SONAGRAPH. A CARTOONIFIED SPECTRAL MODEL FOR MUSIC COMPOSITION

Andrea Valle
CIRMA/StudiUm - Università di Torino
andrea.valle@unito.it

ABSTRACT

This paper presents SonaGraph, a framework and an application for a simplified but efficient harmonic spectrum analyzer suitable for assisted and algorithmic composition. The model is inspired by the analog Sonagraph and relies on a constant-Q bandpass filter bank. First, the historical Sonagraph is introduced, then, starting from it, a simplified ("cartoonified") model is discussed. An implementation in SuperCollider is presented that includes various utilities (interactive GUIs, music notation generation, graphic export, data communication). A comparison of results in relation to other tools for assisted composition is presented. Finally, some musical examples are discussed, that make use of spectral data from SonaGraph to generate, retrieve and display music information.

1. INTRODUCTION

Access to spectral information is crucial for a large number of audio-related practices spread along the sound/music continuum, such as sound synthesis and processing [1, 2], audio restoration [3], composition for acoustic instruments (algorithmic/assisted composition [4, 5]), music information retrieval [6]. The great majority of available applications are based on Fourier transform, as implemented by the Fast Fourier Transform (FFT) algorithm. The efficiency (in computational terms) and the effectiveness (in terms of results) of the FFT are well known. While data gathered via FFT allow for reconstruction and manipulation of the input signal (audio level), they are not immediately suited for perceptual and music tasks, so that various post-processing operations are needed to extract music information [6]. In contrast to the audio level, this higher level, both perceptual and musical, might be called –following [7]– sound object level (see also [8]). Starting from FFT data, in order to pass from the first level to the second, a double conversion step has to be taken into account, that concerns both time and frequency [6]. With respect to FFT, time resolution is defined by the hop size (the step size in which the window is to be shifted across the signal), while the phenomenological one instead concentrates on a (not obvious) notion of sound event. The other step is required to convert frequencies (in Hertz) into a pitch-based representation (a typical but not necessary example being 12-TET). This step is notoriously problematic because FFT samples frequency in a linear fashion, while the tonotopic representation in the ear, thus at the basis of musical practices, is logarithmic. This means that half of the information output from an FFT concerns, in perceptual terms, the highest octave, a quarter the second octave, and so on. In essence, high frequencies are over-represented and/or low frequencies are under-represented [2]. Spectral information is typically inspected visually, and many data visualization softwares and libraries are available to generate sonographic (i.e. spectrum over time) representations. Well-known applications targeted at the music domain are Sonic Visualiser [9], Acousmographe [1], ianalyse5 [2]: as they all rely on FFT, they all provide a sonographic visualization based on a linear distribution of frequencies. This issue is often partially solved by drawing logarithmically the frequencies, but since the starting information is linear with respect to frequency, the result is usually a sonogram that looks blurred in the low register while becoming very detailed in the highest one. To sum up, FFT spectral data are in principle too large and only partially fitting if the sound object level is at stake. While typically these issues are practically solved with satisfying results, still they leave room for different designs.

2. MAIN GOALS AND INTENDED USERS

SonaGraph is a spectral analyzer that, by means of a very basic design, aims at providing a spectral representation in form of a sketch, assuming that a sketch results in minimal but correct, clear and relevant information about a certain sound object. Because of its barebone structure, such an analyzer is efficient as it performs a large data reduction. While not fitted for audio manipulation (because of data loss), it allows to gather and easily manipulate musically relevant data from the musician's perspective (i.e. at the sound object level). Its design is inspired by the so-called "cartoonification", proposed in the audio domain in relation to the modelling of acoustic behaviour [10]. Cartoonification is a procedure that leads to a drastically simplified model that nevertheless remains consistent with some general principles. SonaGraph is intended as a tool bridging the audio level to the symbolic one by providing a simplified spectral data structure that can be easily understood by musicians without a particularly deep knowledge of the

[1] http://inagrm.com/en/showcase/news/203/acousmographe

[2] https://logiciels.pierrecouprie.fr

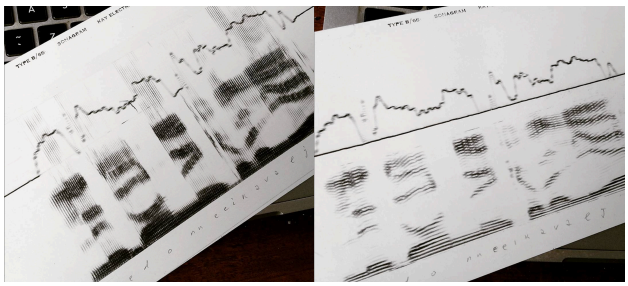

Figure 1. Sonagraph tracings: large (left) and narrow (right) bandwidth filter setting.

mathematics at the basis of more accurate methods and frameworks (e.g. [11]). As spectral information is immediately converted into pitch, standard music symbolic concepts and operation (e.g. chords and the relative typical manipulation) can be directly applied by the musician to spectral data. Moreover, thanks to the one-to-one correspondence between spectral data and music symbols, automated music notation can be generated easily while precisely representing the reduced spectrum. In the following, the general framework is introduced, then an application is presented. As musicians are the intended target, a comparison with BACH [12], a state-of-the-art tool used by composer in the same context (assisted composition including spectral analysis and automatic notation transcription) is provided. Finally, two musical applications are discussed.

