BASS LINE ESTIMATION FROM MUSIC

Matts Höglund
School of Computer Science and Communication
Royal Institute of Technology
Stockholm, Sweden
mattsho@kth.se

Robin Rajamäki
School of Electrical Engineering
Aalto University
Espoo, Finland
robin.rajamaki@aalto.fi

ABSTRACT

Fundamental frequency estimation of a single source from a mix of several other sources is a challenging task to say the least. Most pitch trackers work well on simple monophonic audio, but few are designed to handle complex polyphonic input like music.

In this paper we attempt to create an algorithm robust enough for estimating bass lines from music. The algorithm we propose is based on the weighted autocorrelation function (WACF) [6], with some added pre- and post-processing steps. A novel MIDI based evaluation framework with various quality metrics is also introduced. Finally, our algorithm is applied on a small database of musical audio snippets in order to assess the performance using our framework.

1. INTRODUCTION

1.1. Background and use cases

Audio to MIDI transcription is a current topic in the music technology world. For example, software such as Ableton Live[1] now supports audio to MIDI conversion of drum-loops and mono/polyphonic audio material.

One can easily imagine a range of similar uses for a bass line detector of the kind presented in this paper. As isolated bass line tracks (stems) are seldom available for other than commissioned remixers, it could be desirable to reliably extract bass lines from commercially available recordings (full mixes). The detected bass lines could then easily be transcribed into notation for bass players or producers, who don't have access to the original multi-track recordings.

1.2. Literature review

Pitch tracking of monophonic input signals is nowadays considered a solved problem within the research community [1]. Especially in the domain of speech processing, algorithms such as YIN [2] are reported to achieve detection rates of close to 100 % on some speech databases. Although some attempts at extracting melody and bass lines from more complex signals exist, they seem to be few in number and rarely operate on what is commonly understood as music. For instance, automatic transcription algorithms for polyphonic music, i.e. audio files generated from MIDI by re-synthesis, are described in [3] and [4]. One of the few who actually seem to have even attempted the task using real music is Masataka Goto, who reports detection rates of up to 92 % [5].

[1] http://cdn2-downloads.ableton.com/manuals/901/user_manual_en.pdf, accessed 24.7.2013

1.3. Problems

The single largest problem in estimating the fundamental frequency of a specific source from a full mix is the obvious masking in time and frequency of the other sources. Although the bass line can usually be assumed to have the lowest fundamental frequency in the mix, other issues remain. Since wavelength is inversely proportional to frequency, blockwise audio processing requires a fairly long analysis window in order to fit enough (in practice, more than one) wavelengths of a low frequency into a single signal block. As a result, the temporal resolution can be rather poor. Another issue is the low spectral resolution of fft-based methods at low frequencies. This can be countered to some extent through zero-padding, but effectively all of the spectral methods implemented through the course of this work failed to provide satisfying results. It is therefore the authors' opinion that time-domain methods are generally to be preferred to spectral methods when working in the low frequency region.

1.4. Paper overview

This paper is structured as follows: In the next section, the algorithm and its parts are presented in detail. After this, the used quality metrics, evaluation methods and results are discussed. In the concluding section, a rough plan for possible future work and improvements is sketched out, along with a summary of the main points and contributions of the work.

2. ALGORITHM OVERVIEW

The proposed algorithm can roughly be divided into three standard blocks, which will be inspected next in further detail. A block diagram of the algorithm is depicted in figure 1 and the parameters of each processing step are listed in table 1.

2.1. Pre-processing

In the pre-processing block, the entire input audio file is processed in three steps with the main aim of reducing the dimensionality of the data. This could also be viewed as trying to isolate the bass line in order to make things easier for the core pitch tracking algorithm.

Firstly, the signal is downmixed to mono, as it can be assumed that the bass line contains little relevant information in the stereo field. Secondly, the signal is low-pass filtered, while most of the bass line's energy can be expected to lie in the low frequency region. In this case, the signal is anti-causally filtered by an 8^{th} order zero-phase Butterworth IIR-filter to assure minimum lag and phase distortion, but to simultaneously provide a steep cut-off. In

the final stage the signal is downsampled, since it still has an unnecessary high sampling rate considering its band-limitation after the low-pass filtering. With a sampling frequency of about 4500 Hz the performance of the algorithm was noticeably speeded up, without significantly degrading the pitch tracker's frequency resolution.

