THE SNAIL: A REAL-TIME SOFTWARE APPLICATION TO VISUALIZE SOUNDS

Thomas Hélie

S3AM team, STMS, IRCAM-CNRS-UPMC
1 place Igor Stravinsky, 75004 Paris, France
thomas.helie@ircam.fr

Charles Picasso

Analysis/Synthesis team, STMS, IRCAM-CNRS-UPMC
1 place Igor Stravinsky, 75004 Paris, France
Charles.Picasso@ircam.fr

ABSTRACT

The Snail is a real-time software application that offers possibilities for visualizing sounds and music, for tuning musical instruments, for working on pitch intonation, etc. It incorporates an original spectral analysis technology (patent-pending) combined with a display on a spiral representation: the center corresponds to the lowest frequencies, the outside to the highest frequencies, and each turn corresponds to one octave so that tones are organized with respect to angles. The spectrum magnitude is displayed according to perceptive features, in a redundant way: the loudness is mapped to both the line thickness and its brightness. However, because of the time-frequency uncertainty principle, using the Fourier spectrum (or also Q-transform, wavelets, etc) does not lead to a sufficient accuracy to be used in a musical context. The spectral analysis is completed by frequency precision enhancer based on a post-processing of the demodulated phase of the spectrum. This paper presents the scientific principles, some technical aspects of the software development and the main display modes with examples of use cases.

1. INTRODUCTION

Spiral representation of the audio spectrum allows the combination of scientific signal processing techniques and of a geometric organization of frequencies according to chroma and octaves. Compared to spectro-temporal representations, it offers a complementary or alternative solution that is natural for musical applications. For this reason, a piece of software or hardware applications have been developed (see [1, 2, 3] and [4] for a review). Scattering methods based on spiral geometries have also been proposed, with applications in audio classification, blind source separation, transcription or other processing tasks [5]. Other methods exploiting circular geometries through the use of chroma have been designed for several automatic musical analysis tasks (see e.g. [6, 7]). Such methods are efficient for music information retrieval and its applications. For the musicians and for musical or audio communities, visualizing raw data under such natural geometries (without any decision process) is also interesting: it allows humans to monitor their actions through a direct feedback, with potential human-learning issues if the perceptible feedback is accurate enough.

This paper presents a real-time application, *The Snail*, that gather several properties to provide an intuitive and accurate rendering:

(P1) Spiral abacus. One chroma is one angle and one round is one octave;

(P2) Perceptual simple[1] mapping. Twice louder is an audio stimulus, twice visible is the graphic symbol (with redundancy: twice brighter, twice larger);

(P3) Frequency accuracy and stationarity. The frequency accuracy can be adjusted to enhance or select frequency components (partials) according to a targeted tolerancy, for: (a) instrument tuning tasks, or (b) musical interpretation (glissando, vibrato, orchestral mass effect, etc) and training.

For a tuning task, some "high accuracy" (that can still be controlled by musicians) can require about[2] a 2Hz-precision (voice and wind instruments), a 1Hz precision (string instruments) or much lower (0.1Hz) for analogue synthetizers in order to control slow beatings between several oscillators. To visualize musical interpretations, or in a musical training context, such an accuracy can be relaxed (typically from 4Hz to 10Hz) because of the vibrato, the mass effect (non synchronized signals when several instrumentists play the same notes), the pitch contour, etc. The software application has been designed to handle properties (P1-P3) and to propose solutions that cover such musical contexts.

The paper is organized as follows. Section 2 provides some recalls on the motivation and the problem statement. Section 3 presents the scientific principles used in the application. Section 4 addresses the software development, including the user interface design, the software structure and platforms. Finally, section 5 gives some conclusions and perspectives.

