MIDI MINER – A PYTHON LIBRARY FOR TONAL TENSION AND TRACK CLASSIFICATION

Rui Guo[1] **Dorien Herremans**[2] **Thor Magnusson**[1]
[1] University of Sussex, United Kingdom
[2] Singapore University of Technology and Design, Singapore
r.guo@sussex.ac.uk

EXTENDED ABSTRACT

We present a Python library, called Midi Miner, that can calculate tonal tension and classify different tracks. MIDI (Music Instrument Digital Interface) is a hardware and software standard for communicating musical events between digital music devices. It is often used for tasks such as music representation, communication between devices, and even music generation [5]. Tension is an essential element of the music listening experience, which can come from a number of musical features including timbre, loudness and harmony [3]. Midi Miner provides a Python implementation for the tonal tension model based on the spiral array [1] as presented by Herremans and Chew [4]. Midi Miner also performs key estimation and includes a track classifier that can disentangle melody, bass, and harmony tracks. Even though tracks are often separated in MIDI files, the musical function of each track is not always clear. The track classifier keeps the identified tracks and discards messy tracks, which can enable further analysis and training tasks.

The spiral array [1] is a three-dimensional mathematical model of tonality developed by Prof. Elaine Chew. It consists of a helix for pitch classes, chords, and keys. It is important to note that this model takes pitch spelling into account, meaning that Ab and G# will have a different position in the pitch class helix. Since MIDI files do not capture pitch spelling, we include an algorithm for estimating pitch spelling. First, the key index of the MIDI file is set as the most frequently occurring pitch class. The mode of the key (major or minor) cannot be differentiated at this stage, but the key index can inform the pitch spelling. Based on the estimated key index, the piece will fall into one of two classes, the first is based on the key indices C, D, E, G, A, and B, and the second on Db, Eb, F, Gb, Ab ad Bb. The accidentals of pieces in the first class will be mapped to sharps if needed, whereas the accidentals of pieces in the second class will be mapped to flats. After the pitch spelling has been estimated for all note, the final key can be determined using the spiral array key detection algorithm [2].

Tonal tension [4] define three measures of tonal tension: cloud diameter (dissonance), cloud momentu (tonal movement), and tensile strain (distance to the key). Mini Miner implements these three tension measures based on a default window length of one half note (customizable by the user). The implemented model works for polyphonic music, by weighing the occurrence of each pitch class per time window. After inputting a MIDI file, Midi Miner also provides a graphical representation of the three tension measures. Figure 1 provides an example output for the tensile strain measure (i.e., distance to the key). A sliding window of 16 beats with average tensile strain is used to detect potential key changes by comparing the ratio between the mean tensile strain and the adjacent window. If this $ratio > 2$ for four consecutive windows, then a key change is flagged. In Figure 1, Midi Miner detects a key change in bar 61.

Midi track classifier Midi Miner also implements a track classifier that untangles melody, bass, and harmony tracks. These tracks are of interest for anybody preparing a dataset for, e.g., music generation. A random forest classifier was trained by using 5,006 Chinese pop songs collected from the Internet. Among those files, 3,265 files have at least one track label. These MIDI track labels are strings marked by the author, which usually provide information about the track type. By looking at the labels in those MIDI files, we manually added specific labels for melody, bass, chords, which were used to train our classifier. The drum tracks are

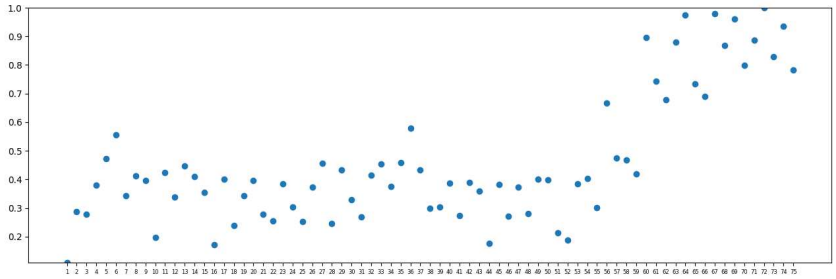

Figure 1. Tensile strain for each bar of the MIDI file available online[1].

currently discarded. Similar to [6], who select 34 features to classify the melody track, we computed 30 features for each track. These are used as input to train our classifier and are described in detail on our github page. The performance of the track classifier is evaluated in Table 1 using a test set (25% or the original dataset). Based on the classification result, Midi Miner will extract the melody, bass and harmony tracks and output a new MIDI file with only those tracks.

track	prediction	precision	recall	F1 score
melody	False	0.99	0.99	0.99
	True	0.95	0.93	0.94
bass	False	0.98	1.00	0.99
	True	0.97	0.92	0.95
harmony	False	0.98	0.98	0.98
	True	0.92	0.89	0.91

Table 1. Track classification result

Download Midi Miner The Midi Miner library is available online[2]. In the future, we aim to include a more comprehensive extraction of MIDI features, as well as more complete music analysis tools.

ACKNOWLEDGMENTS

This work has received funding from the Chinese scholarship council.

REFERENCES

[1] E. Chew. Mathematical and computational modeling of tonality. *AMC*, 10:12, 2014.

[2] C.-H. Chuan and E. Chew. Polyphonic audio key finding using the spiral array ceg algorithm. In *2005 IEEE International Conference on Multimedia and Expo*, pages 21–24. IEEE, 2005.

[3] M.M. Farbood. A parametric, temporal model of musical tension. *Music Perception*, 29(4):387–428, 2012.

[4] D. Herremans and E. Chew. Tension ribbons: Quantifying and visualising tonal tension. In *Second International Conference on Technologies for Music Notation and Representation (TENOR)*, volume 2, pages 8–18, Cambridge, UK, 05/2016 2016.

[5] D. Herremans, C.-H. Chuan, and E. Chew. A functional taxonomy of music generation systems. *ACM Computing Surveys (CSUR)*, 50(5):69, 2017.

[6] David Rizo, Pedro J Ponce De Leon, Antonio Pertusa, Carlos Pérez-Sancho, and José Manuel Iñesta Quereda. Melody track identification in music symbolic files. In *FLAIRS Conference*, pages 254–259, 2006.

[2] `https://github.com/ruiguo-bio/midi-miner`