2.2. Blockwise processing

The fundamental frequency estimation is done blockwise over the pre-processed audio signal. The core function here is the Average Magnitude Difference Function Weighted Autocorrelation Function (= AMDFWACF or, for short, WACF) [6]. For a signal block of length N, the function is formed by taking the product (ACF) and difference (AMDF) of the block ($x[n]$) with delayed copies of itself ($x[n+\tau]$), thus forming the autocorrelation function

$$ACF[\tau] = \frac{1}{N}\sum_{n=0}^{N-1} x[n]x[n+\tau], \tag{1}$$

and the average (squared) magnitude difference function

$$AMDF[\tau] = \frac{1}{N}\sum_{n=0}^{N-1} (x[n]-x[n+\tau])^2. \tag{2}$$

As the AMDF and ACF have minima respectively maxima at the lags where the signal is at its most periodic, dividing the two functions with one another pronounces the resulting function's peaks and valleys, making it slightly robuster against false pitch estimates. The estimates can be found from the WACF function

$$WACF[\tau] = \frac{ACF[\tau]}{1+AMDF[\tau]}, \tag{3}$$

by searching for the function's second highest peak after $\tau = 0$. As τ is a discrete sample index in this case, the estimated fundamental frequency (in Hz) is obtained by dividing the downsampled signal's sampling frequency ($f_s, d = f_s/M$) by τ (in samples)

$$\tilde{f}_0 = \frac{f_{s,d}}{\tau}. \tag{4}$$

Center clipping is also applied as a blockwise pre-processing step, since it pronounces the peaks of the WACF, increasing the likelihood of a correct pitch estimate [7].

2.3. Post-processing

The final processing of the estimated fundamental frequency vector ($\tilde{f}_0$ in figure 1) is done in the post-processing block, which consists of three main non-linear processing steps. The aim of these steps is to delete unwanted estimates and smoothen the final estimate vector ($\hat{f}_0$).

To start with, the estimate vector is thresholded with the blockwise obtained signal energy, which is first smoothed by a one-pole low-pass filter. It is assumed that all frames, whose energy is under a specific threshold, are silent. That is, no bassline is playing during these frames and the estimates are set to 0. An obvious drawback of the method is when the input audio signal contains a dominating one-time event in the low frequency region, which for a too high threshold can lead to the unwanted deletion of correctly estimated notes.

The penultimate post-processing step consist of a decision rule, where notes shorter than a specific length and outside the allowed value range are deleted. After this, the estimate vector is smoothed by a median filter.

2.4. Processing modes

Due to the very different SNR conditions of isolated bass stems and full mixes, it was deemed necessary to supply the proposed bass line estimation algorithm with two main modes of operation. The main difference between the modes lies in the "fastness" of the detection envelope. Basically, one of the processing modes (mode 2) trades temporal resolution and accuracy for added robustness and is consequently better suited for processing full mixes. It should be noted that note offsets are almost completely ignored in this mode and consequent notes of the same pitch are merged together. Although this may produce more false estimates (by ignoring silent gaps between notes), the errors are arguably perceived as less disturbing than for very fragmented error distributions. This fact also makes the (inevitable) manual MIDI editing less cumbersome, as one has fewer notes to delete, move around and change in length.

Table 1: *The algorithm's parameters (see footnotes for mode 1).*

Parameter	**Value**
Filter (type, order, cut-off [Hz])	Butterworth LP, 8, 300
Downsampling factor, M	$\lfloor \frac{f_s}{4410Hz} \rfloor$
Frame/Hop length [ms]	70/30
Center clipping	30 % of frame abs. max.
Energy threshold	25 % of global max.[a]
One-pole LP smoothing coeff., α	0.9 [b]
Search range [Hz]	30-300
Note min. length [ms]	50
Median filter length [ms]	175[c]

[a] 20 %
[b] 0.5
[c] 100

3. EVALUATION

A framework, based on MIDI files, was created to evaluate the performance of the algorithm described above. In this section the framework and the used metrics are described and the final results are presented and discussed.

3.1. Test set

The test set consists of ten short clips from songs of varied genre with available audio or MIDI stems. From these stems ground truth files were made by using the algorithm estimates and then correcting them by hand with a MIDI editor. The analyzed songs are listed in table 3.

3.2. Comparison of ground truth and estimate

In order to compare the ground truths with the estimated bass lines, the MIDI files are resampled. This simplifies the subsequent metrics calculations, as the data is transformed into two equidistantly sampled vectors of the same length. Following the resampling, some quantization error is introduced. This is discussed later in section 3.4.

