Reconstruct and Decompose: Extracting and Sonifying Rhythmic, Melodic and Spectral Patterns for Analytical and Creative Use

Dr. phil. Egor Polyakov
HMT Leipzig
egor.poliakov@hmt-leipzig.de

Personal Background

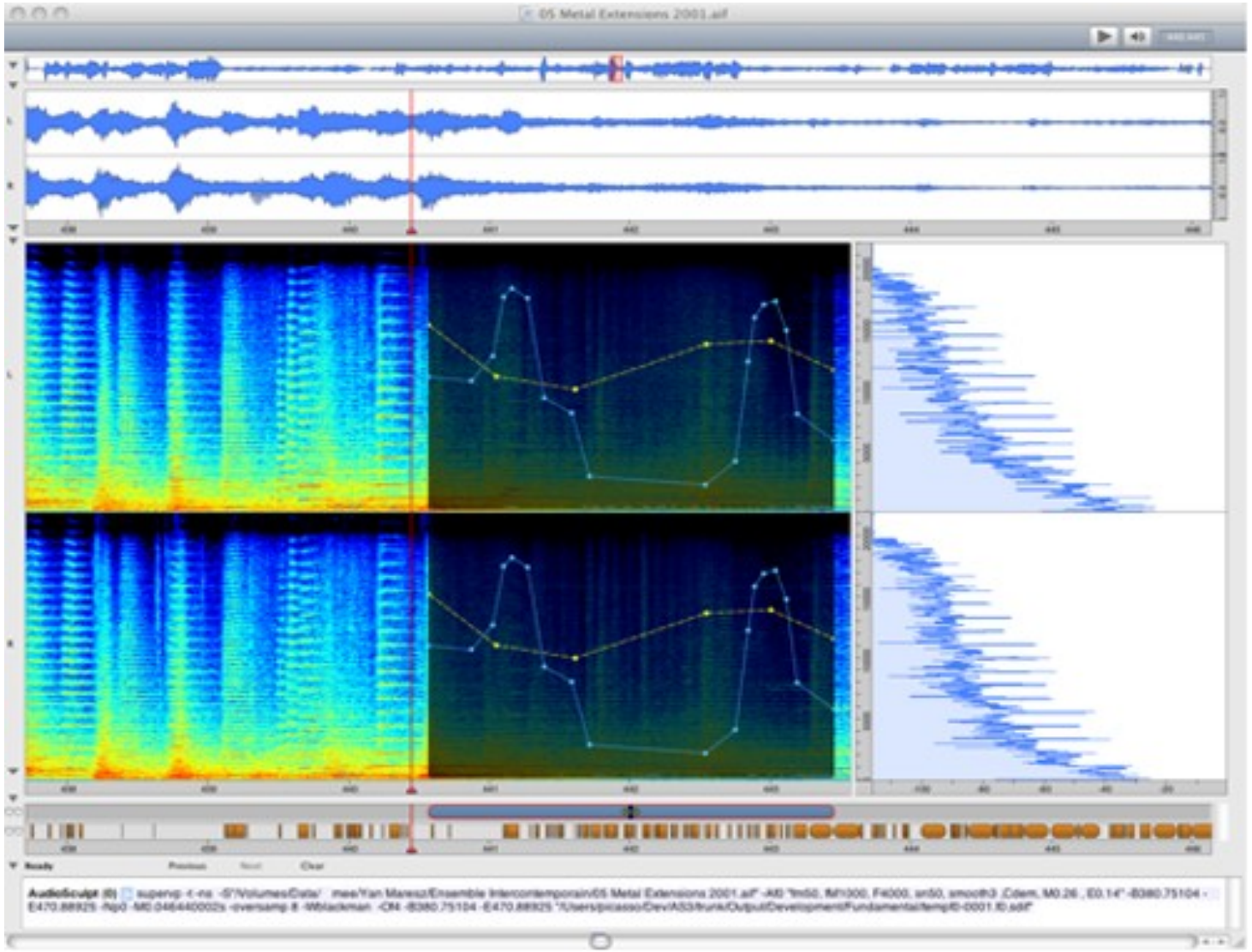
- Post-doc researcher: HMT Leipzig
- Specialization:
  - Electroacoustic composition
  - Computer-assisted music analysis
- Roles:
  - Lecturer: Music analysis, informatics, acoustics
  - Personal tutoring: Realizing live-electronics & fixed media compositions
- Software Proficiency:
  - Max/MSP, Open Music, Audiosculpt, Csound
- Programming: Python

Teaching Emphasis

- Core focus: Spectral analysis & FFT-based sound processing

=> Many students lack programming experience!

