Death Stranding

Vasya

Abstract

This paper presents the sound piece *Death Stranding* which was composed from May to October of 2024 for the 8th annual Diffusion Festival in Baltimore. It consists of four channels which each feature entirely distinct content but which are synchronized with each other to produce a cohesive aural experience.

It was created using a Python implementation of a technique called concatenative synthesis and is an experiment in utilizing that technique to structure human voice into a dynamic composition featuring a range of different timbres to produce an effective diversity of instrumentation forming a greater sonic narrative.

I. Introduction

This piece is an exercise in large-scale composition using concatenative sound synthesis. Also known as CSS, concatenative synthesis is a technique that, like granular synthesis, involves dividing input audio data into a library, or *corpus* of short segments called *sound units* of relatively short duration. This corpus is then compared analytically with a *target* input which is matched with the nearest sound unit from the corpus based on information taken from both the corpus and the target. These selected units are finally synthesized (or concatenated) into a single output. [1]

CSS is normally used for speech synthesis but has also been implemented in musical contexts. IRCAM's CataRT for Max/MSP is a realtime implementation that has been used in live settings to provide accompaniment to musical performances [2]. The Fluid Corpus Manipulation project (FluCoMa) provides another realtime solution for Max, Puredata, and Supercollider with extensive features for controlling all stages of the synthesis algorithm [5].

II. Technical development

The present piece was composed with a nonrealtime Python script. This script differs from other implementations slightly, due to the goals of the script which, unlike most other CSS algorithms, was specifically intended for long-form composition of an entire piece rather than live accompaniment or textural flourishes.

This algorithm differs from other CSS algorithms primarily in the size computations of the sound units, which are obtained here by counting a specified number of waveform cycles (each cycle being defined as two zero-crossings), rather than being uniformly sized or based on onset detection (as is the case with many FluCoMa applications [5]). This results in a corpus of very disparately sized units, varying greatly based on complexities of the sound signal.

The concatenation of sound units is here accomplished without a windowing function[1]. Windowing is normally incorporated in order to avoid interruptions or irregularities in a signal, which was unnecessary here due to the sound units all beginning and ending at zero (thus preserving the continuity of the signal regardless of the output sequence). The script also eschews a memory sieve function common to CSS algorithms to avoid the frequent repetition of sound units. This is because the stuttering effect of consecutively repeating units stitched without windowing, usually considered undesirable for musical contexts, was in this case retained for its jarring and inorganic timbre which contrasted with the familiarity of the sound corpus to produce an effect of tension between the quotidian and the unsettling artifacts of digital manipulation, and to improve the diversity of textures present in the final work.

The selection function works by calculating the descriptors, in this case the mel-frequency cepstral coefficients (MFCCs)[2], from each sound unit and matching them with corresponding MFCCs taken from the target audio, which for analytical purposes is sliced according to the same cycle-counting function that produced the sound corpus.

Finally the selected sound units are here concatenated in sequential order, irrespective of the relative lengths of the units, or the timing of the target file. The resulting output accordingly differs in overall length and in specific timing to the actual target file depending on the lengths of each sound unit which was selected. This approach is rather unintuitive and requires extensive manual editing in order to match the timing between different iterations of the script for multichannel composition, but was a necessary concession to avoid the problem of overlapping audio which is inherently engendered by the nonuniformity of unit lengths.

[1] A windowing function in audio signal processing is a function which multiplies segments of audio by a curve with overlap between successive segments in order to smooth out the transition from one segment to the next, or as an aid in fast Fourier transform (FFT) calculations.

[2] Mel-frequency cepstral analysis is a form of frequency analysis which uses the Fourier transform, adjusted to the frequency biases of human hearing, and analyzed further by a discrete cosine transform. It is well suited to audio matching due to its ability to approximate human sound recognition and to describe timbre without including unnecessary information. For more in-depth insights about MFCCs and their usefulness see [3-4].

The Python CSS script which was used to create the composition has been published on GitHub [6].

