images do not contain faces and correct positives are rare events.

Thus the classifier is build with cascaded Discrete AdaBoost classifiers [6, 7]. Boosting is a meta classifier technique, which builds a strong classifier out of a large number of simple or weak classifiers. During training an optimal subset is selected from a large pool of weak classifiers in order to form the ensemble of the strong classifier. Stumps (see below) are used as weak classifiers. A single stump is applied only to one dimension of the input features. As suitable stumps are selected during training, AdaBoost is also a feature selection technique.

Stumps: A stump is a degenerated decision tree with one split node and two leaf nodes [6]. It is applied to the element j of feature vector $\mathbf{x}$. It can be written as $h_t(\mathbf{x})$, with t a running index specifying the stump. The parameters of a specific stump are the dimension j, the threshold θ_t and the sign information $p_t \in \{-1, +1\}$. An example $\mathbf{x}$ is classified as follows:

$$h_t(\mathbf{x}) = \left\{ \begin{array}{rcl} +1 & : & \text{if } x_j \cdot p_t \geq \theta_t \\ -1 & : & \text{else} \end{array} \right. . \qquad (2)$$

Discrete AdaBoost: The first AdaBoost classifier for the cascade is trained as follows (see [6] for details):

- All positive examples for the current note are drawn. l is the number of positive examples, which could be drawn. It lies in the order of several hundred, depending on the size of the data set and the note number.
- The negative examples are drawn randomly. The number of negative examples m is limited to $m \leq 2l$.
- In each iteration the stump with the lowest error is selected and added to the ensemble. The iteration is stopped, if either the desired recognition rates are achieved or a maximum number of iterations is reached, which means that the training was not successful. The maximum is the smaller value of $l/2$ or 150.

In later stages only examples which are classified either as correct positives (positive examples) or false positives (negative examples) by the preceding stages are drawn.

A trained classifier consists of a set of stumps, together with a weight α_t for each stump and one global threshold value α. Given the feature vector $\mathbf{x}$ of an example, the final decision of such a classifier is made as follows:

$$h(\mathbf{x}) = \left\{ \begin{array}{rcl} 1 & : & \text{if } \sum_t \alpha_t h_t(\mathbf{x}) - \alpha \geq 0 \\ -1 & : & \text{else} \end{array} \right. . \qquad (3)$$

Cascade of Classifiers: The cascade (see [7]) sorts out negative examples rapidly, while preserving almost all positive examples. An example classified positive by the first classifier of the cascade is handed over to the next classifier for further investigation and so on. As soon as an example is classified negative during this chain, the example is classified negative by the whole cascade and the process is stopped.

The single classifiers are tuned via their threshold value α to achieve specific recognition rates. The goal is to accept almost all examples e.g. 0.999 of the positive examples as positive while accepting at most e.g. 0.5 or less of the negative examples as positive ones. The training iteration of a single classifier is stopped, as soon as enough weak classifiers are added in order to fulfill also the correct positive rate. If this cannot be achieved in the maximum number of iterations, the classifier is discarded and the cascade ends, otherwise training proceeds with the next classifier for this cascade. With this structure the overall expected rates converge very fast. For example with 20 cascades the expected false alarm rate is $0.5^{20} \approx 9.6e^{-7}$ and the expected hit rate is $0.999^{20} \approx 0.98$.

5. EXPERIMENTS

Two slightly different systems are tested, one for the detection of note onsets for which the system is intentionally designed and one for frame-wise detection of the presence of the note intended mainly for comparison with systems from literature. Both systems are also tested under noisy conditions.

Audio Data and Ground Truth: The MIDI files used in [3] were used as training and test data for the experiments. They originate from the Classical Piano Midi Page `www.piano-midi.de`. The dataset consists of 87 training and 26 test pieces. They were converted to sound files with timidity++ [14], a sample based MIDI software synthesizer and with a high quality General MIDI soundfont. The ground truth is directly extracted from the MIDI files as described in the following paragraphs.

Onset Detection: During training a concatenated feature vector is considered a positive example, if a note onset takes place during one selected part feature vector. Feature vectors which are not a positive example, but have nearby note onsets are removed from the training data. The remaining vectors are used as negative examples. During detection a postprocessing is applied in order to debounce the detector: after detection of an onset the detection of further onsets is suppressed for the following two feature vectors. During testing all vectors, which are not positive examples according to the previous rule, are negative examples. A misalignment of one time frame, which can easily result from binning issues, is still considered a correct recognition.

Frame-wise Detection: During training a vector is considered a positive example, if a note is at least partly present in the time frame. Time frames after a note off command are removed from the training data to remove the reverberation. The remaining examples are used as negative examples.

Results: The recognition results are compared with the system in [3], which itself contains a comparison with [15] and [16]. For simplicity we use the accuracy values which are calculated from the number of true positives (TP), false positives (FP) and false negatives (FN) as follows:

$$Acc = \frac{TP}{FP + FN + TP}. \qquad (4)$$

The training data naturally contains very few, sometimes even no positive examples for very high and very low pitches, especially for the onset detection case. Therefore of the grand piano keyboard range 21 to 108, only onset detectors for the note numbers 36 to 96 have been trained, which cover almost all notes played. For a fair comparison with the other systems which also had detectors for the notes outside the inner range, dummy onset classifiers which always predict negative are used. For the frame-wise detection, all classifiers with at least 1 positive example are trained.