2. MOTIVATION AND PROBLEM STATEMENT

2.1. Motivation

Our very first motivation, at the basis of the Snail development, appeared in 2012: it simply consisted of representing the spectrum of sounds in a complete way (with magnitude and phase) on an abacus that organizes frequencies f (in Hertz) with respect to angles θ (in radians) according to chroma. This corresponds to the mapping[3]

$$\theta(f) = 2\pi\ \log_2(f/f^\star), \tag{1}$$

on a typical audio frequency range $f \in [f^-, f^+]$ with $f^- = 20 > 0$ Hz and $f^+ = 20$ kHz, and where the tuning reference frequency (typically, $f^\star = 440$ Hz) is mapped to angle 0. To have a bijective mapping between frequencies and such a chroma organization, angles θ must be complete by an octave information, under a 3D form (see [8, Fig. 8, p.105] and the so-called spiral array in [9, p.46]) or a spiral 2D form. A simple choice[4] is to map the radius ρ such

[1] Only the loudness of the spectrum is considered. Masking or dynamic loudness modeling are not taken in account in this study.

[2] The musical values are usually measured in cents. Here, they are given in Hertz, typically for a reference note at frequency 440Hz.

[3] This formula provides a counter-clockwise winding, and its negative version a clockwise wending.

[4] Other conventions can be chosen. In particular, preserving the length of the frequency axis yields an analytic expression for ρ. However, for more than 2 octaves, this choice makes the low-frequency range too small for visualization in practice.

that it is increased by a unit value at each rising octave, as

$$\rho(f) = 1 + \log_2(f/f^-). \quad (2)$$

The goal was to bring together standard tools of signal processing (Fourier [10], or also constant-Q [11], wavelet analysis [12], etc) and a natural musical representation of frequencies, in order to explore its applicative interests in a real-time context. A first real-time tool were built, based on a Fourier analysis (see figure 1) and tested.

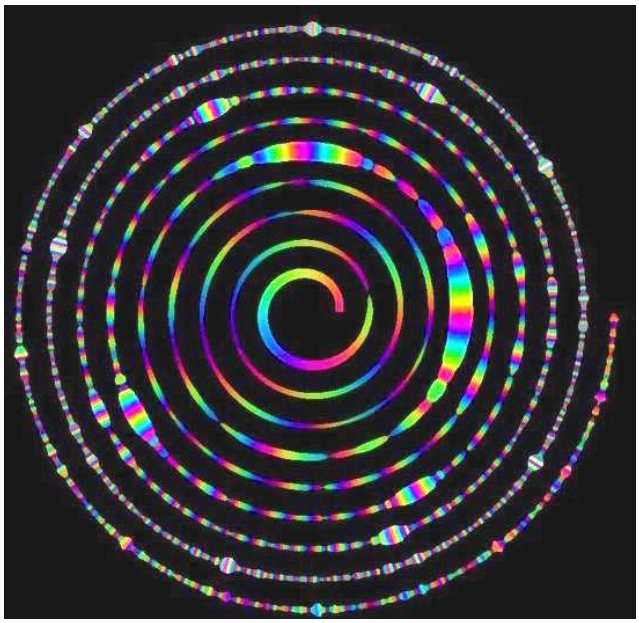

Figure 1: *Basic representation of the spectrum on a spiral abacus (figure 3.18 extracted from [13]): the signal is composed of a collection of pure sines, analyzed with a Hanning window (duration* $50\,ms$*). The line thickness is proportional to the magnitude (dB scale on a 100dB range), the color corresponds to the phase (in its demodulated version to slow down the color time-variation, without information loss, and with a circular colormap to avoid jumps between* 2π *rd and* 0 *rd).*

Its practical use on basic (monophonic or poor) musical signals proves to be attractive but the separation and the frequency location of partials are not accurate enough (also using contant-Q or wavelet transforms) to be used in a musical context.

2.2. Problem statement

To cope with this separation and accuracy difficulties, reassignment methods are available [14, 15] as well as methods based on signal derivatives [16, 17] (see also [18] for a comparison). These methods allows the estimation of frequencies from spectrum information, including for partials with locally time-varying frequencies.