- Solution: Emphasis on accessible and user-friendly tools
- Favorite tool: IRCAM Audiosculpt
  - Pros: Reputable for spectral analysis & sound resynthesis
  - Cons: Not open-source or cross-platform
- Addressing Audiosculpt limitations:
  - Secured volume license
  - Ensured OSX computer access for students



Waveform

FFT Analysis/Manipulations

Command line input (supervp)

Audiosculpt (1993-2019)

FFT-based Resynthesis Method by Marco Stroppa (IRCAM)

1. Analyze using Audiosculpt, then save as SDIF
2. Load the SDIF in OM and resynthesize using OM2Csound/OMChroma
3. An alternative: Utilize SuperVP as a command line tool for both analysis and resynthesis

Challenges

The sonification and resynthesis processes of analyzed material in these environments were tedious, necessitating the use of several software platforms:

- Requires familiarity with three distinct environments: Audiosculpt, OM, and Csound
- Limited tutorials available, making it impractical for instructional use

Issues with Audiosculpt:

- Limited to 32-bit; IRCAM did not update to a 64-bit version
- Not compatible with OSX version 10.15, introduced in 2019
  - Implication: Post-OSX 10.15 Apple computers couldn't run Audiosculpt

Audiosculpt Alternatives:

Spear

- Faced several bugs during use
- Design revolves around partial tracking and linear prediction

Sonic Visualizer

- Absence of SDIF support; cannot be merged with the OM setup

Max/Max for Live Patches:

- Developed various patches specifically for FFT-Resynthesis, including pitch and freeze effects
- Pros: Well-suited for real-time processing
- Cons:
  - Lacks detailed signal and spectrum visualization
  - No support for signal related offline calculations
  - Not open-source

Towards a Unified Solution:

- **Initial Engagement**: Delved into Python in 2020, deepening my expertise with symbolic music representations during the CAMAT project at HfM Weimar (2021-2022)
- **Inspiration**: Heavily influenced by the FMP Notebooks by Meinard Müller
- **The Objective**: Set out to devise a **Py**thon-based solution, echoing Stroppa's techniques. The goal: A unified tool eradicating the need to navigate three separate environments
- **Features and Vision**:
  - Emphasizing extensive visualizations with Plotly to foster accessibility and enhance the learning experience
  - Leveraging pandas data frames as the core container for FFT/DFT data.
  - Employing Verovio for symbolic notation

AudioSpylt

- **Py**thon-based toolbox tailored for sound analysis, re/synthesis, and diverse visual & symbolic sound representations
- Operates within the Jupyter Notebooks environment
- Predominantly designed for instructional use, emphasizing varied resynthesis patterns. Initially developed for composition students I mentored last year.
- Alpha version available on GitHub

Audio Generation with Python: The Basics

Visual Analogy: Generating audio can be likened to plotting a waveform

- Think of the audio waveform as a visual plot where time is on the x-axis and amplitude is on the y-axis

Sampling Rate: Consider it as the "resolution" of your canvas

- Higher sampling rates provide more detailed audio representation, similar to how high-resolution images are more detailed

Amplitude Representation: Python uses IEEE 754 double-precision binary floating-point format by default

- In layman's terms: This precision is more than adequate for representing audio, even at 32 bits

Complex Spectra Creation: By summing multiple sine waves, you can create more intricate audio spectra

- Each sine wave can represent a distinct frequency component. When they're combined, they form a complex audio signal

Basic Representation: Frequencies and their corresponding amplitudes can be tabled

