PYPIANOROLL: OPEN SOURCE PYTHON PACKAGE FOR HANDLING MULTITRACK PIANOROLLS

Hao-Wen Dong, Wen-Yi Hsiao and Yi-Hsuan Yang
Research Center for IT Innovation, Academia Sinica, Taipei, Taiwan
{salu133445,wayne391,yang}@citi.sinica.edu.tw

ABSTRACT

The pianoroll representation represents music data as time-pitch matrices. Although it has been used widely as a way to visualize music data, it has not been much used in computational modeling of music. To promote its usage, we present in this paper a new, open source Python package called *Pypianoroll* for handling multitrack pianorolls. The core element of *Pypianoroll* is a new, lightweight multitrack pianoroll format that contains additional tempo and program information along with the pianoroll matrices. It provides tools for creating, manipulating, storing, analyzing and visualizing multitrack pianorolls in an intuitive way. Code and documentation are available at `https://github.com/salu133445/pypianoroll`.

1. INTRODUCTION

A *pianoroll* is a symbolic music representation that records the presence of pitches at each time step as a binary time-pitch matrix, as exemplified in Figure 1. To handle multitrack music, we can similarly define a *multitrack pianoroll* as a list of pianorolls, each corresponding to one specific track (or instrument). There are many possible formats to represent the multitrack pianoroll. In this paper, we propose a new format that provides information regarding the program and tempo of the tracks. We then present *Pypianoroll*, an open source package released under the MIT License that provides the following functionalities:

- Core classes that store symbolic music with the proposed format (Section 2)
- Manipulation utilities (Section 3)
- Efficient and space-saving I/O (Section 4.1), and conversion utilities to/from MIDI files (Section 4.2)
- Utilities for visualization (Section 5.1), evaluation (Section 5.2), and content analysis (Section 5.3).

2. MULTITRACK PIANOROLL FORMAT

The proposed format stores the program and tempo information alongside the pianoroll matrices. For each track, we use an integer to specify its program number in a MIDI-like way, and, following [3], a boolean number to indicate whether it is a percussion track. Oftentimes, the program numbers indicate the instruments used during playback.

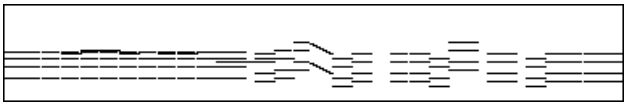

Figure 1: Example pianoroll, where the vertical and horizontal axes represent pitch and time, respectively.

As for the tempo information, many pianoroll formats, such as the one implemented in pretty_midi [3], use the *absolute timing*, where the actual timing (in second) of the note onsets and offsets are used for the time axis. But, this introduces performance-level attributes to the pianorolls.

In order to decouple music composition from music performance, we opt for the ***symbolic timing*** *and make the time axis invariant to the tempo.* We store the tempo value (in bpm) at each time step as an additional array and remove the tempo information from the pianoroll matrices. As a result, each beat has the same length regardless of the tempo and we store the number of time steps used to represent a beat, i.e., the *beat resolution*, as metadata. In this way, the length of a note can now represent a musically-meaningful amount of time, e.g., a 4th or 8th note. [1]

In *Pypianoroll*, we implement the `Multitrack` class as a base class for multitrack pianorolls using the proposed format. As shown in Tables 1 and 2, a `Multitrack` object consists of a list of `Track` objects, each of which is composed of a pianoroll matrix, a program number, a boolean drum indicator and its name. A `Multitrack` object also contains a beat resolution (used for all the pianorolls), a tempo array, a downbeat array (that indicates the locations of downbeats) and its name. Note that other extended classes can be implemented on top of this base class. For example, we are designing an extended class that makes onset/offset explicit to deal with repeating notes.

3. MANIPULATION UTILITIES

We provide in *Pypianoroll* functions for manipulating multitrack pianorolls in different levels. For more information, we refer the readers to the online documentation.

