Composition as Audio Signal Processing

[*]Assignment for Computer Science Tripos – Part III – L312

Yulong Huang
Department of Computer Science and Technology
University of Cambridge
Cambridge, United Kingdom
yh419@cam.ac.uk

***Abstract*—Automatic composition methods generally use symbolic representations of music and describe the composing process with algorithms. However, symbolic encodings have limited expressiveness. By treating music as sound signals, DSP (digital signal processing) algorithms that manipulate audio inputs can be used in automatic composition. This article presents methods of implementing simple function-like composition techniques with DSP. These techniques can produce short pieces like Bach's Inventions, even though they suffer from audio artifacts, and methods for implementing complicated variation techniques are not clear.**

I. Introduction

Composition traditionally involves working with symbolic representations of music. In computer music, symbolic encodings are used to turn music into data, and composition processes are described with algorithms. One of the earliest attempts in automatic composition was carried out in this manner: notes were represented by texts and the algorithm for compositions were hard-coded [5]. A recent example is the work by Zad et al. [8], which uses MIDI files to represent music and the genetic algorithm for composition. However, the expressive power of symbolic encodings has a limit, because they abstract away details in the reality. For example, the MIDI format cannot express acoustic effects of the environment, and it cannot encode microtonal music without an extension.

One way of solving this problem is treating music as audio signals and working with them directly. No information is lost because the sound is kept in its original form. Digital signal processing (DSP) is the tool for manipulating audio signals and it is closely related to music. For instance, the pitch and the loudness of a note correspond directly to the frequency and the amplitude of a signal. Do compositional techniques have their counterparts in DSP as well? If so, how to describe composition tasks as DSP algorithms?

Composition involves creating new musical ideas and writing new musical passages based on existing ideas, like functions that take music as inputs. This article focuses on the latter – function-like compositional techniques. Section 2 shows that elementary techniques such as transposition, inversion, and augmentation are enough for composing simple classical pieces such as Bach's Inventions. Section 3 describes how to implement these techniques in audio signal processing, and Section 4 discusses the limitations in the approaches.

II. Function-like Composition Methods

This section explains the importance of function-like composition methods. In particular, it introduces *transposition*, *inversion* and *augmentation*, and demonstrates that these techniques are enough to compose one of Bach's *Inventions*.

In classical music, various works are formed by a central musical idea (called the *theme*, the *subject*, or the *motif* in different scenarios) and materials derived from it – fugues, sonatas, theme and variations, etc. The process of writing new passages based on an existing one can be viewed as a function, whose input is the existing musical idea and the output is a new one. For example, *variations*, which means repeating the theme with changes, can be seen as a function that slightly modifies the input in rhythm, melody, or harmony. Another example is *counterpoint*, which takes a melody and gives an independent melodic line in harmony with the input.

Many musical forms are essentially a collection of the theme and passages generated from the theme, put together in a fixed order. For example, the *theme and variation form* is a musical structure that presents a theme at the beginning, followed by a series of variations; a *fugue* is a contrapuntal piece that defines strictly the entry time and tonality of counterpoints to the theme. This shows the importance of function-like compositional techniques – provided an initial theme, a piece can be made with these techniques only.

Furthermore, it is possible to compose with just a few simple techniques. One example is the *Two-part Invention in C Major (BWV772)* by J. S. Bach. This piece is made of a short motif, its inversion, augmentation, and inverted augmentation (Figure 1). These passages are transposed to different registers and then assembled to make the piece. Transposition means shifting the notes up or down in pitch. Inversion is flipping the melody "upside-down", reversing its shape. Augmentation is extending the length of notes (they are doubled here). Bach's example shows that a piece of music can be made from a few function-like compositional techniques. The next problem is how to implement these techniques as functions for audio-signal inputs.