3. THE ANALOG SONAGRAPH

The SonaGraph model is inspired by the Sonagraph, an analog device initially developed at Bell Labs in telecommunication field [13] and commercialized by Kay Electric from the '50s. The Sonagraph allowed to plot a representation of sound spectrum over time (a "sonagram"). Its hardware implementation was based on a bank of heterodyne filters performing a multistep spectrum scan (thus, it did not work in real time), connected to a stylus that burned progressively (i.e. frequency by frequency) a special paper foil [13, 14]. Time resolution depended on the (adjustable) rotation speed of the cylinder on which the paper was drawn. Features of the Sonagraph were the possibility of using two different bandwidths (large and narrow) and the ability to plot frequencies in a dual mode, that is, linearly and logarithmically. Figure 1 shows two examples of Sonagraph's tracings for the same signal respectively with large and narrow bandwidth setting for the filter bank, both plotted with linear frequency (each tracing also includes the signal's amplitude envelope on top).

By making sound structure extensively accessible for the first time, the Kay Sonagraph allowed fundamental advances in two domains. The first is acoustic phonetics, starting from the 1952 pioneering study of Jakobson, Fant and Halle [15], that, by means of sonagraphic exploration, led to the acoustic definition of phoneme (even if the very first use can be traced back to 1947 [16]). The second is bioacoustics, and in particular ornithology. Once available, the Sonagraph immediately became a fundamental instrument for the study of bird singing as it provided visible and annotable forms to represent the extraordinary variety of ornithological phonations [14]. For Pieplow, this "golden age" of the Sonagraph extended for more than forty years, from 1951 to 1995 [3]. A third application has been crucial for the fate of contemporary music, and thus it is particularly relevant here. In the domain of acoustic analysis of music instruments, Leipp made abundant and pioneering use of the Sonagraph to exemplify a vast set of acoustic phenomena and music instrument behaviours [17]: Leipp's approach was a fundamental pivot for the reflection that would lead during the '70s Grisey and his companions to the technical and aesthetic formulation of Spectralism [18]. The Sonagraph for the first time showed sound as a sort of virtual score: a form of notation in which notes are replaced by continuous graphic elements [19]. Bridging ornithology and music, Mâche's zoomusicological approach largely relied on sonagraphic traces [20].

Indeed, since more than twenty years the Sonagraph has been superseded by FFT-based softwares [14]. Interestingly, Leipp [17] strongly supported the perceptual appropriateness of the sonagraphic display even if all his examples were plotted linearly. In the ornithological field, on the contrary, Marshall insisted with equal emphasis on the perceptual requirement of a logarithmic frequency display, due to the tonotopic organization of the auditory system, a feature common to all vertebrates, including both birds and humans. For Marshall, linear frequency displaying was simply "absurd" [21].

4. A CARTOONIFIED MODEL

The Sonagraph provides a reference for a "cartoonified" model of spectrum analysis based on a filter bank. The idea of a filter bank that extracts information from audio signal is actually at the basis of the venerable technique of the Vocoder, proposed at Bell Laboratories by Homer Dudley in the late 1920s as a device for representing the vocal signal (by means of an "encoder") and then resynthesizing it (through a "decoder" component) [22]. The main issue with a filter bank technique is that it does not respect the phase, which is crucial in signal reconstruction. Hence the digital algorithm of the Phase Vocoder that, as the name indicates, instead takes into account phase, and can be seen as a filter bank interpretation of Fourier Transform [23]. While FFT is the standard spectral tool in music information retrieval [6], filter bank approaches to frequency decomposition have been proposed as an alternative [24, 25]. Even if phase is required for signal reconstruction (i.e. at the audio level), it can be discarded in case of spectral analysis targeted at musical information extraction (i.e. sound object level). In relation to this aspect, it can be observed that the typical output in terms of musical information relies on standard music notation (the so-called common practice notation, CPN). Thus, as already discussed, if FFT data are gathered, a large data reduction has to be performed in order to output CPN, and various techniques have been proposed accordingly [6]. A

[3] http://earbirding.com/blog/archives/1229

cartoonified approach to the problem can be pursued by reversing the perspective, that is, by starting from output data, i.e. pitches. While MIDI protocol encodes pitches in 2^7 values ($0 - 127$), typical musical data in CPN are satisfyingly represented by means of a piano keyboard (88 pitches). In both cases, the amount of data is fairly lower than typical FFT frequency resolutions. Following a cartoonified approach, in the SonaGraph architecture (Figure 2) the analysis step is thus performed through a bank of 88 constant-Q bandpass filters, tuned logarithmically in relation to piano keys (Log filter bank). The rationale for such a "musical" choice is to achieve a good compromise between the pitch resolution and the size of the bank, while covering more than 4 KHz (precisely, $27.5 - 4186$ Hz). The filter bank has an overall tuneable resonance factor Q (therefore a variable band depending on the central frequency), on the model of the narrow/wide band distinction of the analog Sonagraph (but more general). Each filter is connected to an amplitude envelope follower (Amp follower), as in Dudley's Vocoder encoder. The output signal resulting from amplitude following is slightly integrated (Smoother: in fact, a sort of low pass filtering) to eliminate too rapid variations. While this operation permanently compromises phase information, it provides a clearer information on amplitude variation at the sound object level (lower time resolution). Each filtered signal, once smoothed, is then converted from linear amplitude to deciBel (Converter) and sampled at a regular rate (Sampler). The sampling rate (sr) determines the spectral time resolution and can be adjusted (as in the variable speed of the metal drum in the Sonagraph). Appropriate values for sample rate depend on the analysis' goals and on the spectral variation of the signal that is being considered: empirically, values between 10 and 50 Hz (in this last case, already at audio rate) provide a good compromise between the sound object level and audio accuracy. For each sample, the Sampler module returns a vector of 88 values estimated in dB (hence on, bin), which is stored as a column in a 2D matrix. In the latter, intuitively, rows represent the time domain, each containing the values of the sampled signal for a single filter at rate sr. The two-dimensional signal thus obtained can be analysed in real-time (Analyzer) or stored (Archivist) and imported later. With respect to the audio signal (and the FFT one), Analyzer takes into consideration an extremely reduced data amount, making the implementation of operations on the single bin –and more generally on subsets of the matrix– very simple and computationally inexpensive, thus fitting real-time operations. For example, one can easily investigate maxima and minima in spectral regions by selecting certain frequencies or obtain information about spectral peaks, e.g. ranking the first n frequencies in relation to amplitude or with respect to a threshold (i.e. frequencies with amplitude higher than t).

