Pypianoroll: Open Source Python Package for Handling Multitrack Pianoroll

Hao-Wen Dong, Wen-Yi Hsiao and Yi-Hsuan Yang
Research Center for IT Innovation, Academia Sinica, Taipei, Taiwan
[Documentation] `https://salu133445.github.io/pypianoroll/`

>> Core Classes

- use ***symbolic timing***

→ each beat has the same length (`beat_resolution`)

→ note length can represent a *musically-meaningful* amount of time (such as a 4th or 8th note)

- save tempo information in the `tempo` array

Attributes of a `Multitrack` object

Attribute	Description
tracks	List of `Track` objects
beat_resolution	Resolution of a beat (in time step)
tempo	Array that records the tempo value (in bpm) at each time step
downbeat	Array that indicates the locations of downbeats (the first beat of a bar)
name	Name of the multitrack

Attributes of a `Track` object

Attribute	Description
pianoroll	Pianoroll matrix
program	Program number according to General MIDI Level 1 specification
is_drum	Whether it is a percussion track
name	Name of the track

>> Manipulation Utilities

track level

- `append_track`
- `merge_tracks`
- `remove_tracks`
- `remove_empty_tracks`
- `get_merged_pianoroll`
- `get_stacked_pianoroll`

pianoroll level

- `clip`
- `binarize`
- `transpose`
- `pad_to_multiple`
- `assign_constant`
- `trim_trailing_silence`

>> Evaluation Metrics

- `empty_bar_rate`
- `qualified_note_rate`
- `drum_in_pattern_rate`
- `polyphonic_rate`
- `n_pitches_used`
- `n_pitch_classes_used`
- `in_scale_rate`
- `tonal_distance` [2]

*(designed for evaluating **generative system** [1])*

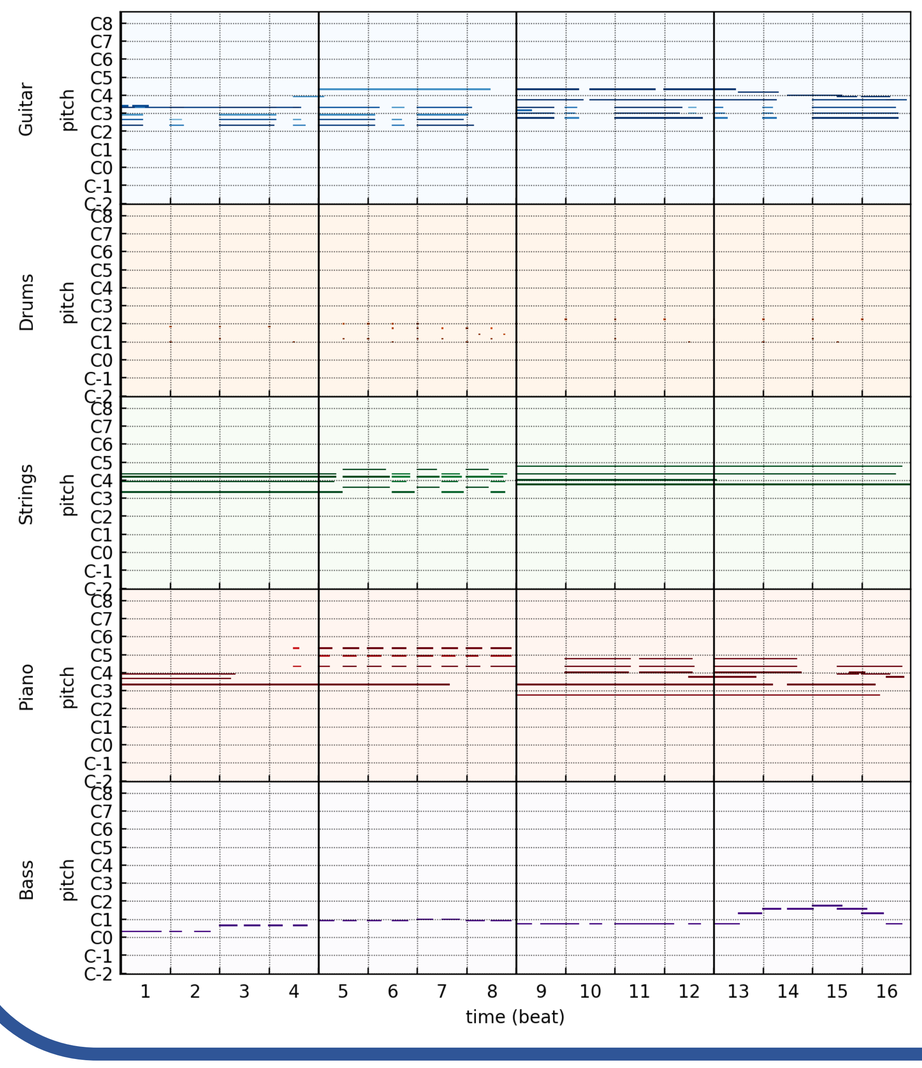
>> Content Analysis Utilities

- key detection
- melody recognition
- chord recognition
- chord-related feature extraction

(future plan)

(may contribute to applications like lead sheet arrangement [3])

>> Visualization



>> Data I/O

(use compressed column storage to save space)

Multitrack object

`pypianoroll.save()` / `pypianoroll.load()` ⇄ **npz file** (.npz)

`pypianoroll.write()` / `pypianoroll.parse()` ⇄ **MIDI file** (.mid)

(use `pretty_midi` [4] for MIDI I/O)

(plan to support MusicXML format in the future)

>>References

[1] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang. MuseGAN: Symbolic-domain music generation and accompaniment with multi-track sequential generative adversarial networks. In *Proc. AAAI*, 2018.

[2] C. Harte, M. Sandler, and M. Gasser. Detecting harmonic change in musical audio. In *Proc. ACM Workshop on Audio and Music Computing Multimedia*, 2006.

[3] H.-M. Liu and Y.-H. Yang. Lead sheet generation and arrangement by conditional generative adversarial network. In *Proc. ICMLA*, 2018.

[4] C. Raffel and D. P. W. Ellis. Intuitive analysis, creation and manipulation of MIDI data with pretty_midi. In *ISMIR Late Breaking and Demo Papers*, 2014.