III. Signal-Processing Methods for Composition

This section explains the DSP methods for implementing transposition, augmentation and inversion for audio signals. A technical background for definitions and concepts in sound-

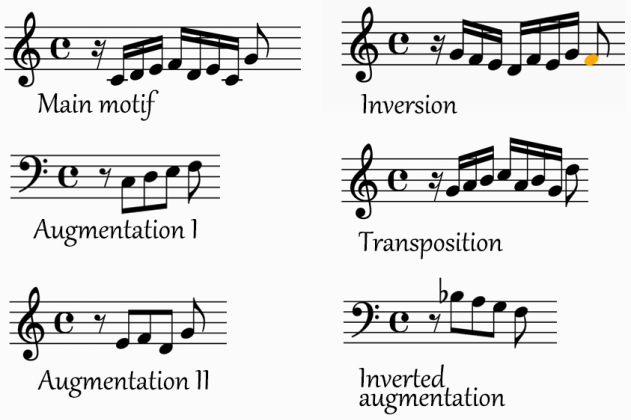

Fig. 1. Materials in Bach's Two-part Invention in C Major (BWV772). The orange note is a *mutation*, which is a small change in the motif. This article does not consider mutations for simplicity.

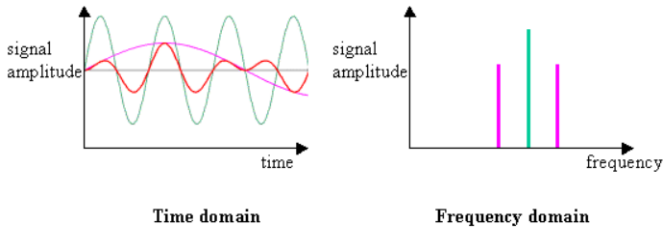

Fig. 2. Sine wave signals in time domain and frequency domain.

signal processing is provided, followed by an introduction of the OLA algorithm and time-frequency domain methods.

A. Background

Sound is made of many frequency components. The pitch humans perceive is the lowest frequency component, call the *fundamental frequency*. Components with higher frequency are called *harmonics*, and they determine the timbre of the sound. A sound signal can be analyzed in the *time domain* and the *frequency domain*. In the time domain, the signal is a function of time $f(t)$ that describes the change of air pressure at a point. The time-domain signal can be transformed to the frequency domain, which describes how much of the signal is present at a certain frequency with function $F_f(\omega)$ (Figure 2). Fourier transform is the method for transforming time-domain signals to frequency-domain signals. In practice, Fourier transform is commonly implemented by the Fast Fourier Transform (FFT) algorithm.

B. Transposition, Augmentation and OLA

Transposition is changing the pitch of notes upward or downward. In signal processing, this corresponds to making the frequency of the sound higher or lower, without affecting its duration or timbre. This problem is called *pitch shifting*. It is easy to raise the pitch of a sound signal: speeding it up increases its frequency, resulting a higher pitch. The problem is that the duration of the input sound should not change.

Time stretching is a technique that extends or shrinks the duration of sound without changing its pitch. It corresponds to augmentation in composing, which is extending the lengths of notes. Pitch shifting can be done by firstly extending the sound's duration, and then speeding it up.

OLA (Overlap and Add) is a basic time-domain method for time stretching [9]. OLA works by dividing the sound waveform in small overlapping segments, which adds up to form the original signal. A segment is obtained by multiplying the original signal with a *window function*, which is a function that is only non-zero with in a certain interval. To increase the duration of the signal, each segment is repeated multiple times, and then added up to make the extended signal. Similarly, segments can be eliminated to decrease the duration (this corresponds to *diminution* in music). With the OLA time stretching, transposition and augmentation for sound signals can be implemented.

More advanced algorithms similar to the OLA exist, for example, the *PSOLA (Pitch Synchronous Overlap and Add)* and the *phase vocoder*. These algorithms all work in three stages: the *analysis stage*, the *processing stage*, and the *synthesis stage* [1]. At the analysis stage, the input signal is decomposed to short segments. These segments are then modified at the processing stage, and finally reconstructed to the output signal at the synthesis stage. In practice, the OLA does not work well with voice and string instrument sounds [6], and the advanced methods produce better results.

