MoMusic: A Motion-Driven Human-AI Collaborative Music Composition and Performing System

Weizhen Bian[1*], Yijin Song[1*], Nianzhen Gu[1*], Tin Yan Chan[2], Tsz To Lo[2], Tsun Sun Li[2], King Chak Wong[2], Wei Xue[1†], Roberto Alonso Trillo[2†]

[1]Department of Computer Science, Hong Kong Baptist University, Hong Kong
[2]Academy of Music, Hong Kong Baptist University, Hong Kong
{19250932, 19251599, 19251130, 20235003, 19229402, 19216971, 20216572}life.hkbu.edu.hk,
weixue@comp.hkbu.edu.hk, robertoalonso@hkbu.edu.hk

Abstract

The significant development of artificial neural network architectures has facilitated the increasing adoption of automated music composition models over the past few years. However, most existing systems feature algorithmic generative structures based on hard code and predefined rules, generally excluding interactive or improvised behaviors. We propose a motion based music system, MoMusic, as a AI real time music generation system. MoMusic features a partially randomized harmonic sequencing model based on a probabilistic analysis of tonal chord progressions, mathematically abstracted through musical set theory. This model is presented against a dual dimension grid that produces resulting sounds through a posture recognition mechanism. A camera captures the users' fingers' movement and trajectories, creating coherent, partially improvised harmonic progressions. MoMusic integrates several timbrical registers, from traditional classical instruments such as the piano to a new "human voice instrument" created using a voice conversion technique. Our research demonstrates MoMusic's interactiveness, ability to inspire musicians, and ability to generate coherent musical material with various timbrical registers. MoMusic's capabilities could be easily expanded to incorporate different forms of posture controlled timbrical transformation, rhythmic transformation, dynamic transformation, or even digital sound processing techniques.

Introduction

Music has played a significant factor throughout human history though its origin is immemorial. With the prosperity of technologies, music creation has become more available and efficient, facilitating multiple music growth and lowering the threshold of creating music. The use of computers in the music generation also has a long history. The music composition system has been widely used in many areas, such as entertainment, education, healthcare, etc. The algorithm used in computer-generated music has two domains: rule-based or deep neural network (DNN)-based.

The DNN-based music generation algorithm is popular with the development of machine learning and deep learning algorithms. The typical music generation model is a variational auto-encoder (VAE) (Kingma and Welling 2013), which is used in MusicVAE (Roberts et al. 2018). With recurrent neural network (RNN) showing more powerful ability in sequential modeling, many autoregression algorithms have been proposed to generate sheet music, including melody-RNN (Waite et al. 2016), Anticipation RNN (Hadjeres and Nielsen 2020), DeepBatch (Hadjeres, Pachet, and Nielsen 2017), and hierarchical RNN (Wu et al. 2019). However, the low processing speed limits the performance of the autoregression model. To better handle long sequence inception, as an alternative, music composition methods based on transformers have been developed (Huang et al. 2018), (Huang and Yang 2020) recently.

While using rule-based algorithms for music creation has been a theoretical and practical concern since Ancient Greece, it has gained increasing prominence since the 1960s (Dean 2018). Following Landy 2009, we distinguish between note-based music, which works with sequences of usually pitch-centered events (e.g., most forms of notated music), and sound-based music, which focuses on the spectral dimension of sound. Our proposed model introduces a unique combination of both.

We argue that, broadly, all music-making involves the exploration of rules or procedures that have an algorithmic dimension (from scores to computer programs), all involving formalized abstractions of sound. Early examples of the application of computational, algorithmic solutions to music writing can be found in Iannis Xenakis (Xenakis 1992) stochastic models, the pioneering US algorithmic music by Lejaren Hiller (Hiller and Isaacson 1979) or the extensive output of the German-Dutch composer Gottfried Michael Koenig (Bertolani 2020). Further pioneering contributions were made by the US League of Automatic Composers, linked ensembles such as The Hub, and other projects such as George Lewis Voyager (Lewis 2000), a set of co-improvising algorithmic structures.

*These authors contributed equally.
†Corresponding authors.