To address property (P3), a basic method is proposed, that does not use frequency estimation. It consists of applying a contracting contrast factor to the spectrum magnitude (no reassignment). This factor is designed to weaken the energetic parts for which the phase time-evolution does not match with that of the bin frequency, according to a targeted tolerance (see § 3). In short, the method can be illustrated by the following analogy: the idea is similar to applying a stroboscope to the rotating phase of each bin, at the bin frequency (phase demodulation), and to select the magnitude of the sufficiently slow rotations. This approach has some relevant interests for the visualizer application:

- the targeted accuracy is consistant with musical applications and can be adjusted independently from the analysis window duration;
- it is robust to noisy environnement since the most unstationary is a component, the cleaner is the output;
- in particular, for tuning tasks, a sustained long note played by an instrumentist can be significantly enhanced compared to fast notes (played by some neighbors before a repetition), by using a very selective threshold (1 Hz), while the tuning accuracy is very high.

3. SCIENTIFIC PRINCIPLE

The Snail is composed of two modules [19]: (A) a sound analyzer, (B) a display.

3.1. Analyzer

The analyzer takes as input a monophonic sound, sampled at a frequency F_s. It delivers four outputs to be used in the visualizer: (1) a tuned frequency grid (Freq_v) and, for each frame n, (2) the associated spectrum magnitudes (Amp_v) and (3) demodulated phases (PhiDem_v), (4) a "phase constancy" index (PhiCstcy_v). Its structure is described in figure 2. It is decomposed into 7 basic blocks, labelled (A0) to (A6), *see figure 2*.

Blocks (A0-A1) are composed of a gain and a Short Time Fourier Transform with a standard window (Hanning, Blakman, etc) of duration T (typically, about 50 ms) and overlapped frames. The initial time of frame n is $t_n = n\tau$ ($n \geq 0$) where $\tau < T$ is the time step.

Blocks (A2-A3) process interpolations. A frequency grid (block A2) with frequencies

$$\text{Freq\_v}(k) = f^\star . 2^{(m_k - m^\star)/12}$$

is built according to a (rational) uniformly-spaced midicode grid [20]

$$m_k = m^- + (k/K)(m^+ - m^-), \quad 0 \leq k \leq K,$$

where $m^\star = 69$ is the midi code of the reference note (A4), m^- is that of the lowest note, m^+ that of the highest one, and where $f^\star$ denotes the reference tuning frequency (typically, $f^\star = 440$ Hz). In the Snail application, K is chosen to have 50 points between each semi-tones, providing a resolution of 1 cent.
For each frame n, block (A3) builds the spectrum complex magnitude S_k associated with the frequency grid $\{f_k\}_{0\leq k\leq K}$ based on an interpolation method (e.g. affine).

Block (A4) computes the modulus Amp_v(k) and the phases φ_k from the complex magnitudes delivered by block (A3).