C. Inversion, spectrogram and time-frequency domain

To implement inversion for sound, frequency domain methods are needed. However, pure frequency-domain methods are generally not useful in audio signal processing, because it only gives information about which frequency component presents, but tells nothing about when it appeared or disappeared. This is like a piece of music score that only records which notes are used. To avoid this problem, sound signals are transformed to a *time-frequency domain*, described by a two-dimensional function $X(t, \omega)$. The function's value at (t, ω) gives the information about how much of the signal is present in frequency ω at time t. The common time-frequency domain transformation method is the *STFT (short time Fourier transform)*. As its name suggested, STFT works by dividing the signal to short segments, and perform Fourier transform on each segment.

Since a sound signal in the time-frequency domain is a 2D function, it can be represented with an image: the value of $X(f, \omega)$ is mapped to the image colour on coordinate (f, ω). This image is called the *spectrogram*. The spectrogram can be thought of the counterpart of music score in DSP (Figure 3). Inversion in music is reflecting the music score over a note, so it can be implemented by reflecting the spectrogram over a frequency (Figure 4). The inverted spectrogram is then transformed back to audio using the inverse STFT.

Spectrogram inversion does not preserve the timbre of the sound. Consider a sound signal with many high-frequency harmonics but a few low-frequency harmonics. After inverting

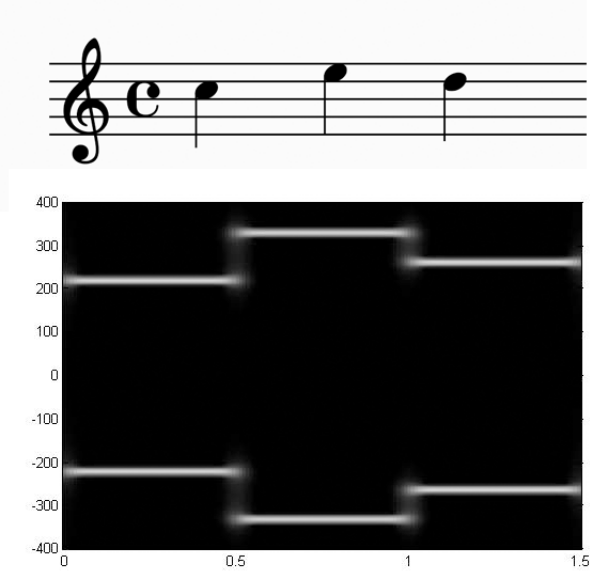

Fig. 3. Spectrogram is the counterpart of music scores.

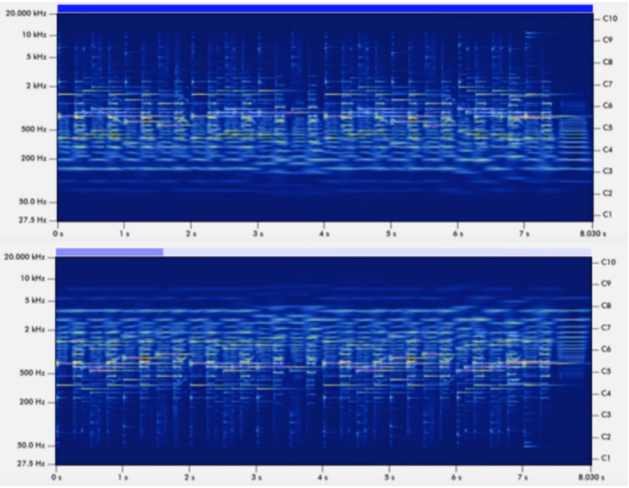

Fig. 4. An example of spectrogram inversion from [4].

its frequency band, the new signal will have a few high-frequency harmonics but many low-frequency harmonics – the timbre is changed significantly. *Octave inversion* is a refined version of spectrogram inversion that solves the problem [4]. Instead of inverting the entire frequency band of the sound, octave inversion works by inverting each octave band of the sound separately, so that harmonics with higher frequencies would not be moved to low-frequency bands.

IV. Limitations

This section discusses the limitations of methods I explained in Section 3. There are two main problems. (i) They cannot describe complicated variation techniques. (ii) They suffer from audio artifacts.