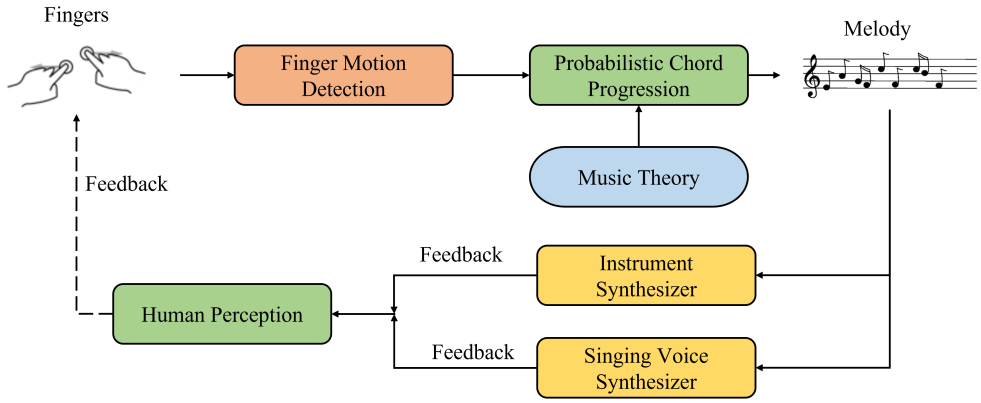

Figure 1: The diagram of MoMusic.

Building on Nierhaus historical survey of algorithmic composition (Nierhaus 2009) and Fernndez & Vicos expansion of his work (Fernández and Vico 2013), we propose partial automation of some of the processes involved in music composition (harmonic sequencing and voice-leading). We differentiate between different methodological approaches, including a) symbolic knowledge-based systems, b) Markov chains, c) artificial neural networks, d) evolutionary methods, and e) self-similarity and cellular automata processes (Fernández and Vico 2013).

Besides the algorithm, the human being is essential in music creation. Because as one of the art forms, music aims to serve humans. Human involvement in music creation plays a significant role in music's rhythm and melody. Therefore, systems that enable interactive control for music creation will be more attractive and expressive. For example, the Blob Opera (Google 2000) generates opera songs using a machine learning method, can be controlled by users, and inspires their creativity.

Here, we present MoMusic, a motion-driven human-AI collaborative music composition and performing system. MoMusic can not only generate popular instrument melodies, but it also provides a novel human voice instrument choice. Both AI technologies and music theories are involved in this system. Moreover, the performance is human-awarded since the user can control the music progression in a real-time live video. Finally, users can choose different instrument sounds according to their preferences. In addition, users can input their voice and convert it through MoMusic to synthesize precise human instruments with classic tones for later performance.

Proposed System

System Overview

The diagram of our proposed MoMusic system is shown in Fig. 1, which mainly consists of modules for a) finger motion detection, b) algorithmic music composition, and c) sound synthesis. Using a camera, the coordinates of the left- and right-hand fingers are detected. They are exploited to drive the music composition algorithm that follows the music chord, probabilistic progression model. Given the composed music melody, the music audios are generated using the timbres of instruments such as piano and violin. Apart from the conventional instruments, a controllable virtual singer is also trained to sing the melody without lyrics. A key feature of the system is that it is implemented in real-time. Therefore, a feedback loop is established so the users can judge the generated music using the perception system and adjust the subsequent music using finger movements.

Probabilistic Chord Progress Model

The systems musical material is organized following the principles of set theory (Sowell 1977). We assign a number to each pitch of the chromatic 12-tone scale, with 0 corresponding to a movable C and 11 to B natural. Our pitch matrix looks at the potential combination of two pitches, defined by the values of the X and Y axes, that may be part of a triad (three-note structure) characteristic of a Fuxian tonal harmonic sequence (Mazzola 2017). The model thus infers the third note that completes the harmony from the two-note pair determined by the fingers locations. Each triadic chord is then referred to as a second matrix that organizes chord sequences according to a probabilistic model based on a systematic analysis of chord progressions in contemporary popular music. Each chord has an N number of potential continuators organized according to their probabilistic weight, but the system applies a randomized selection process. The overall structure of the harmonic sequences moves from C to C, as we took C Major as a referential key, using G as a logical mid-point and a virtually large number of chord extensions.