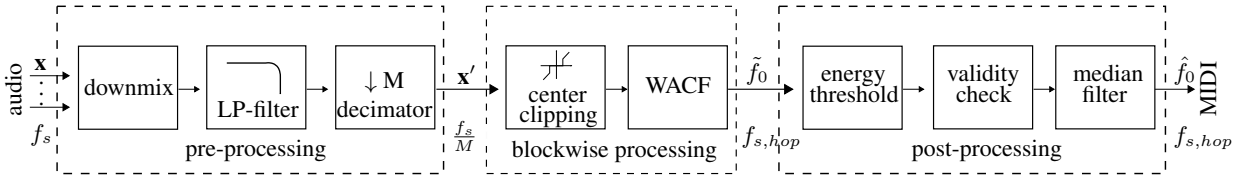

Figure 1: *Block diagram of the bass line estimation algorithm.*

3.3. Metrics

The metrics used in the evaluation are accuracy, precision, recall, octave error rate and onset error. Table 2 contains the description and unit of each metric. Since the detection algorithm allows 30 MIDI notes, multi-class classification metrics were used. The first three metrics are macro-averaged over the binary classification metrics of each class (in this case MIDI note)[8].

Table 2: *Evaluation metrics and descriptions*

Metric	Unit	Description
Average accuracy	%	Average fraction of correct classifications per note
Average precision	%	Average confidence in that the estimated tone is correct
Average recall	%	Average sensitivity for false estimates
Octave error rate	%	Fraction of octave errors in all errors ($\pm$ 2 octaves allowed)
Average onset error	ms	Average absolute onset error

3.4. Results and discussion

The performance of the algorithm on the full mix test set can be seen in table 3, along with the average performance of the available stems. One can see that the results are significantly better for the stems, as expected. In the case of precision and recall, the latter is generally better due to the algorithm's tendency to generate more false positives than false negatives. This is partly a consequence of ignoring pauses between notes (as discussed in section 2.4). The octave errors on the other hand, are mainly caused by inherent drawbacks of the WACF algorithm. When the true fundamental frequency has a lower amplitude than another harmonic component, the algorithm will output a false estimate.

The song with the best performance of the test set is 'Danny Rich - Fortune Teller', which is a drum & bass track with a simple (no staccato) and loud bass line. Therefore, the precision and recall are high and no harmonic errors occur. As a counterexample, the algorithm doesn't perform as well on the pop/rock song 'Coldplay - Clocks'. The base line is more complex, played in staccato and has more frequent note changes - hence the poorer performance. The estimated (incorrect and correct) and unidentified notes of these two songs are visualized in figure 2.

In addition to the issues already mentioned in the previous section, table 3 also confirms assumption about the algorithm's inherent low temporal resolution (see section 1.3). An average absolute onset error of 133 ms for full mixes and 77 ms for stems is noticeable and rather disturbing. The resampling of the MIDI files also introduces a small quantization error proportional to the resampling period. For the evaluation, a resampling period of 5 ms was used. This roughly corresponds to the time delay which is barely noticeable by the human hearing.

4. CONCLUSIONS

4.1. Future Work

Since the main problems of the proposed algorithm stem from inherent limitations of the WACF, in future work it would be sensible to do a thorough error analysis of the current implementation or to explore alternative time-domain pitch trackers. One could also try a completely different approach to the whole pitch estimation process, e.g. by using expectation maximization as proposed by Goto in [5]. For comparability, the results of any other implemented pitch trackers should then be evaluated using our framework and database.

There is still work to be done on the evaluation and database as well. Firstly, some perceptual modeling of errors could give a more truthful picture of how "correct" an estimated bass line actually sounds. Secondly, a large database of MIDI ground-truth files synced with the waveforms of musical clips (of varied genre) would be necessary to give realistic predictions of the performance of the algorithm a) generally or b) on a given style of music or tempo etc.

4.2. Summary

The central idea of this work was to explore how well a combination of relatively simple audio processing algorithms work on complex waveforms (= music) in a musical information retrieval (MIR) task such as bass line estimation.

For this task, a bass line detection algorithm was developed. The proposed algorithm uses the average magnitude difference function weighted autocorrelation function (WADF) at its core for pitch tracking, in addition to a host of pre-and post-processing steps such as low-pass filtering, downsampling, energy thresholding, note validity checking and median filtering. The final fundamental frequency estimates are then converted into MIDI.

In order to evaluate the performance of the designed algorithm an evaluation framework was developed. It was decided to base the evaluation on MIDI files (i.e. ground-truths and estimates), which were re-sampled in order to extract the following metrics: accuracy, precision, recall, octave error rate and onset error.

