AUTOMATIC BASS LINE TRANSCRIPTION FOR ELECTRONIC MUSIC

R. Oğuz Araz
Music Technology Group
Universitat Pompeu Fabra
oguza97@gmail.com

ABSTRACT

Current bass line transcription systems either require a separate bass line track or attempt to transcribe the bass line directly from mixed multi track recordings. In the absence of separate bass line tracks, their performances are limited to the success of polyphonic transcription algorithms, which notoriously perform worse than their monophonic counterparts. In this work, we re-formulate the bass line transcription task and design a system that can transcribe and reconstruct bass lines for helping with the music production process. We use our domain knowledge on electronic music and develop a python library for performing monophonic chorus bass line transcriptions. Taking a polyphonic recording, our system can find a beat-synchronized chorus section, source-separate its bass line, perform transcription and output a bass line reconstruction MIDI file. The transcribed bass lines are locked tightly to the beat grid with an 1/32th (1/128th note in the common time) beat onset resolution, which can capture rich bass line grooves. Using this batch processing integrated python library, we test the performance on a collection of Tech House music tracks and evaluate the outputs by visually inspecting their spectrograms and listening to the bass line MIDI reconstructions.

1. INTRODUCTION

Fundamental frequency (F0) estimation systems yield frame-wise time-frequency pairs that carry the pitch information [1]. For applications such as F0 sonification or pitch track visualization, this frame-wise representation can be used directly. Whereas, for creating symbolic representations or MIDI files that can reconstruct the transcribed instruments - MIDI reconstructions, the outputs require further processing: they should be described in reference to the beat grid and be quantized both in time and in frequency. In addition, for tasks that require the transcription of only certain sections of a composition, for example the chorus, the pitch track should be segmented structurally. A more fundamental problem arises from the narrow frequency intervals between the pitches of the sub-bass frequency region, making it a hard task to transcribe the bass lines of electronic music for both humans and algorithms.

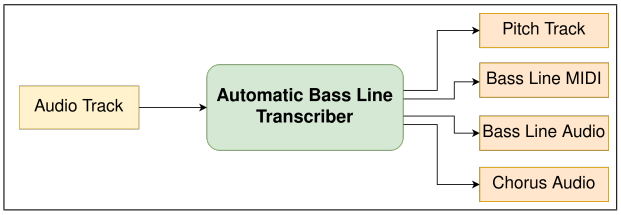

Figure 1. System Diagram

In order to deal with these problems, we design the automatic bass line transcriber (ABLT). ABLT transcribes the chorus bass line with estimating the beat grid, detecting chorus sections, separating the bass line from the rest of the instruments and using an F0 estimator with a custom quantization method. System inputs and outputs are displayed in Figure 1. The system can be accessed from its repository [1] .

2. SYSTEM COMPONENTS

ABLT consists of an extractor system that isolates the chorus bass line and a transcription system that is tailored for transcribing the frequency range between 32.7 and 130.81 Hz (from C1 to C3). We put such a frequency limit since this range carries the bass line fundamental frequencies and is mostly free from other instruments in electronic music tracks [2].

2.1 The Bass Line Extraction

Here we describe how we isolate the chorus bass line.

2.1.1 Beat Detector

In order to describe the onset and offset events in reference to the metrical structure, we start with estimating the beat positions. To this end, we use the state-of-the-art beat detection algorithm provided in the madmom library [3]. The estimated positions are used in estimating the BPM and forming the beat grid which also contains the divisions and the bars.

[1] https://github.com/raraz15/automatic_bass_line_transcriber

2.1.2 *Chorus Detector*

A bass line evolves throughout the composition, hence different sections can contain varying phrases. Therefore, we develop an algorithm that detects the main phrase's location in time. We use the chorus section since it is made up from the main musical phrases. We introduce an energy-based chorus detector that estimates where a chorus exists in the beat grid. We search for drops since it is common to many electronic music genres for a chorus to be marked by a drop, following immediately a breakdown section. We monitor the bass frequency energy movement throughout the beat grid to detect drops.

2.1.3 *Source Separator*

The chorus section contains multiple instruments, therefore we separate the bass line from the rest using the source separation model, Demucs [4]. After the bass line is separated, we use a low-pass filter to get rid of the unwanted artifacts. We cut-off at note C3 and normalize the filtered bass line.

2.2 The Bass Line Transcriber

In this section we describe the transcription process of the isolated chorus bass line that results in time-frequency and beat-note pairs.

2.2.1 *F0 Estimation*

We look for F0 estimators that can both resolve the smallest frequency interval of 1.95 Hz of the sub-bass pitches and the shortest possible duration of a 1/4th beat bass notes. Moreover, we require a 1/32th beat onset resolution to capture expressive grooves. Under these opposing constraints, we observed that spectrogram-based estimators fail. Therefore, we focus on waveform based estimators and choose the pYIN algorithm which searches for possible correlations in the waveform [1].

Our choice of pYIN is motivated by its computation speed and operation range of frequencies which includes the sub-bass frequency range. Moreover, comparing pYIN with the state-of-the-art CREPE F0 estimation network, we observed that pYIN performs better. This was expected due to the lack of sub-bass frequency examples in the CREPE training set [5]. Using the open-source librosa pYIN implementation, we search for hop and frame sizes to achieve the best time, frequency and onset resolution [6].