The SonaGraph cartoonified analysis framework is represented in Figure 2, region A, while region B depicts some added functionalities. The frequency resolution is indeed calibrated on the 12-TET system. The latter is intended as a standard reference grid providing a uniform perceptual pitch sampling as codified by Western practice. But

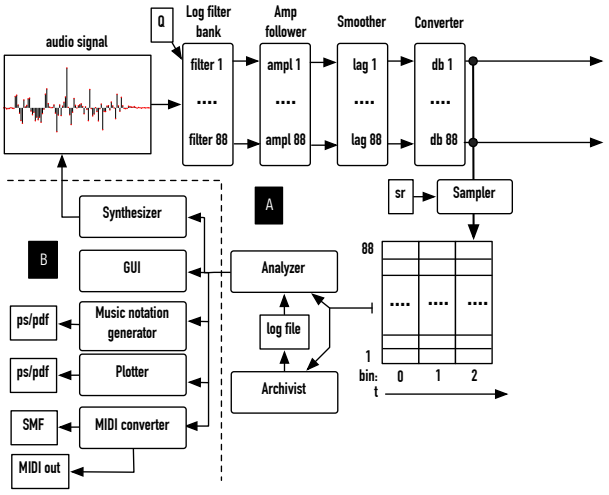

Figure 2. SonaGraph architecure.

the filter bank can be easily modified, e.g. it can be tuned by providing an array of frequencies rather than a uniform half-tone step (see [26] for a discussion on the relations between spectrum and tuning) or it can be expanded/reduced in size. As an example, to reach a quarter-tone resolution, the bank can be doubled in size in order to double resolution, or, in a non real-time fashion, two filtering processes can be run with different tunings, then gathered data can be properly assembled.

5. APPLICATION DESCRIPTION

The SonaGraph framework has been implemented as a set of classes for the SuperCollider audio and music programming environment [27, 28]. The latter provides sound processing capabilities via its audio server component, but also GUI programmable elements, MIDI support and a high level OOP language to manipulate spectral data, thus seamlessly bridging audio and sound object level. Moreover, other functionalities are available via internal access to the Unix terminal. All the functionalities represented in Figure 2 are encapsulated in the classes described by Figure 3: boxes represent classes while operations are indicated by labels written in plain text. Underlined text indicates the three main operation metacategories: analysis, integration/communication and graphic export.

SonaGraph is the main class, providing general audio management (including resynthesis), sampling, analysis, archiving. In relation to Figure 2 it implements section A plus Synthesizer and MIDI converter from section B. It also provides a common unified method interface for the other classes (integration, in relation to Figure 3). The filter bank is implemented via second order band pass filters, with a default (but modifiable) very high Q ($= 1000$), that has proven effective. Other audio functionalities include a fundamental frequency detection analysis (implementing the Tartini algorithm [29]), synced to the same sampling rate of the Sampler, and a bank of sine oscillators and a virtual piano instrument, both to resynthesize spectral data as a control step (Synthesizer in Figure 2). SonaGraph also includes MIDI support, both in terms of real-time MIDI

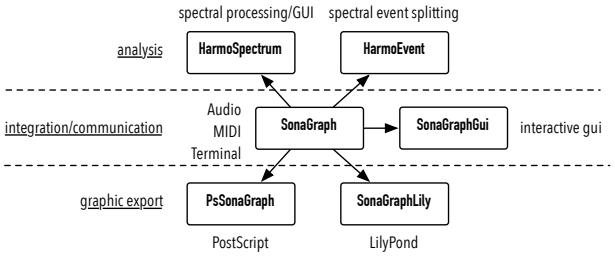

Figure 3. SonaGraph classes.