[1] Note that by setting beat resolution to the sampling rate, pianorolls in absolute timing are still compatible to *Pypianoroll* with potentially unexpected behaviors for functions that rely heavily on the beat resolution.

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

Table 1: Attributes of a `Multitrack` object.

Attribute	Description
pianoroll	Pianoroll matrix
program	Program number according to General MIDI Level 1 specification [2]
is_drum	Whether it is a percussion track
name	Name of the track

Table 2: Attributes of a `Track` object

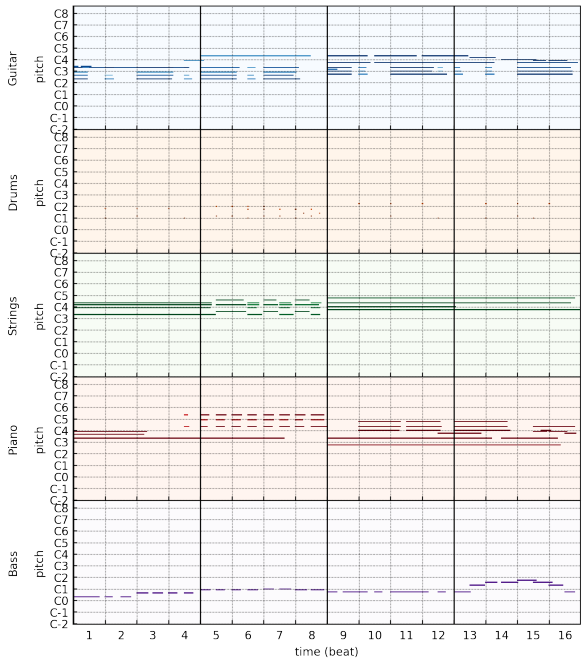

Figure 2: Example visualization of a five-track pianoroll with *Pypianoroll*. The lightness represents the velocities. [3]

4. DATA I/O

4.1 Save & Load

The pianoroll representation may be space-inefficient due to the large number of zeros (i.e., it is sparse). To address this, we implement space-saving I/O by storing the pianorolls with compressed column storage (CCS), leading to a lossless compression rate of about 100:1 on average.

4.2 Conversion Utilities

We provide two-way conversion utilities between MIDI files and multitrack pianorolls. We parse and write MIDI files using pretty_midi [3] and implement the conversions between absolute and symbolic timings, and between event- and matrix-based representations. To further increase the data source, we plan to implement conversion utilities for other formats, e.g., MusicXML, in the future.

5. OTHER UTILITIES

5.1 Visualization Utilities

Appropriate visualizations of multitrack pianorolls can be useful for both analysis and demonstration purposes. Hence, we provide in *Pypianoroll* convenient visualization utilities for multitrack pianorolls. It comes with several style presets while retaining high flexibility. We show in Figure 1 an example visualization of a five-track pianoroll.

5.2 Evaluation-related Metrics

We implement in *Pypianoroll* the evaluation metrics proposed in [1] along with several new metrics, including *empty_bar_rate*, *qualified_note_rate*, *drum_in_pattern_rate*, *n_pitches_used*, *n_pitch_classes_used*, *polyphonic_rate*, *in_scale_rate* and *tonal_distance*. These metrics can be useful for evaluating automatic music generation systems that take multitrack pianorolls as the target outputs.

5.3 Content Analysis Utilities

We plan to provide content analysis utilities in the future. This may include key detection, chord recognition, chord-related feature extraction and melody recognition, all of which are done in the symbolic domain (using pianorolls) not in the audio domain. Such utilities may contribute to applications such as *lead sheet arrangement* [2].

6. CONCLUSIONS

We have proposed in this paper an open source Python package for handling multitrack pianorolls. We hope it can facilitate the use of pianoroll formats in research topics such as music generation and transcription.

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

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

[2] `https://www.midi.org/specifications/item/gm-level-1-sound-set`

[3] Although a pianoroll is usually considered to be binary-valued, we can extend it to be real-valued by treating the values as note velocities. In *Pypianoroll*, the pianoroll matrices can be either binary- or real-valued.