To test the influence of the concatenation of several vectors, the performance of the onset detection for 1, 2, 4 and 8 concatenated vectors has been tested. The results are given in table 1. The gain in performance for an increasing number of vectors is obvious. As expected the additional data is valuable and the evolution of the note over time can be evaluated beneficially. As the performance degrades with 8 vectors again, 4 concatenated vectors have been used for the following experiments.

Number of vectors	1	2	4	8
Number of features	683	1366	2732	5464
Length in ms	22.8	45.5	91.0	182.0
Accuracy	54.1 %	73.8 %	87.4 %	76.5 %

Table 1. Onset detection with different numbers of vectors.

The results for the main experiment of onset detection are given in table 2 (a). The new system clearly outperforms the existing ones. Table 2 (b) shows the results for the frame-wise detection. Again the existing systems are outperformed.

Algorithm	(a) Onset	(b) Frame-wise
AdaBoost Cascade	87.4 %	75.2 %
Poliner and Ellis	62.3 %	67.7 %
Ryynänen and Klapuri	56.8 %	46.6 %
Marolt	30.4 %	36.9 %

Table 2. Accuracy (a) onset and (b) frame-wise detection.

Performance on Noisy Data: Pink noise is added to the test data. Table 3 shows the results depending on the SNR. A significant amount of noise robustness is achieved: The accuracy degrades only a little bit for moderate noise levels of SNR $\geq$ 30 dB and is still acceptable for a SNR of 20 dB, especially for the onset detection. It sharply falls at higher noise levels. Preliminary experiments showed that using noisy training data enhances the accuracy for testing on similar and higher noise levels while degrading the accuracy for lower noise levels only slightly.

SNR	(a) Onset	(b) Frame-wise
∞ dB	87.4 %	75.2 %
40 dB	84.3 %	68.9 %
30 dB	73.1 %	56.1 %
20 dB	46.5 %	34.2 %
10 dB	19.9 %	12.9 %
0 dB	4.0 %	2.7 %

Table 3. Accuracy (a) onset and (b) frame-wise detection with pink noise of given SNR added to the test data.

6. CONCLUSION

The presented system gives a new quality of recognition performance for onset detection, even under noisy conditions. However it should be mentioned that not all aspects can be compared directly. The same MIDI data, but different audio data generated with different synthesizers have been used. Also the time resolutions are different. However this does not invalidate the comparison in general.

The approach should be seamlessly applicable to similar instruments with properties similar to the piano, i.e. pronounced, percussive attack and tonal, highly structured sustain phase.

7. REFERENCES

[1] A.P. Klapuri, "Multiple fundamental frequency estimation based on harmonicity and spectral smoothness," *IEEE Trans. Speech Audio Process.*, vol. 11, no. 6, pp. 804–816, Nov. 2003.

[2] Masataka Goto, "A real-time music-scene-description system: predominant-F0 estimation for detecting melody and bass lines in real-world audio signals," *Speech Communication*, vol. 43, no. 4, pp. 311–329, 2004.

[3] G. E. Poliner and D. P. W. Ellis, "A discriminative model for polyphonic piano transcription," *EURASIP J. Appl. Signal Process.*, vol. 2007, no. 1, pp. 154–154, 2007.

[4] H. Thornburg, R. J. Leistikow, and J. Berger, "Melody Extraction and Musical Onset Detection via Probabilistic Models of Framewise STFT Peak Data," *IEEE Trans. Audio Speech Language Process.*, vol. 15, no. 4, pp. 1257–1272, May 2007.

[5] Simon Dixon, "Onset Detection Revisited," in *Proc. DAFx-06*, Sept. 18–20, 2006, pp. 133–137.

[6] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting," *Annals of Statistics*, vol. 28, pp. 337 – 407, 2000.

[7] R. Lienhart, A. Kuranov, and V. Pisarevsky, "Empirical Analysis of Detection Cascades of Boosted Classifiers for Rapid Object Detection," in *DAGM*, 2003, pp. 297–304.

[8] L.O. Endelt and A. la Cour-Harbo, "Wavelets for sparse representation of music," *Proc. WEDELMUSIC 2004*, pp. 10–14, September 2004.

[9] C. G. v. d. Boogaart and R. Lienhart, "Fast Gabor Transformation for Processing High Quality Audio," *ICASSP 2006, Toulouse, France*, vol. 3, pp. 161–164, 2006.

[10] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[11] K. Lee and M. Slaney, "Acoustic Chord Transcription and Key Extraction From Audio Using Key-Dependent HMMs Trained on Synthesized Audio," *IEEE Trans. Audio Speech Language Process.*, vol. 16, no. 2, pp. 291–301, Feb. 2008.

[12] Beth Logan, "Mel frequency cepstral coefficients for music modeling," in *ISMIR 2000, Proceedings*, Plymouth, Massachusetts, oct 2000.

[13] M. Goto, "A chorus section detection method for musical audio signals and its application to a music listening station," *IEEE Trans. Audio Speech Language Process.*, vol. 14, no. 5, pp. 1783–1794, Sept. 2006.

[14] T. Toivonen and M. Izumo, "TiMidity++ MIDI to WAVE converter," `http://timidity.sourceforge.net/index.html.en`, 2004.

[15] M. Ryynänen and A. Klapuri, "Polyphonic Music Transcription Using Note Event Modeling," in *IEEE WASPAA*, New Paltz, New York, USA, Oct. 2005, pp. 319–322.

[16] M. Marolt, "A connectionist approach to automatic transcription of polyphonic piano music," *Multimedia, IEEE Trans.*, vol. 6, no. 3, pp. 439–449, June 2004.