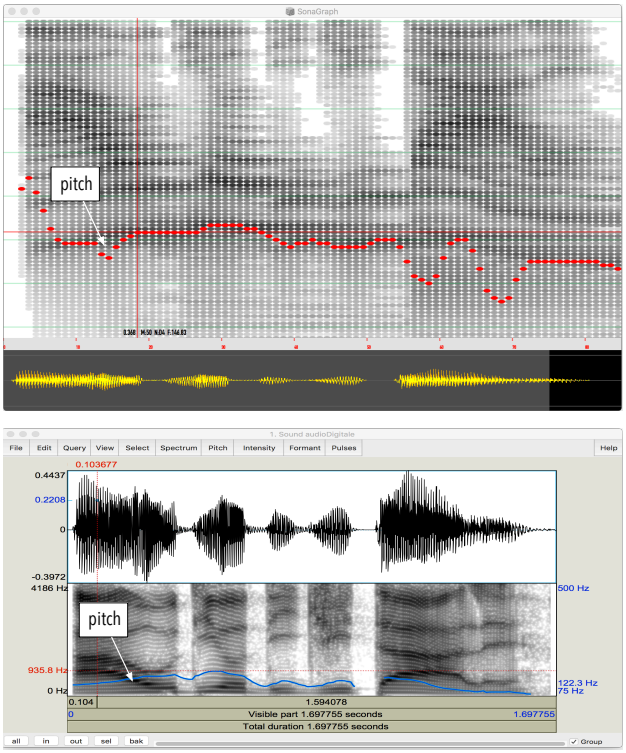

Figure 4. SonaGraph interactive GUI (top) and Praat (bottom).

communication (e.g. to MIDI synthesizers, as those supported by DAWs) and Standard MIDI File (SMF) creation. The SMF export creates voices by selecting rows in the sonogram. In order to simplify the MIDI file structure, a voice grouping algorithm is applied, so that consecutive pitches having an amplitude greater than a selected threshold are grouped together in a single note.

The SonaGraphGui class supports an interactive GUI for inspection and playback of the analyzed sound. Its aim is to help the exploration of gathered data both visually and aurally. Mainly inspired by Praat GUI [4], it provides a scalable window showing the sonogram (top) and the waveform (bottom), and including also a visualization for the estimated fundamental pitch. Figure 4 shows the SonaGraph GUI (top) an the Praat GUI (bottom) for the same signal (a voice sample: formants are apparent) for sake of comparison. In the SonaGraph GUI, mouse-pointing in the window results in a vertical red line indicating the selected bin and a horizontal one showing the frequency.

[4] http://www.praat.org/

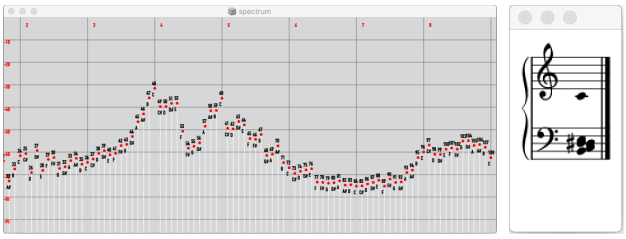

Figure 5. HarmoSpectrum GUI.

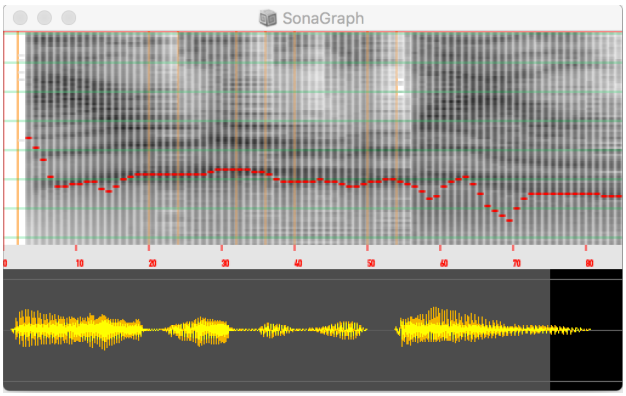

Figure 6. HarmoEvent onset data in the GUI.

On the left of the vertical line, time is indicated (in Figure 4: `0.368`), on the right MIDI pitch (`50`), note (`D4`) and frequency (`146.83`) are shown for the selected point. Space bar allows to start/stop source playback from the selected bin. While clicking, a synthesized short piano note is generated to provide a reference for the pointed pitch. A threshold for amplitudes can be set, so that values under the threshold are not shown. In the SonaGraph GUI, pitches in red indicate the fundamental pitch estimation (like the blue line in the Praat GUI, in both cases a label has been added to the GUIs in Figure 4 to help the reader). Note that formants are highly visible in a speech sound (and help comparison between the GUIs), but they do not coincide with the fundamental pitch.

The HarmoSpectrum class features all the methods for spectrum processing, including interactive GUI and music notation generation. Conceptually, it operates on a single bin while operations on multiple bins (i.e. selecting a bin range representing more time samples) are handled by SonaGraph by averaging amplitudes and passing the averaged single bin to HarmoSpectrum. Data are available to the SuperCollider language for further manipulation and can be explored visually through an interactive dedicated GUI (Figure 5, left). The GUI shows the spectral envelope for the selected, averaged spectrum (bins 40 to 50 from Figure 4, left): each component is labelled with MIDI note (top) and symbolic name (bottom), while octaves are indicated by vertical lines. By clicking on the window, a piano note is played back for the selected pitch as a reference. As it can be seen from spectral peaks in Figure 5, left, the most prominent pitches in MIDI notation are 47, 48, 50, 51, 52, 60 (49 is just under the selected threshold). Given an amplitude threshold, they can be viewed directly in musical