- This table serves as a fundamental blueprint of the audio's spectral content

	Frequency (Hz)	Amplitude
0	240.0	0.004024
1	320.0	0.001932
2	395.0	0.001883

without time domain

	freq_start	freq_stop	time_start	time_stop	amp_min	amp_max
0	71.428571	71.428571	0.03	0.1	0.494724	0.494724
1	128.571429	128.571429	0.03	0.1	0.477712	0.477712
2	157.142857	157.142857	0.03	0.1	0.465799	0.465799
3	185.714286	185.714286	0.03	0.1	0.511994	0.511994
4	242.857143	242.857143	0.03	0.1	0.567231	0.567231
5	271.428571	271.428571	0.03	0.1	0.604748	0.604748

with time domain

Toolbox overview

1. **Instructional Notebooks**: Designed to provide comprehensive explanations and demonstrations on core audio concepts

- **wave_sampling_window**:
  - Covers sampling rate, Nyquist Frequency, window functions, and implications of sampled material length on frequency resolution
- **wave_vs_dft_3d**:
  - Displays 2D and 3D representations of DFT spectra, emphasizing sine/cosine component visuals

2. Analysis Notebooks:

- **audio_load_dft:**
  - Incorporates basic audio editing functions such as trim and fade
  - Offers customizable peak detection methods
  - Features thresholding functions and split analysed data into multiple DFTs

3. Visualizations and Symbolic Rendering:

visual_tsv:

- plotting scripts for TSV/data frames

symbolic_mei:

- symbolic visualizations tailored for data frames

4. TSV Manipulations and Resynthesis:

- **df_pitch_stretch:**
  - Implement pitch/stretch alterations on TSVs with time domain data
- **2df_copypaste, 2df_merge**:
  - Execute freeze effects and various kinds of spectral interpolation
- **resynth**:
  - Resynthesize based on TSVs containing time domain data

Why Verovio?

Limited Rendering Options in Python:

- Python presents limited choices when it comes to sheet music rendering

Limitations of Music21:

- Music21 relies on external programs such as LilyPond or MuseScore for rendering MusicXML data
- The rendering process results only in PDF or PNG formats

Verovio's Native Python Integration:

- Verovio can operate as a native package within the Python environment
- Support for MEI, MusicXML, Humdrum
- Rendering directly as SVG, providing scalable and high-quality sheet music graphics

Issues with Current Symbolic Music Formats

1. Lack of Modern Notation Support:

- Current symbolic music formats largely cater to traditional notation techniques
- Modern notation, particularly those employed in contemporary, experimental, and electroacoustic genres, often isn't adequately represented

2. LilyPond's Limited Scope:

- While LilyPond stands out by offering some degree of support for unconventional notation techniques, it only has a brief section dedicated to these

3. Increasing Demand for Modern Notation:

- As music from the 20th century enters the public domain, there's an influx of compositions using graphic or microtonal notation
- Urgent need for updated symbolic music formats that can accommodate these notations

4. Transcription and Annotation Needs:

- The rise of live-electronics and acousmatic music genres necessitates accurate transcription and annotation tools

"The sonogram, a graphic spectral analysis by computer, has been regarded as a solution to the visual representation of electroacoustic music. I regard it as a very useful aid rather than a solution. A sonogram is a type of literal spectral analysis at a chosen resolution: at too high a resolution detail becomes lost in a blur; at too low a resolution there is insufficient detail. But a sonogram is not a representation of the music as perceived by a human ear – in a sense it is too objective. Its shapes therefore have to be interpreted and reduced to perceptual essentials. In other words, someone has to decide what to retain and discard from the representation, and more particularly, try and determine how much detail is pertinent to the alert listener. ... How much is too much, and how much is not enough?" (p. 108).

Smalley, D. (1997). Spectromorphology: explaining sound-shapes. *Organised Sound, 2*, 107 - 126.

"Sonograms are undoubtedly very useful images when used in analysis, particularly when dealing with complex spectra found in some sound-based compositions. The issue to be raised here is: can we hear everything that we see in these images? Of the information we cannot perceive, how relevant is it in the end?" (p. 203).

Landy, L. (2007). Understanding the Art of Sound Organization. MIT Press.

Thank you!

AudioSpylt alpha:
https://github.com/klirr2007/audiospylt
E-Mail: egor.poliakov@hmt-leipzig.de

Credits to Haewon Son, Xiaolin Pan and
Niayesh Ebrahimisohi for feedback and testing