One example of the chord progression is shown in Fig. 2, in which the current status is the C chord. If a G chord is touched, then one of the chords in the set{C, a, F, d, e, Bb} may follow, while if a D chord is being detected, the chord in {F, G, e, C, Bb, Eb} may follow. The subsequent actions follow the same logic as described in the above.

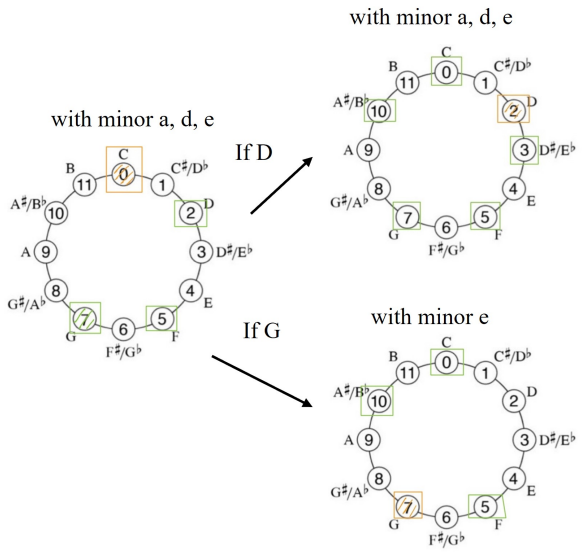

Figure 2: Chord Progression Example.

From Finger Motion Capture to Melody

Finger Motion Capture The whole system relies on human input to guide the generation of musical melodies. Considering the practicality of building an interactive system, a camera is used to capture information from humans. Since generally, the movement of hands can convey more explicit information, such as rhythm and coordinates, than body or facial movements, here we capture the finger motions from the camera and map such information to the melodies and chords.

The MediaPipe (Zhang et al. 2020), a Python library that supports estimating the coordinates of all fingers, is used for finger motion detection. Fig. 3 shows the finger tracking method used in the MediaPipe. The Hand Landmark Model in MediaPipe shows the precise location of 21 hand-knuckle key points inside the detected hands, from which we predict the coordinates of the user's forefingers. In this paper, only the index figures are used for control. The real-time video streams are captured and fed into the MediaPipe to achieve real-time processing.

As shown in Fig. 4, by using the OpenCV (Bradski 2000), an operation panel divided into a grid is also shown in the captured videos to constrain the users to move the fingers in certain valid regions. Because the MediaPipe detection can be inaccurate, a smaller operation region is used, which is set to be the upper middle part of the captured image and is found to improve the accuracy and smoothness of finger detection. Since each octave is divided into 12 equal semitones, a 12×12 uniform grid is adopted. The operation panel also facilitates converting the absolute coordinates of the fingers to integer values used as indexes in the tonal triadic combination matrix. Here, to make the controlling more flexible, the left-hand and right-hand fingers are used to define the row index Y and column index X, respectively. Based on

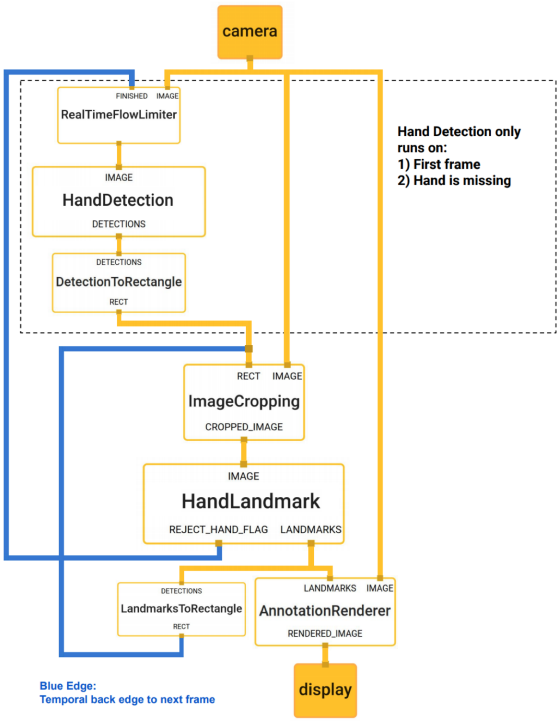

Figure 3: MediaPipe graph for hand tracking (Zhang et al. 2020).