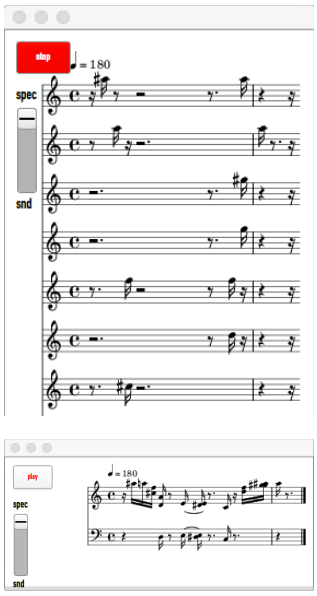

Figure 7. Generated notation imported into GUI.

notation by invoking a dedicated method that exploits the LilyPond environment for notation description and typesetting [30]. The method writes LilyPond code on a temporary file, renders it by calling the LilyPond executable via Unix terminal, and loads the rendered image into a window (Figure 5, right). By clicking on the window, the chord is played back using a synthesized piano. The LilyPond code can be written on a user-specified file so to remain accessible for further usage.

HarmoEvent performs some basic operations on sonogram evolution over time in order to recognize discontinuities. As the only information is the spectral one, it detects an "event" by comparing the difference between averaged amplitudes of adjacent bins. Two parameters are available: a threshold value for minimum amplitude difference, and minimum distance (in bins) between events. While very crude, this operation implements a typical approach to onset detection by spectral-based novelty function (or spectral flux, [6]). Once detected, events can be automatically visualized in the GUI (see vertical orange lines in Figure 6, that also demonstrates different displaying threshold and ratio with respect to Figure 4). A segmentation procedure is available that split the signal between adjacent onsets, thus obtaining event sub-signals: it applies a minimum envelope to avoid clicks, and exports the resulting event signals to audio files.

As already discussed, CPN visualization and export are crucial in bridging spectral content to music application. Following the model discussed for HarmoSpectrum, SonaGraphLily manages the mapping from sonographic data (rather than spectral snapshots) to music notation by generating LilyPond code. It creates LilyPond text source files, renders them as graphic files and –if needed– loads them into GUI for real-time playback. Automatic generation of notation is a complex topic, and various heuristic approaches have been proposed [31, 32]. In our case, the threshold setting for filtering out lower amplitudes is

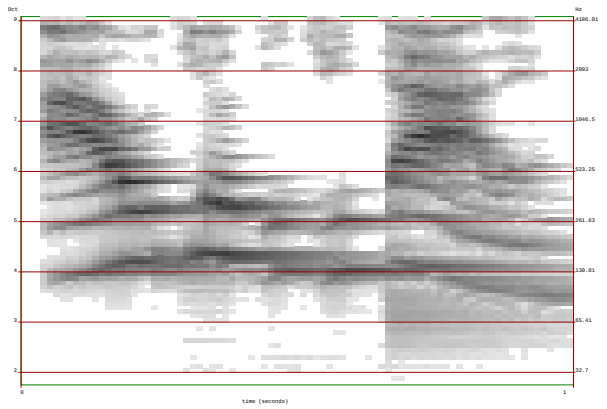

Figure 8. A PostScript rendering.

crucial in order to avoid cluttered notation. Two solutions are provided by the SonaGraphLily class. In the first case, voices are created from sonogram pitch rows and displayed on separate staves. While this visualization is useful from an analytical point of view, the number of voices may escalate quickly (with default values, up to 88), thus becoming visually unmanageable. A second solution groups notes according to a standard piano notation, with much more (but potentially too much) compact results. For sake of comparison, Figure 7 shows for the same audio sample (again, the one from Figure 4) the two notations as loaded in the GUI. As in SMF export, a voice grouping algorithm is applied (not applicable in Figure 7). The voice-based notation window (Figure 7, top) has been cut to 7 voices (of 16) for sake of readability. The transcription algorithm transparently maps the sonogram's time resolution by assigning a semiquaver duration value to each bin. Tempo is thus calculated by taking into account the sample rate (here, 12 Hz $= 12/4 \times 60 = 180$ bpm), while meter is set to $4/4$. In Figure 7, the rendered notation files are loaded into a GUI, that allows for playback using both synthesized piano (for note data) and the original audio sample (as a comparison), providing also a crossfade slider for variable mixing (spec/snd, spectrum vs sound).

Finally, the PsSonaGraph class is dedicated to graphical export of the sonogram, using the standard (but customizable) grey scale for plotting amplitudes of frequencies over time. It creates a PostScript file [33] from analysis data with adjustable graphic parameters, and converts it into PDF format via terminal utilities. Figure 8 shows a PostScript output from the sonogram in Figure 4, by setting a higher amplitude threshold. It includes reference octave and frequency annotations at each side of the red lines.

6. A SHORT ANALYTICAL COMPARISON

Although cartoonified, to an informal perceptual appreciation the frequency resolution of SonaGraph provides useful and adequate clues on the spectral information of the sound taken into account, even if the model per se is undoubtedly oriented specifically towards harmonic information. In order to further verify the results, in this section two automatic transcriptions from spectral data over time

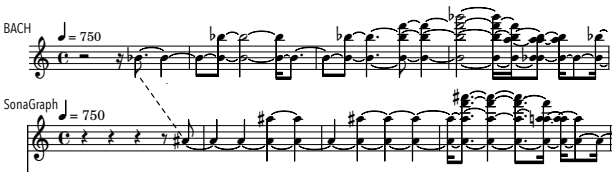

Figure 9. Music notation transcription for "trumpet".