III. Compositional process

The composition of this piece used sound data taken from two sources. The source used to create the sound corpus was the Project Coswara dataset, an audio dataset of voluntarily submitted vocal sound samples collected by the Indian Institute of Science (IISc) from individuals who either tested positive or negative for COVID-19 [7]. This dataset was intended to be used in training machine listening algorithms to detect symptoms of COVID-19 in speech data. Each sample included in the Coswara dataset consists of breathing, coughing, voiced vowels, and counting. Included is metadata relating to the subject who was recorded for every submission but for the purposes of the present composition this metadata was ignored.

The target audio that was used was a capture of the first 37 minutes of the 2019 video game *Death Stranding*, as uploaded to YouTube by the gaming publication IGN [8]. Gameplay audio was desired for its dynamic range across amplitude and frequency domains and for its clarity of sound as well as the pacing of dynamic changes in comparison to other acoustic recordings. This type of target data was uniquely rich in these qualities and allowed for an especially vibrant diversity of results. Music proved a comparatively poor target due to its cohesive range of timbres and its temporal predictability.

For each of the four channels in this piece, approximately thirty minutes of audio was arbitrarily selected by hand from the Coswara dataset, allowing no overlap between the respective selections. A separate corpus was constructed for each channel and all were matched to the same target audio. The four outputs were manually syncronized with one another and the original ~30 minutes of audio was edited to slightly under ten minutes in order to avoid redundancy. Each corpus corresponds exactly to one channel; spatialization was intentionally avoided. Additionally, each loudspeaker in this piece plays exactly one voice at any time with no overlap being present at any point.

The audio was finally mastered using matchering-cli, a Python script for easily using the matchering library for Python to master audio quickly by comparison to a target [9].

IV. Conclusion

The four channel sound piece shall serve as our conclusion. It is recommended that the audience has read this article explaining the piece before experiencing the actual audio. Failing this, the present text may be read at any time afterwards.

Acknowledgments

Zane Kanevsky provided inspiration for part of the CSS algorithm through personal correspondence.

R. M. Francis provided initial inspiration for this project through his piece *AToGRNN Prism* and through personal dialog about the compositional process.

M. C. Schmidt curated the Diffusion Festival in which this piece was presented and provided critique which informed the editorial stage of composition.

References

[1] D. Schwarz, "Current Research in Concatenative Sound Synthesis," Proceedings of the International Computer Music Conference, Barcelona, Spain, Sept. 5-9, 2005.

[2] D. Schwarz, G. Beller, B. Verbrugghe, and S. Britton, "Real-Time Corpus-Based Concatenative Synthesis with CataRT," 9th International Conference on Digital Audio Effects, Montreal, Canada, Sept. 2006.

[3] A. Jirari, "Mel Frequency Cepstral Coefficients (MFCC) for Speech Recognition," GeeksforGeeks. https://www.geeksforgeeks.org/mel-frequency-cepstral-coefficients-mfcc-for-speech-recognition/ (accessed: 17-Oct-2024).

[4] V. Velardo, "Mel-Frequency Cepstral Coefficients Explained Easily," YouTube, Oct. 5, 2020. https://youtu.be/4_SH2nfbQZ8 (accessed: Oct. 18, 2024).

[5] J. Hart, "Sound into Sound," FluCoMa, https://learn.flucoma.org/explore/constanzo (accessed: Oct. 17, 2024).

[6] Vasya, "wavmatch," GitHub, Jul. 2, 2024. https://github.com/drillkicker/wavmatch

[7] Indian Institute of Science, "Coswara-Data," GitHub, Oct. 12, 2022. https://github.com/iiscleap/Coswara-Data (accessed: Jul. 3, 2024).

[8] IGN, "The First 37 Minutes of Death Stranding Gameplay (Captured in 4K)," Youtube, Nov. 7, 2019. https://youtu.be/KevFGaS298s (a)ccessed: Jul. 3, 2024).

[9] S. Grishakov, "matchering-cli," GitHub, Jan 11 2020. https://github.com/sergree/matchering-cli (accessed: May 17, 2024).