the predefined boundaries of the grids, the absolute coordinates of the fingers are discretized by

$$X = round(\frac{X_{\text{left}} - L_{\text{bd}}}{c}), \quad (1)$$

$$Y = round(\frac{Y_{\text{right}} - B_{\text{bd}}}{c}), \quad (2)$$

where X and Y are the rows and column indexes of the tonal triadic combination matrix derived from the fingers coordinates. X_{left} and Y_{right} are the X-coordinate and Y-coordinate of the left and right fingers, respectively, L_{bd} and B_{bd} denote the left and bottom boundaries of the operation panel, and c is the side length of each grid.

We also note that when the fingers move out of the operation panel, X or Y is set to be -1, and the information is ignored in the subsequent processing.

Chord Progression Generation Based on Finger Motion Given the finger position and the probabilistic chord progression model, the chord series can be generated. In the closed-loop framework, the machine consistently updates the positions of controlling fingers and determines the X and Y indexes. Correspondence between the X and Y indexes and one of the triadic chords can be found in the "Tonal Triadic Combinations matrix. The subsequent acceptable chords and the corresponding X and Y combinations are chosen based on the probabilistic chord progres-

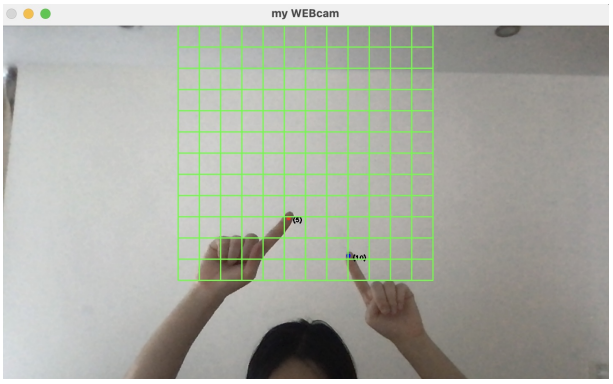

Figure 4: MoMusic operation panel.

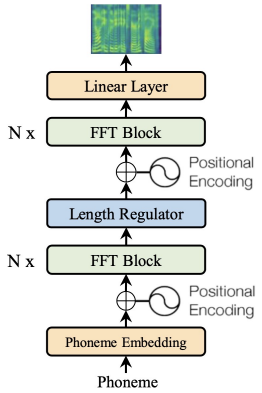

Figure 5: The structure of FastSpeech (Ren et al. 2019)

sion model. In this way, only when the fingers move into an acceptable X and Y combination the machine generates a harmonic continuation of the melody and remains in the current chord set otherwise.

Virtual Human Voice Generation

Conversion from professional pitch scale recordings In addition to the instrumental sounds (e.g., piano and violin), which can be synthesized using libraries such as fluidsynth, we also generate a virtual human that sings along with the instruments, denoted as a "vocal instrument" here. Although it is possible to generate the singing voice for different notes by simply shifting the pitch of one note recording, the singer's timbre cannot be preserved. Since large amounts of recordings for professional singers are available online, it raises the question of whether a controllable "virtual singer" can be produced to sing any note with a consistent timbre. Here, the vocal instrument is produced using voice conversion based on FastSpeech (Ren et al. 2019), and HiFiGAN (Kong, Kim, and Bae 2020), which can endow the standard high-quality human voice corpus with any human timbre so that users can synthesize new human voice instruments freely.