A test set of 10 commercially available songs from various genres were evaluated using these metrics. The results were as expected, with decent performance on tracks with strong and simple bass lines (think electronic dance music) and worse on others. In addition to the typical octave errors, the proposed algorithm also

Table 3: *Evaluation results of the full mix test set.*

	accuracy [%]	precision [%]	recall [%]	harmonic errors [%]	onset error [ms]
Danny Rich - Fortune Teller	62.5	93.4	94	0	43
Beyonce - Crazy In Love	50.1	70.7	74.8	0	70
Coldplay - Clocks	27	40.3	71.5	26.3	201
David Bowie - Ziggy Stardust	25.1	43.4	54.1	13.8	100
Depeche Mode - Personal Jesus	36.1	49.6	57.9	7.4	31
Lyra Project - Here & Now	61.6	83.3	89.8	0	170
Public Enemy - Get Up Stand Up	22.9	42.9	46.3	0	131
Captain Plan - Get You Some	36.6	36.3	64.7	0	49
She Makes Wars - Minefields	12.8	27.1	47.1	17.9	348
Alex Walker - Gotta Lose	11.4	29.1	34.9	3.8	189
Average	**34.6**	**51.6**	**63.5**	**6.9**	**133**
Average Stems	**60.9**	**76.8**	**77**	**3.5**	**77**

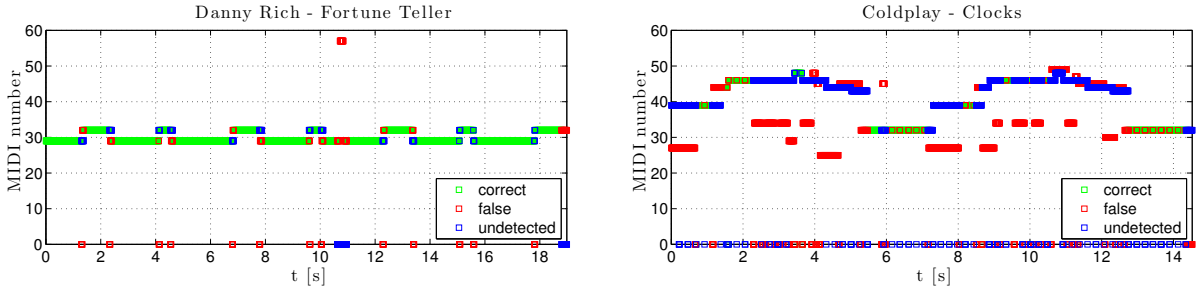

Figure 2: *Songs with false, correct and unidentified estimates.*

produces many false positive estimates as a trade-off for added robustness against perceptually more disturbing errors.

5. REFERENCES

[1] Patricio De La Cuadra, Aaron Master, and Craig Sapp, "Efficient pitch detection techniques for interactive music," in *Proceedings of the 2001 International Computer Music Conference*, 2001, pp. 403–406.

[2] Alain de Cheveigné and Hideki Kawahara, "YIN, a fundamental frequency estimator for speech and music," *Journal of The Acoustical Society of America*, vol. 111, pp. 1917–1930, 2002.

[3] M. Ryynänen and A. Klapuri, "Automatic bass line transcription from streaming polyphonic audio," in *Proc. 2007 IEEE International Conference on Acoustics, Speech, and Signal Processing*, Honolulu, Hawaii, USA, Apr. 2007, pp. 1437–1440.

[4] Stephen Hainsworth Malcolm and Malcolm D. Macleod, "Automatic bass line transcription from polyphonic music," in *In Proc. International Computer Music Conference, Havana*, 2001.

[5] M. Goto, "A robust predominant-f0 estimation method for real-time detection of melody and bass lines in cd recordings," in *Proceedings of the Acoustics, Speech, and Signal Processing, 2000. on IEEE International Conference - Volume 02*, Washington, DC, USA, 2000, ICASSP '00, pp. II757–II760, IEEE Computer Society.

[6] Hajime Kobayashi and Tetsuya Shimamura, "A weighted autocorrelation method for pitch extraction of noisy speech," in *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*. IEEE, 2000, vol. 3, pp. 1307–1310.

[7] Alexander Lerch, *An Introduction to Audio Content Analysis: Applications in Signal Processing and Music Informatics*, Wiley/IEEE Press, ISBN: 978-1-1182-6682-3, 2012.

[8] Marina Sokolova and Guy Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing & Management*, vol. 45, no. 4, pp. 427 – 437, 2009.