Bassline Generation from Audio

Kelian Li *Alexander Lerch*

Center for Music Technology, Georgia Institute of Technology

***Abstract*—This paper presents a system that automatically generates basslines from audio signals. The proposed rule-based method uses the extracted audio and music features. The generated basslines in MIDI representation fit the original audio in terms of the pitch and the rhythm. A graphic user interface (GUI) is also developed which allows some high-level control over the final output.**

Keywords—generative music, rule-based systems

I. Introduction

In the age of democratized music production, the so called "bedroom producers" no longer need to receive formalized music training, music theory and song arranging to be part of the creative community and create professional music. Instead, they rely on assisted compositional tools such as MIDI packs, sequencers and MIDI effects. Besides MIDI, audio loops of single instrument and pre-made backing tracks, commonly known as beats, are very popular among amateur and professional arrangers and songwriters. While it is arguable ~~that~~ whether these compositional tools enhance or hurt the originality in popular music in general, many producers with limited music knowledge can benefit from these tools.

The more advanced MIDI effects, sometimes referred as music theory tools, either generate MIDI from the input from users (e.g. choosing chord progressions) or take the existing MIDI data as the input. However, the input mechanisms make both of the two methods limited in some aspects. The user interface of music generation systems can easily become too complicated to navigate and may not fit the need of users with different levels of music knowledge. Generating music from the existing MIDI data appears to be a practical approach, but it faces some challenges in real-world scenarios. Either MIDI files or MIDI tracks in digital audio workstations are guaranteed to have proper instrumentation labels. Furthermore, the wide variety of synthesized sounds can not represented as a set of finite MIDI program change numbers. Thus, MIDI may not provide either sufficient nor accurate information about the musical content. In fact, many music theory tools only take a single MIDI track as the input. In this case, they can perform tasks like generating arpeggios or creating melodies from the analyzed chords.

Although MIDI plays an important role in music production, a complete song often consists of both recorded tracks and virtual instruments. Audio samples, loops, and backing tracks are also widely used in music production. Hence, audio signal should not be neglected in building assisted compositional tools, especially in the popular music domain. Since MIDI tracks can be converted into audio, and they are supposed to be converted into audio eventually, it is guaranteed that the audio signal will contain all the musical content.

The objective of this paper was to develop a controllable system that would generate bassline from the provided audio. As explained above, without the dependency of any symbolic musical representation, this system can be applied to more general use cases. The audio is expected to be in the category of popular music, and the system relies on the its harmonic and rhythmic content. The companion GUI allows users to control some aspects of the music generation without the requirement of music training.

II. Background

Composing music has involved a series of activities, such as the defining the melody, the rhythm and the harmony, writing counterpoint or voice-leading, arrangement or orchestration. Algorithmic composition is defined as "using some formal process to make music with minimal human intervention" [1]. Probably the earliest example of algorithmic composition is the Musikalisches Wüurfelspiel (Dice Music) attributed to Wolfgang Amadeus Mozart which uses randomness to determine musical elements. With the advent of computers, composers have the opportunity to develop more sophisticated algorithms. Early algorithmic composition systems are based on pre-defined rules, stochastic, or the combination of both [2]. More recent algorithmic composition methods diverge from ruled-based systems and use data-driven machine learning techniques. The proposed methods were mostly based on Markov chains, genetic algorithms while artificial neural networks were adapted into algorithmic composition most recently [3].

The computer aided algorithmic composition (CAAC) system, a sub-field of algorithmic composition, is defined as "software that facilitates the generation of new music by means other than the manipulation of a direct music representation" [4]. In other words, the output must not be a "copy" of the input after direct manipulations by users. This broad definition includes some rather simple systems such as MIDI arpeggiators.

III. Related Work

Music generation from from audio is a rarely studied topic in the research community. Hung et al. [6] proposed a method based on encoder-decoder architecture that can generate new score rearrangement from audio by modifying the timbre representation. Lei et al. [7] used variational autoencoders to build a conditional drum pattern generation model. This model can generate symbolic drum pattern was generated given the accompaniment melodic tracks.

The only research related to bass score generation from audio is proposed by Haki and Sergi [8] in 2019. Their method uses word-based sequence-to-sequence learning, a natural language processing technique, to generate bass music score from the given drum audio in two dance music styles. The basslines are stylistically coherent and rhythmically interlocked with the drum loop. The input audio is not transcribed into symbolic representations for training. Instead, the quantized onsets detected from a 2-bar drum loop in eight frequency bands are the input in this model.