Model (FastSpeech+HiFiGAN) The main structure of the FastSpeech model is shown in Fig. 5. The FastSpeech model takes the audio of a standard high-quality vocal instrument as input and converts the audio features, such as pitch and energy, into the Mel-spectrogram of the target audio. FastSpeech adopts the structure of the Feed-Forward Transformer (FFT), which stacks multiple FFT blocks at the input side to realize the conversion of phonemes to graphs. Multiple FFT blocks are also present on the output side, eventually converting the timbres.

The Mel-spectrogram obtained from the FastSpeech model is not audible and thus cannot be used to construct human vocal instruments directly. The Mel-spectrogram is converted into the time-domain signal by the HiFiGAN model. HiFiGAN comprises a single generator and two discriminators: multi-scale and multi-period. During the training, the generator generates the target timbre audio, and the discriminator is responsible for determining whether a mapping is predicted or actual. After the GAN network is continuously trained, the generator's parameters are continually adjusted to generate audio from the mapping. The generator and discriminators in the MoMusic and several extra losses are introduced negatively to improve training stability and model performance. Ultimately, in time the user cannot reach the height and accuracy of the singer's singing; with the help of MoMusic's voice conversion model, the audio of everyday speech can be generated to cover a wide range of pitches for the human voice instruments.

Implementation

Virtual Human Voice Model Training

Now we describe the details of training the FastSpeech and HiFiGAN models for generating the virtual human voice.

Without of loss of generality and considering the voice quality and pitch range of the singer, 4 hours of songs by Maria Callas were collected from public sources such as Youtube and Spotify, and the vocal tracks were extracted by using the pre-trained music source separation model Demucs. The vocal signals were then resampled to 24 kHz sampling rate, and all frame-level features including the Mel-spectrogram, pitch, and energy, were computed using a window size of 25 ms and 10 ms frame hop. The FastSpeech and HiFiGAN adopted model structures the same as the original papers. During training, the batch size of FastSpeech and HiFiGAN were 32 and 16, respectively. The FastSpeech was trained for 2000 epochs and the HiFiGAN stopped after 200,000 steps.

In order to generate the virtual singing voices for different notes with consistent timbres, we recorded a singing voice of a professional singer for 12 pitches and converted the singing voice from the professional singer to Maria Callas' timbre. This way, a vocal instrument that can be flexibly controlled to sing different notes is produced.

Real-time Control

Instrument Timbre Toolbox We used the python library, mingus (Spaans 2015), to interpret music theory and handle the sheet music processing. In the sheet music processing, the finger positions were mapped to specified notes according to the "Tonal Triadic Combinations" matrix, and chords were further generated according to the probabilistic chord progression model and the finger movements. The "chord" class in mingus was used to inquire about the note combination from the chord name, and the generated music was written into bars. The musical sheets were further processed by FluidSynth (Moebert 2018), which is a MIDI synthesizer that supports 259 instruments presets and 11 drum kits, to generate the instrumental sounds. Since FluidSynth uses the SoundFont (.SF2) file format for the timbre of each instrument, we also created the corresponding SoundFont file for the vocal instrument.

Two Threads At the same time, we also adopted two threads to update the coordinates of finger detection in parallel and generate corresponding music in real time. The strategy decouples the finger motion detection and sound generation modules, simplifies the control process, and improves the stability and smoothness of the system. When the user's finger reaches the specified position, the system could instantly convert the corresponding random notes and chords according to the music theory.

Result

Through MoMusic, users can play music by changing the position of their fingers and freely choosing the instrument (traditional instruments like piano or human voice instruments) they prefer. Due to the uncertainty of the finger trajectory, the variability of the corresponding sounding area, and the randomness of the combinations of chord and note that conform to the Probabilistic Chord Progress Model, the rhythms and notes generated by the finger movements are constantly changing to form an exciting melody. The demo of the result can be found here: https://drive.google.com/file/d/1RvigW9Z1jgb2cMoIykFphX7nckrZr5ZZ/view?usp=sharing

Conclusion and Future Work

In conclusion, MoMusic is a motion-driven music composition system equipped with a probabilistic chord progress model and virtual human voice model. With proper finger trajectories, users can generate harmonious music through MoMusic. It is interactive and expressive.