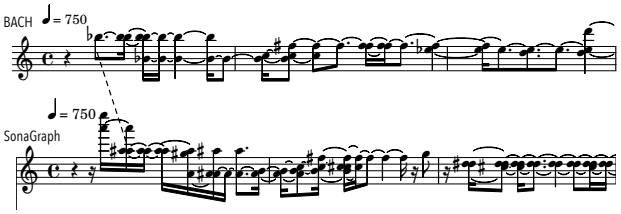

Figure 10. Music notation transcription for "wind turbine".

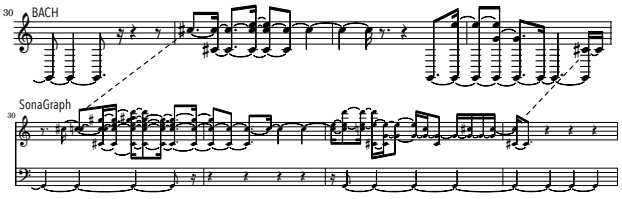

Figure 11. Music notation transcription for "octandre".

into musical notation are compared. In particular, the comparison has been performed between SonaGraph and Audiosculpt [5] /BACH [12]. In Audiosculpt, FFT data are processed via partial tracking, in order to filter out amplitudes lower than a selected threshold. The resulting SDIF file is imported into the BACH library for Max/MSP that allows for pitch interpretation of spectral data. As BACH is a state-of-the-art tool in assisted composition, it shares with SonaGraph the same purpose and intended users. Four mono files have been taken into account. With the aim of comparing the two systems, samples have been chosen to represent spectral configurations with different features in relation to different acoustic situations (harmonic/inharmonic, monophonic/polyphonic, music/environment). All sound files have been normalized previously to the analysis processes. The analysis by SonaGraph has been performed according to the previous discussion, with a sample rate $= 50$ Hz and by selecting all amplitudes > -30 dB. Such a sample rate results in a very high tempo $= 750$ bpm, as the rationale in this case (rather than in Figure 13) is not to provide a performance indication but to capture spectral transformation. In BACH, a time quantization is introduced, so that it matches the SonaGraph's one for sake of comparison, both in terms of tempo, note value quantization (16th) and meter $(4/4)$. BACH's pitch resolution has also been constrained to half-tones (standard MIDI notes). As it is possible to export from BACH music notation as LilyPond code, results can be easily compared (even if BACH export uses a single treble clef, while SonaGraph a piano staff). Figures 9-12 show the transcriptions for four audio samples. Amplitude threshold for SonaGraph example is kept at -30 dB, while Audiosculpt/BACH threshold is adjusted so to provide comparable examples. As apparent from the examples, BACH and SonaGraph makes use of two opposite enharmonic transcription strategies, respectively assigning $\flat$ and $\sharp$ alterations. Temporal misalignment in terms of notation results from implementation details, depending on a fixed offset at initialization, and, if relevant, is indicated by a dashed line.

Results are substantially the same if a harmonic spectrum is taken into account, as in the case of a melodic trumpet phrase (Figure 9). Figure 10 shows the transcription of a sample from a wind turbine, presenting some harmonic components over a very noisy background. In this case, while generally coherent, transcriptions have proven to strongly depend on amplitude threshold. The SonaGraph's one (bottom) is strongly sensitive to some higher frequency components that characterize the sound attack. They can be revealed in the Audiosculpt analysis by lowering the threshold, but many other frequencies then become relevant for transcription. This clearly shows that the two analysis model have a different sensitivity. The same situation applies to Figure 11, a transcription from an excerpt from dense orchestral sound (a tutti in *ff* from Varèse's *Octandre*). While the overall material is approximatively the same, sensitivity to amplitudes varies between the two analysis. A general difficult case for spectral analysis is related to noisy sounds. Figure 12 shows a transcription from a coin tossed on a hard surface, with no clear harmonic content. While both transcriptions capture a generic energy accumulation in the same higher frequency region, details in terms of pitch sensibly vary.

In conclusion, transcriptions in both environments are substantially coherent, sometimes revealing more or less clearly various perceptual details, as a result of the different sensitivities to amplitude. While the BACH system is generally more flexible, the automatic notation, relying on FFT, is generated by a hidden process, not directly accessible to the musician. On the other side, SonaGraph data, while simplified, are directly mapped into notation, and their manipulation by the user can be transparently observed into it.

7. MUSICAL EXAMPLES

A first straightforward musical application of the SonaGraph framework has occurred in the piece/installation *Orologio da rote* [6] . The piece is a reflection on signals broadcast in the Italian mediascape and includes a set of music quotations from historical jingles from RAI, the Italian national public broadcast service. It is scored for 3 modified radios and an automated piano, in particular a Yamaha Disklavier that can be driven by MIDI messages. Once collected from various sources, both acoustic signals and music jingles have been analyzed via SonaGraph and the resulting spectral data stored. During the performance, data are converted into MIDI, and MIDI messages sent in real-time to

[5] http://anasynth.ircam.fr/home/english/software/audiosculpt

[6] https://soundcloud.com/vanderaalle/orologio-da-rote

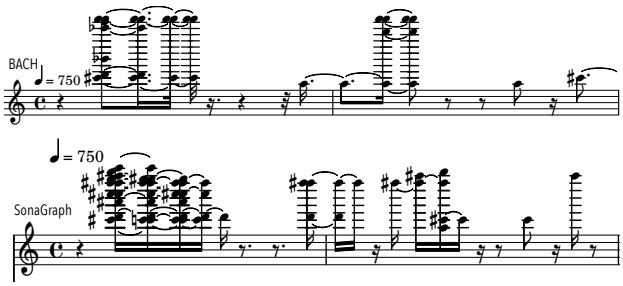

Figure 12. Music notation transcription for "coin".