Although the research of bassline generation in academia is not many, several commercial products are devoted to address this need. Pitch correction software like Melodyne sometimes offers chord detection and MIDI export features that allow users to get the root notes of the chords after some edits [9]. Studio One takes it one step further by directly generating the bass track that consists of the root notes [10]. Another noticeable product is the Band Creator guitar pedal by DigiTech [11]. This pedal can generate bass and drum parts that matches the chords and rhythm of guitar playing. In collaboration with Band-in-a-Box, a music accompaniment generation software, Band Creator provides 12 music genres and 12 styles of each genre for users to select from.

IV. Method

A. Rule-based System

The system proposed in this paper is based on the common practice of bass guitar writing in popular music. One important role of bass is to establish the chord progression. In most popular music genres, the root note of a chord is one of the most common notes for the bassists to play. Playing the third and the fifth notes in tertian chords (triads and extended chords) forms inversions. Choosing chord inversions instead of the root position can be a good or bad choice depending on the context. Therefore, inversions are not incorporated in the system. Passing note is a non-chord tone that connects two chord tones by moving in the same direction stepwise. In bassline writing, passing notes are commonly used to connect two chords and add movements to the music. There are more advanced pitch writing techniques for bass, but they are not implemented due to the scope of this work [12].

As an accompaniment instrument, basses are expected to interlock with the drums. It is a common technique for the bass to sync in time with the kick drum, and the purpose is to accentuate the rhythmic punch. The bass part can also sync with the rhythm of other accompaniment instruments and avoid clashing with the main melody [12].

The dynamics of the bass playing usually aligns with the relative intensity of the music. The bass plays soft in quiet parts, and it plays loud in loud parts.

B. Feature Extraction

This system is written in Python, and relies on music information retrieval (MIR) libraries for feature extractions. The extracted audio features are the source of music generation.

- Chord recognition of the audio signal provides the corresponding root note pitches and the starting and ending positions of each chord. The bassline then plays the root notes in the proper instrumental range. The chosen chord recognition algorithm was developed by Korzeniowski and Gerhard [13], and implemented by Böck et al. [14] in Madmom python library.
- Key detection extracts the ~~one most possible~~ major or minor scales of the audio signal. This scale is used to determine the pitch of the passing tones. The integrated key recognition algorithm was developed by Korzeniowski and Gerhard [15] and implemented in Madmom library [14].
- Onset detection finds the starting points of all musically relevant events in the audio. The rhythm of the bassline comes from the onsets directly. The threshold of peak-picking determines the selection of the final onsets. The lower the threshold is, the more the selected onsets will be. In the case that the input signal is the mixed tracks of popular music, the the onsets of percussive instruments (kick drums, snare drums, etc.) are more likely to be picked. The chosen onset detection algorithm was proposed by Böck and Gerhard [16] and implemented in Madmom library [14].
- Beat detection outputs the possible beat positions of the music. The use of beat detection is to calculate note density which will be discussed in the implementation section. The beat detection algorithm was proposed and implemented by Brossier [17] and implemented in Aubio library [17].
- Boundary detection in structure segmentation provides the possible boundary positions of music sections. Obtaining segmented sections assists the controllable bassline generation in each music section. The structure segmentation algorithm was proposed by Serra, Joan, et al [18] and implemented implemented in MSAF library [19] [20].
- Loudness and note density are both positively correlated with arousal [21] [22]. The relative differences of K-weighted integrated loudness between the entire audio and each section are used to provide a suggested note density changes between music sections. The loudness is calculated by pyloudnorm library [23].

C. Implementation

Section boundaries and beat positions are obtained from structure identification and beat detection. In each section of the music, the total number of beats is calculated. The next step is to find the threshold value in the onset detection function that makes the note density value close enough to the target value. The note density in this system is defined as follows:

$$Note\ Density = \frac{number\ of\ notes}{number\ of\ beats}$$

The total amount of bass notes are the same as the total amount of onsets. Because the number of selected onsets is negatively correlated with the threshold value, an algorithm is then implemented to find this threshold from a given note density value. This algorithm iterates over the onset peak-picking function and changes the threshold from the error between the target note density and the current note density until the the error is small enough.