Many perspectives can be improved in the future. Since only two forefingers are used, for now, we can assign different functions to the rest fingers. For example, to control the volume, playing time, instrument, etc. Furthermore, because we only implement C major chord progression in this system, more major chord progressions can be created later. We also want to achieve the result that more than two chords play simultaneously, which can generate more dulcet music.

References

Bertolani, V. 2020. Process and Form: Selected Writings on Music by Gottfried Michael Koenig. *Notes*, 76(4): 592–595.

Bradski, G. 2000. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Dean, R. T. 2018. *The Oxford handbook of algorithmic music*. Oxford University Press.

Fernández, J. D.; and Vico, F. 2013. AI methods in algorithmic composition: A comprehensive survey. *Journal of Artificial Intelligence Research*, 48: 513–582.

Google. 2000. Blob Opera.

Hadjeres, G.; and Nielsen, F. 2020. Anticipation-RNN: Enforcing unary constraints in sequence generation, with application to interactive music generation. *Neural Computing and Applications*, 32(4): 9951005.

Hadjeres, G.; Pachet, F.; and Nielsen, F. 2017. Deepbach: a steerable model for bach chorales generation. In *International Conference on Machine Learning*, 1362–1371. PMLR.

Hiller, L. A.; and Isaacson, L. M. 1979. *Experimental Music; Composition with an electronic computer*. Greenwood Publishing Group Inc.

Huang, C.-Z. A.; Vaswani, A.; Uszkoreit, J.; Shazeer, N.; Simon, I.; Hawthorne, C.; Dai, A. M.; Hoffman, M. D.; Dinculescu, M.; and Eck, D. 2018. Music transformer. *arXiv preprint arXiv:1809.04281*.

Huang, Y.-S.; and Yang, Y.-H. 2020. Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions. In *Proceedings of the 28th ACM International Conference on Multimedia*, 1180–1188.

Kingma, D. P.; and Welling, M. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Kong, J.; Kim, J.; and Bae, J. 2020. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. *Advances in Neural Information Processing Systems*, 33: 17022–17033.

Lewis, G. E. 2000. Too many notes: Computers, complexity and culture in voyager. *Leonardo Music Journal*, 10: 33–39.

Mazzola, G. 2017. The Case of Counterpoint and Harmony. In *The Topos of Music II: Performance*, 839–840. Springer.

Moebert, T. 2018. FluidSynth project page. https://www.fluidsynth.org/. Accessed 6. May 2018.

Nierhaus, G. 2009. *Algorithmic composition: paradigms of automated music generation*. Springer Science & Business Media.

Ren, Y.; Ruan, Y.; Tan, X.; Qin, T.; Zhao, S.; Zhao, Z.; and Liu, T.-Y. 2019. Fastspeech: Fast, robust and controllable text to speech. *Advances in Neural Information Processing Systems*, 32.

Roberts, A.; Engel, J.; Raffel, C.; Hawthorne, C.; and Eck, D. 2018. A hierarchical latent vector model for learning long-term structure in music. In *International conference on machine learning*, 4364–4373. PMLR.

Sowell, T. 1977. The Structure of Atonal Music. In *The Structure of Atonal Music*. Yale University Press.

Spaans, B. 2015. Mingus project page. https://code.google.com/p/mingus/. Accessed 9. January 2015.

Waite, E.; et al. 2016. Generating long-term structure in songs and stories. *Web blog post. Magenta*, 15(4).

Wu, J.; Hu, C.; Wang, Y.; Hu, X.; and Zhu, J. 2019. A hierarchical recurrent neural network for symbolic melody generation. *IEEE transactions on cybernetics*, 50(6): 2749–2757.

Xenakis, I. 1992. *Formalized music: thought and mathematics in composition*. 6. Pendragon Press.

Zhang, F.; Bazarevsky, V.; Vakunov, A.; Tkachenka, A.; Sung, G.; Chang, C.; and Grundmann, M. 2020. MediaPipe Hands: On-device Real-time Hand Tracking. *CoRR*, abs/2006.10214.