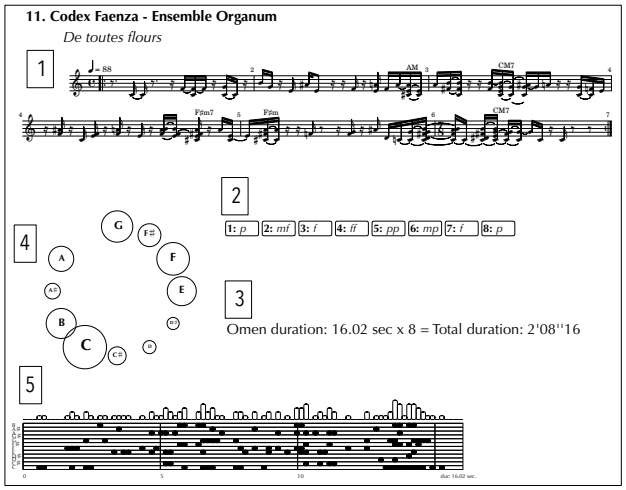

Figure 13. Omen notation for *Haruspex cledonomans*, 11.

the Disklavier (with amplitudes mapped onto velocities). Most of the piano material is thus a sort of spectral (and ghost-like) reconstruction of existing sounds, not dissimilarly from Peter Ablinger's Speaking piano [7] .

A second project has involved SonaGraph in composition and heavily relies on music information retrieved from gathered data. The work *Haruspex cledonomans* is a collection of "ominous formulas" for improvisation written for *ad libitum* improvisers [8] . Each piece of the collection is a formula that describes information for two logical music layers, the "omen", written in standard music notation, and the "prophecy", to be constructed/improvised while the omen is playing. Omens originate from 43 short audio phrases extracted from recordings of various composers/musicians, including classic, jazz, rock and ethnic sources. All the fragments have been analyzed via SonaGraph. In each analysis, the sample rate has been matched by ear to be approximatively synchronized to music pulse. Then, the spectral data have been automatically transcribed into musical notation, disregarding octaves so that only pitch classes are present (i.e. chroma). Omens are intended as the background layer for improvisation, and in each piece information extracted from the sonogram is provided to the musician(s) as a guide for the "prophecy" improvisation. As a consequence, some basic MIR techniques have been applied to SonaGraph analysis data. Figure 13 shows one of the omens. The whole notation is generated automatically from processed spectral data by means of LilyPond (as described before) and the Python-based Nodebox graphic environment [9] , with Python code scripted from SuperCollider. In Figure 13, the five blocks provide various information for the improviser. While blocks 2 and 3 depend on different compositional parameters (here not relevant), blocks 1, 4 and 5 are directly generated from SonaGraph data. Block 1 provides the omen notation in CPN (as discussed before). While the omen has to be played by the background musician, notation also includes a possible chord interpretation of note clusters (i.e. simultaneous collections of more than 2 notes), intended as a guide for the improviser. An analysis step is performed by a specialized class, not yet included in the framework core, ChordAnalyzer. Chords are specified in a template list collecting abstract chord structures (e.g. a major chord is indicated as $\{0,4,7\}$). Templates have been compiled from various music theory sources. If a cluster is found and if it matches a certain chord structure, then the relevant chord indication is written on top of the staff. Chord analysis works enharmonically and disregards chord position. Block 4 ("Pitch class relevance") displays the chroma set with symbolic names, varying the font and the circle sizes proportionally to the amount of occurrences of each pitch class in the sonogram. It thus indicates to the improviser possible pivot notes to be taken into account. Finally, block 5 is a visualization of the omen as a piano roll. It allows to quickly understand chroma distribution over time. Time is proportional to duration, so that the piano rolls of the various ominous formulas are scaled proportionally to their absolute durations. On top, a histogram provides an overall indication of the number of pitches for that time unit ("amount"), as a general density information.

8. CONCLUSIONS

The SonaGraph framework proposes a cartoonified (i.e. simplified but effective) model for spectral analysis oriented toward computer-assisted and algorithmic composition. Geared toward symbolic applications, it extracts spectral information that, while strongly reduced, is still adequate perceptually. Such a reduction makes the model efficient in terms of data storage and manipulation, and suitable for real-time usage. Thus, it can be easily integrated into a pipeline connecting sound to music application, in terms of symbolic representation (music notation and MIDI). In short, while sketchy, the gathered data are transparent to music manipulation (sound object level) rather than to sound (audio level). The SuperCollider code of the actual implementation (still in progress) is available on GitHub [10] and includes help files with examples.

Acknowledgments

The author is grateful to Luca Morino for providing examples in BACH in relation to section 6.

[7] https://ablinger.mur.at/speaking_piano.html

[8] https://soundcloud.com/vanderaalle/sets/haruspex-cledonomans

[9] https://www.nodebox.net/code/index.php/Home

[10] https://github.com/vanderaalle/vanderaalleSC/tree/master/sonaGraph

9. REFERENCES

[1] C. Roads, *Composing electronic music. A new Aesthetic*. Oxford: Oxford University Press, 2015.