The initial note density values of music sections are the same default values before being altered from the relative loudness. The differences between the loudness of each section and the loudness of the full-length audio are reflected linearly on the positional values of the sliders in the GUI. These values are mapped to the note density values in each section. Using these values, the algorithm described above can then outputs a list of onsets which are the note-on positions of the bass. The duration of each note is set to 80% of the length from the starting point of the current note to the next note.

The default pitches of the bass notes are the root notes obtained from chord recognition. After setting the root notes to the bass part, passing notes are added by altering the current pitches. An algorithm scans through each three adjacent notes in the pitch list. If the first two notes have the the same pitch and the interval between the second and the third notes is no greater than two whole steps, the second note will become a passing note. The algorithm searches and chooses a pitch between the first and the third notes from the key scale obtained in the key detection.

MIDI velocities are linearly mapped to the slider values. A random number is added to the velocity in order to create some variation.

Mido lilbrary is used to generate MIDI files [24].

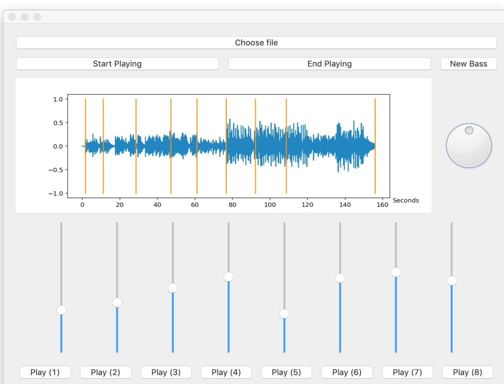

Figure 1: The GUI for the bassline generation control

D. GUI

The GUI is developed on Pyside library [25]. Users can load audio files, and the application will generate the bassline in MIDI file. The application shows the waveform of the audio and the section boundaries using Matplotlib library [26]. The audio and MIDI playback is supported by Pyaudio library and Pygame library [27] [28]. The "Start Playing" button allows the full-length preview of the generated bassline mixed the original audio. By clicking the play buttons under the sliders, users can listen to each individual section. The knob next to the waveform is the volume control of the original audio.

The sliders control the note density and the note velocity in each section. After loading the audio file, the application automatically sets the slider positions as shown in figure 1. After changing any of the sliders, the user need to click on “New Bass” button fo the application to generate new MIDI file. The default range of the slider from 0 to 100 is mapped to the note density range from 0.125 to 2.5. The mapping of the middle part of the slider has a lower linear slope, giving users the ability to fine-tune the note density in the common range where basslines tend to be in.

V. Discussion

Evaluated by the author, this system can provide a workable bassline from the suggested default parameters. After altering the sliders, the results can be improved significantly in some situations. Some drawbacks of the current implementation limit the results.

Because chord recognition does not take account of the previous information and the musical context, chord recognition alone is not sufficient for bassline generation. Human musicians can predict the timing and the chord type without hearing every note of the chord. The results of chord recognition is often delayed, and can fail when the chord notes are not fully presented.

Onset detection performs quite well in many cases. However, using onsets naively is not ideal for the purpose of bassline generation. The snare drums are more likely to be selected than the kick drums when the peak-picking threshold is high. However, the bass part tends to follow the kick drums more than the snare drums.

The current way of adding passing notes can only work in the scenario where there is no modulation and the chords are all diatonic. To generalized the passing note method, the modal scale of the nearby chords may be considered.

VI. Future Work

The sliders are at the same default positions before being altered by the loudness differences. By integrating genre classification, the initial positions can be set to the average note density of the detected genre, and the initial generated bassline should fit better with the style of the music. The exact average note density values can be extracted from large MIDI datasets or audio signals.

Source separation can possibly improve the accuracy of the onset detection and alter its focus from snare drums to kick drums. Automatic drum transcription is another method that may improve the rhythmic aspect.

Because the bass part sometimes plays the same line as the lowest line in other instruments, extracting the bass pitch of the input audio provides an alternative choice other than the chord recognition.

VII. Conclusion

A bassline generation system is proposed by utilizing extracting audio and music features from the audio signals. In real-world scenarios, computer aided algorithmic composition systems using symbolic music representation inputs face challenges of the incomplete and inaccurate data. The proposed rule-based method attempts to fill this gap by generating basslines directly from audio. A GUI is developed for users to control some aspects of the generated bassline.

REFERENCES

[1] A. Alpern. Techniques for algorithmic composition of music. 95:120, 1995.

[2] J. A. Maurer. The history of algorithmic composition, 1999.