This article focused on simple function-like composition techniques: transposition, augmentation, and inversion. Apart from them, complicated variations and advanced techniques (like counterpoints) exist. For example, in Mozart's *Twelve Variations on "Ah vous dirai-je, Maman" (K265)*, the first variation turned the theme into a stream of fast, running sixteenth notes, leaving only a trace of the original melody (Figure 5). Complicated variations are hard to describe with algorithms, and they do not correspond to DSP methods directly. It is not clear how to implement them with signal-processing methods.

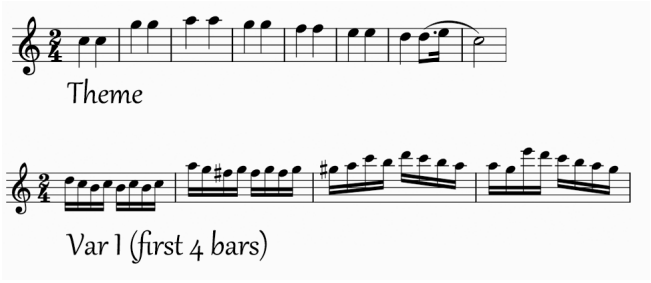

Fig. 5. Mozart's *Twelve Variations on "Ah vous dirai-je, Maman" (K265)*, the theme and the first variation.

The methods described in Section 3 also suffer from audio artefacts, which are unnatural sounds due to processing the sound signal. Examples of possible artefacts are listed below.

- Detuning: In pitch shifting, some frequency component might be shifted slightly more than others, making the output sounds 'out-of-tune'.
- Duplication: In time stretching, short aperiodic sound might be duplicated instead of stretched. For example, a short drum kick might become two kicks in the result.
- Transient smearing: a transient sound with sharp attack might be softened.

These problems can be fixed by tweaking the algorithm's parameters so that artifacts in the results are almost imperceptible, or by modifying the algorithm to reduce artifacts. However, artifacts are hard to eliminate completely in general [6].

V. Conclusion

This article introduced the idea of composition as audio signal processing. It is shown with an example that composition can be done entirely with function-like techniques and an initial musical idea. Then, algorithms of implementing three simple techniques in DSP are introduced, followed by a discussion of their limitations. The most important insight is that compositional techniques, like many other aspects in music, corresponds to concepts in DSP. Therefore, manipulating sound signals directly is a reasonable approach to overcome limitations in working with symbolic abstractions of music.

References

[1] Francis Charpentier and M Stella. Diphone synthesis using an overlap-add technique for speech waveforms concatenation. In *ICASSP'86. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 11, pages 2015–2018. IEEE, 1986.

[2] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music. *arXiv preprint arXiv:2005.00341*, 2020.

[3] Sander Dieleman, Aäron van den Oord, and Karen Simonyan. The challenge of realistic music generation: modelling raw audio at scale. *arXiv preprint arXiv:1806.10474*, 2018.

[4] Patrick Feaster. Turning audio upside down with octave inversion. https://griffonagedotcom.wordpress.com/2017/06/08/turning-audio-upside-down-with-octave-inversion, 2017.

[5] Harriet Ann Padberg. *Computer-composed canon and free-fugue.* Saint Louis University, 1964.

[6] Theo Royer. Pitch-shifting algorithm design and applications in music, 2019.

[7] Richard Savery, Benjamin Genchel, Jason Smith, Anthony Caulkins, Molly Jones, and Anna Savery. Learning from history: Recreating and repurposing Sister Harriet Padberg's computer composed canon and free fugue. *arXiv preprint arXiv:1907.04470*, 2019.

[8] Damon Daylamani Zad, Babak Nadjar Araabi, and Caru Lucas. A novel approach to automatic music composing: Using genetic algorithm. In *ICMC*, 2006.

[9] Udo Zölzer, Xavier Amatriain, Daniel Arfib, Jordi Bonada, Giovanni De Poli, Pierre Dutilleux, Gianpaolo Evangelista, Florian Keiler, Alex Loscos, Davide Rocchesso, et al. *DAFX-Digital audio effects*. John Wiley & Sons, 2002.