[2] E. R. Miranda, *Computer Sound Design*. Oxford: Focal Press, 2001.

[3] S. Canazza, G. De Poli, G. Antonio Mian, and A. Scarpa, "Real time comparison of audio restoration methods based on short time spectral attenuation," in *Proceedings of the COST G-6 Conference on Digital Audio Effects (DAFX-01)*, Limerick, 01 2001, pp. 1–4.

[4] J. Bresson, "Sound Processing in OpenMusic," in *Proc. of the 9th Int. Conference on Digital Audio Effects (DAFx-06)*, Montreal, 2006, pp. 325–330.

[5] A. Agostini, É. Daubresse, and D. Ghisi, "Cage: a High-level Library for Real-time Computer-aided Composition," in *Proceedings ICMC—SMC—2014*, A. Georgaki and G. Kouroupetroglou, Eds., Athens, 2014, pp. 308–313.

[6] M. Müller, *Fundamentals of Music Processing. Audio, Analysis, Algorithms, Applications*. Cham: Springer, 2015.

[7] P. Schaeffer, *Traité des objets musicaux*. Paris: Seuil, 1966.

[8] C. Roads, *The Computer Music Tutorial*. Cambridge, MA, USA: MIT Press, 1996.

[9] C. Cannam, C. Landone, M. Sandler, and J. Bello, "The sonic visualiser: A visualisation platform for semantic descriptors from musical signals," in *ISMIR 2006 - 7th International Conference on Music Information Retrieval*, 2006, pp. 324–327.

[10] D. Rocchesso and F. Fontana, Eds., *The Sounding Object*. Firenze: Edizioni di Mondo Estremo, 2003.

[11] A. Klapuri and M. Davy, Eds., *Signal Processing Methods for Music Transcription*. New York: Springer, 2006.

[12] A. Agostini and D. Ghisi, "A Max Library for Musical Notation and Computer-Aided Composition," *Computer Music Journal*, vol. 39, no. 2, pp. 11–27, 2015.

[13] A. Schneider, *Sound-Perception-Performance*, ser. Current Research in Systematic Musicology. Cham: Springer, 2013, ch. Change and Continuity in Sound Analysis: A Review of Concepts in Regard to Musical Acoustics, Music Perception, and Transcription, pp. 71–111.

[14] P. N. Lehner, *Handbook of Ethological Methods*. Cambridge: Cambridge University Press, 1996.

[15] R. Jakobson, C. M. Fant, and M. Halle, *Preliminaries to Speech Analysis. The Distinctive Features and their Correlates*. Cambridge, Mass.: The MIT Press, 1952.

[16] C. M. Fant, "Historical Notes," *TMH-QPSR*, vol. 47, pp. 9–19, 2005.

[17] É. Leipp, *Musique et acoustique*. Paris: Masson, 1971.

[18] A. Orcalli, *Fenomenologia della musica sperimentale*. Potenza: Sonus, 1993.

[19] C. Regnault, "From quantitative to qualitative. the pertinence of sonographic representation for soundscape analysis," in *Proceedings of inter.noise 2000*, 2000, pp. 1–5.

[20] F.-B. Mâche, *Musique, mythe, nature ou Les dauphins d'Arion*. Paris: Klincksieck, 1983.

[21] J. T. Marshall, "Voice in communication and relationship among brown towhees," *The Condor*, vol. 66, no. 5, pp. 345–356, 1964.

[22] H. Dudley, "The Carrier Nature of Speech," *The Bell System Technical Journal*, vol. XIX, no. 4, pp. 495–515, 1940.

[23] M. Dolson, "The Phase Vocoder: A Tutorial," *Computer Music Journal*, vol. 10, no. 4, pp. 14–27, 1986.

[24] M. Müller, *Information Retrieval for Music and Motion*. Heidelberg: Springer, 2007.

[25] C. Schörkhuber and A. Klapuri, "Constant-Q Transform Toolbox for Music Processing," in *Proceedings of 7th Sound and Music Computing Conference*, X. Serra, Ed. Barcelona: SMC, 2010.

[26] W. Sethares, *Tuning, Timbre, Spectrum, Scale*. London: Springer, 2005.

[27] S. Wilson, D. Cottle, and N. Collins, Eds., *The SuperCollider Book*. Cambridge, Mass.: The MIT Press, 2011.

[28] A. Valle, *Introduction to SuperCollider*. Berlin: Logos, 2016.

[29] P. McLeod and G. Wyvill, "A smarter way to find pitch," in *Proceeding of the 2005 International Computer Music Conference*, X. Serra, Ed., Barcelona, 2005, pp. 138–141.

[30] H.-W. Nienhuys and J. Nieuwenhuizen, "LilyPond, a system for music engraving," in *Proceeding of the XIV CIM 2003*, Firenze, 2003, pp. 167–172.

[31] D. Byrd, "Music notation software and intelligence," *Computer Music Journal*, vol. 18, no. 1, pp. 17–20, 1994.

[32] A. Valle, "Integrated Algorithmic Composition. Fluid Systems for including notation in music composition cycle," in *NIME 2008: Proceedings*, 2008, pp. 253–256.

[33] Adobe, *PostScript Language Reference*, 3rd ed. Reading, Mass.: Addison-Wesley, 1999.