[3] Briot, Jean-Pierre, Gaëtan Hadjeres, and François-David Pachet. "Deep learning techniques for music generation--a survey." arXiv preprint arXiv:1709.01620, 2017.

[4] Ariza, Christopher. "Navigating the landscape of computer aided algorithmic composition systems: A definition, seven descriptors, and a lexicon of systems and research." Proceedings of the International Computer Music Conference (ICMC), 2005.

[5] Ji, Shulei, Jing Luo, and Xinyu Yang. "A Comprehensive Survey on Deep Music Generation: Multi-level Representations, Algorithms, Evaluations, and Future Directions." arXiv preprint arXiv:2011.06801, 2020.

[6] Yun-Ning Hung, I Chiang, Yi-An Chen, Yi-Hsuan Yang, et al. "Musical composition style transfer via disentangled timbre representations." Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 2019.

[7] I-Chieh Wei, Chih-Wei Wu, and Li Su. "Generating structured drum pattern using variational autoencoder and selfsimilarity matrix." Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), 2019.

[8] Haki, Behzad, and Sergi Jorda. "A Bassline Generation System Based on Sequence-to-Sequence Learning." Proceedings of the International Conference on New Interfaces for Musical Expression (NIME), 2019.

[9] "Melodyne 5: Chord recognition and the Chord Track". [Online]. Available: https://helpcenter.celemony.com/M5/doc/melodyneStudio5/en/M5tour_ChordTrackAndDetection?env=standAlone. [Accessed: 28-Apr-2021].

[10] "Chord Track". [Online]. Available: https://s1manual.presonus.com/Content/Arranging_Topics/Chord Track.htm. [Accessed: 28-Apr-2021].

[11] "Band Creator Looper". [Online]. Available: https://www.digitech.com/band-creator/TRIOPLUS.html. [Accessed: 28-Apr-2021].

[12] Drew Swisher. "How to Write a Bassline: 5 Steps for Songwriting Success" Musician on a Mission, 17-S 0. [Online]. Available: https://www.musiciononamission.com/h write-a-bassline/.

[13] Korzeniowski, Filip, and Gerhard Widmer. "Feature learning for chord recognition: The deep chroma extractor." Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), 2016.

[14] Böck, Sebastian, et al. "Madmom: A new python audio and music signal processing library." Proceedings of the ACM international conference on Multimedia, 2016.

[15] Korzeniowski, Filip, and Gerhard Widmer. "Genre-agnostic key classification with convolutional neural networks.", Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), 2018.

[16] Böck, Sebastian, and Gerhard Widmer. "Maximum filter vibrato suppression for onset detection." Proceedings of the 16th International Conference on Digital Audio Effects (DAFx), 2013.

[17] Paul Brossier, Automatic annotation of musical audio for interactive systems, PhD thesis, Centre for Digital music, Queen Mary University of London, 2006.

[18] Serra, Joan, et al. "Unsupervised music structure annotation by time series structure features and segment similarity." IEEE Transactions on Multimedia 16.5 (2014): 1229-1240.

[19] Nieto, Oriol, and Juan Pablo Bello. "Msaf: Music structure analysis framework.", Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), 2015.

[20] Nieto, Oriol, and Juan Pablo Bello. "Systematic exploration of computational music structure research.", Proceedings of the International Society for Music Information Retrieval Conference (ISMIR). 2015.

[21] Schubert, Emery. "Modeling perceived emotion with continuous musical features." Music perception 21.4 (2004): 561-585.

[22] Krumhansl, Carol L. "Music: A link between cognition and emotion." Current directions in psychological science 11.2 (2002): 45-50.

[23] Steinmetz, Christian J., and Joshua D. Reiss. "pyloudnorm: A simple yet flexible loudness meter in Python."

[24] Bjørndalen, Ole Martin. "Mido: MIDI Objects for Python." (2013). http://mido.readthedocs.org/en/latest/

[25] "Qt for Python," Qt for Python - Qt Wiki. [Online]. Available: https://wiki.qt.io/Qt_for_Python. [Accessed: 29-Apr-2021].

[26] Hunter, John D. "Matplotlib: A 2D graphics environment." IEEE Annals of the History of Computing 9.03 (2007): 90-95.

[27] "PyAudio." [Online]. Available: https://people.csail.mit.edu/hubert/pyaudio/. [Accessed: 29-Apr-2021].

[28] "Pygame." [Online]. Available: https://www.pygame.org/wiki/about. [Accessed: 29-Apr-2021].