In order to estimate the F0 of a signal using autocorrelation methods, an estimation window of length equal to the maximum expected period is required, without exceeding the expected minimum period by large [7]. Therefore, the C1 pitch with the longest period sets the minimum possible window length of 30.58 ms. We look for window sizes that are multiples of this duration and compare their time and frequency resolutions. After experimenting with a number of window lengths, we choose a length equal to a single such period. Given the hard nature of the bass line transcription task, we consider the time resolution performance of this length adequate, which could also resolve the sub-bass frequency pitches reasonably well.

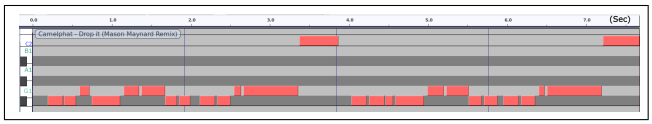

Figure 2. MIDI Reconstruction Example

We search for hop lengths that are proportional to the beat length, and obtain a beat quantized pitch track. After experimenting with various hop sizes, we observed that the most accurate transcriptions resulted from a hop length of 1/32th beats, which also captures rich groove information. We set the hop size to a 1/32th beat, and obtain a theoretical 1/32th beat onset resolution [8]. After applying a 5% confidence level threshold to the F0 estimate, which allows us to filter the parts where the algorithm was not sure of its predictions, we obtain the pitch track.

2.2.2 *Pitch Track Quantizer*

We draw disjoint epsilon balls (intervals) around the frequencies of the 440 Hz tuned scale. If a given frequency is inside any of these balls, we quantize it to that note and if not, we replace it with silence, indicated by 0 Hz. This approach helps in correcting the F0 estimation mistakes during silence regions, and solving the round-off errors during the logarithmic pitch conversion.

Due to their non-steady state nature, F0 estimation becomes challenging in the note onset and offset regions. Therefore, we develop a quantization algorithm to correct possible errors. We use the quarter-beat positions to segment the voiced regions of the bass line. Each voiced region starts with an onset, continues with quarter beats and ends with an offset. Using majority voting, the segments between two successive quarter-beats are quantized uniformly, independent from the rest. The onset segment is quantized as such, and compared with the majority vote of the following segment. If this comparison fails, we take the following segment's vote for the onset segment. The same procedure is applied to the offset segment using its preceding segment.

2.2.3 *MIDI Conversion*

The obtained note transcriptions from the quantization process are converted to MIDI notes. During the conversion process the velocity value of each MIDI note is taken as 120 to simplify the conversion. Using the outputted MIDI file, the chorus bass line can be instantly reconstructed using any DAW choice. An example of a reconstructed MIDI file is given in Figure 2.

3. FURTHER WORK

We plan to improve the chorus detector to use melodic information and integrate velocity estimation to the transcriber. As the F0 estimation and source separation algorithms are improved, they will be integrated.

4. ACKNOWLEDGEMENTS

This project was done as a Bachelor's senior design project at Koç University, Istanbul/Turkey. It could come to life with the helps of Utku Demir, who equipped me with the necessary programming skills and tools, and Furkan Yesiler who guided me during the course of the project and writing its paper.

5. REFERENCES

[1] M. Mauch and S. Dixon, "Pyin: A fundamental frequency estimator using probabilistic threshold distributions," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 659–663.

[2] J. Salamon, J. Serrà, and E. Gómez, "Tonal representations for music retrieval: From version identification to query-by-humming," *International Journal of Multimedia Information Retrieval, special issue on Hybrid Music Information Retrieval*, vol. 2, pp. 45–58, 03 2013.

[3] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, "madmom: a new Python Audio and Music Signal Processing Library," in *Proceedings of the 24th ACM International Conference on Multimedia*, Amsterdam, The Netherlands, 2016, pp. 1174–1178.

[4] A. Defossez, N. Usunier, L. Bottou, and F. Bach, "Music source separation in the waveform domain," 2020. [Online]. Available: https://openreview.net/forum?id=HJx7uJStPH

[5] J. W. Kim, J. Salamon, P. Q. Li, and J. P. Bello, "Crepe: A convolutional representation for pitch estimation," *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 161–165, 2018.

[6] B. McFee, A. Metsai, M. McVicar, S. Balke, C. Thomé, C. Raffel, F. Zalkow, A. Malek, Dana, K. Lee, O. Nieto, D. Ellis, J. Mason, E. Battenberg, S. Seyfarth, R. Yamamoto, viktorandreevichmorozov, K. Choi, J. Moore, R. Bittner, S. Hidaka, Z. Wei, nullmightybofo, D. Hereñú, F.-R. Stöter, P. Friesch, A. Weiss, M. Vollrath, T. Kim, and Thassilo, "librosa/librosa: 0.8.1rc2," May 2021. [Online]. Available: https://doi.org/10.5281/zenodo.4792298

[7] R. L. Rabiner and R. W. Schafer, *Theory and Applications of Digital Speech Processing*. Upper Saddle River, NJ 07458: Pearson, 2011.

[8] E. Scheirer and B. Vercoe, "Extracting expressive performance information from recorded music," Master's thesis, Massachusetts Institute of Technology, Cambridge, United States, 1999.