Block (A5) computes the demodulated phases

$$\text{PhiDem\_v}(k) = \varphi_k - 2\pi\,\text{Freq\_v}(k)t_n.$$

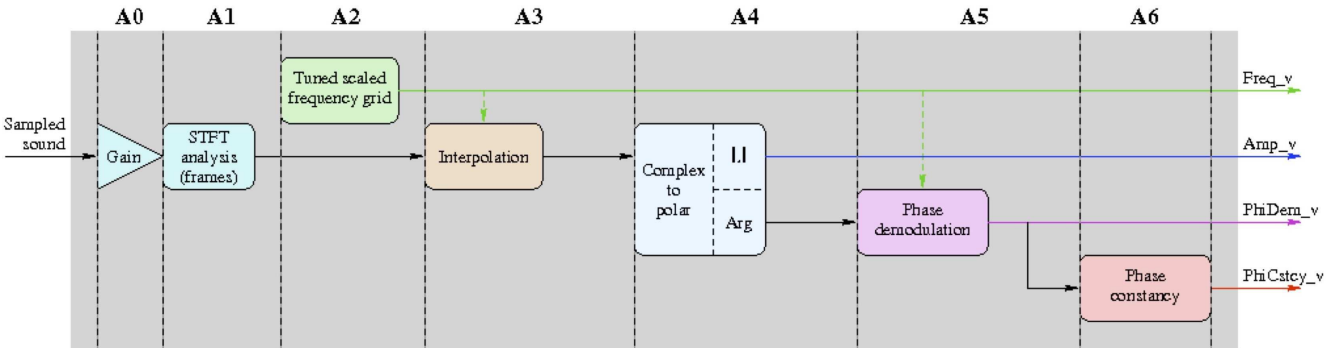

Figure 2: *Block Diagram of the Analyzer*

Block (A6) delivers a phase constancy index as follows. First, the demodulated phases PhiDem_v(k) are converted into a complex value on the unit circle $z_k = \exp\big(i\,\text{PhiDem\_v}(k)\big)$. Second, for each k, independently, this complex value feeds (at each frame) a digital low pass filter (typically, a maximally-flat Butterworth filter), with cutoff frequency F_c, at sampling frequency $1/\tau$. Third, the phase constancy index PhiCstcy_v(k) is computed as the squared modulus of the filter output. Consequently, if the demodulated phase rotates less rapidly than F_c revolutions per second, the phase constancy index is close to 1. If the rotation speed is faster, the index is close to 0.
In short, the phase constancy index provides a quasi-unit factor for the bins for which the spectrum phase is consistently synchronized with the bin frequency, up to a deviation of $\pm F_c$. It provides a quasi-zero factor outside this synchronization tolerance. Its effect is illustrated in figures 3-5, as detailed below.

3.2. Visualizer and illustration of some stages of the analyzer

The visualizer builds colored thick lines to represent the spectral activated zones on the spiral abacus. First, the magnitudes Amp_v(k) are converted into loudness L_v(k), according to the ISO226 norm [21]. Second, the line thickness is built as the product of the loudness L_v(k) and the phase constancy index PhiCstcy_v(k). Third, the color is built. The loudness L_v(k) is mapped to the brightness. The hue and saturation are built according to several modes:

Magnitude: the hue and saturation are built as a function of the loudness L_v(k);

Phase: they are built as a function of PhiDem_v(k), according to a circular colormap (see figure 1);

Phase constancy: they are built as a function of PhiCstcy_v(k) so that the color indicates the quality of the phase synchronization.

An illustration of the accuracy improvement that results from the analyzer is given in figures 3-5 for a C major chord (C-E-G) played on a Fender-Rhodes piano. These figures describe several intermediate stages of the analysis process. In figure 3, the color corresponds to the phase mode and the thickness corresponds to the loudness, without taking into account the correction by the phase constancy index PhiCstcy_v: the accuracy is the same as in figure 1. Figure 4 is the same as figure 3 except that (only for the illustration) the phase constancy index is mapped to the color saturation. The saturated parts correspond the synchronized parts (here, up to $\pm F_c$ with $F_c = 2\,$Hz) whereas the grey parts are the parts to reject. Finally, figure 5 provides the final result, in which the line thickness is multiplied by the phase constancy index: accordingly, the grey parts of figure 4 have disappeared.

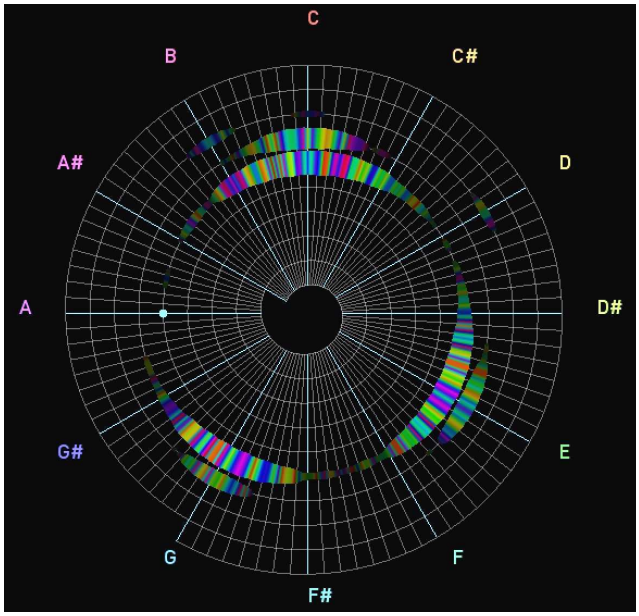

Figure 3: *C major chord (Fender-Rhodes piano): the line thickness depends on the loudness but is not corrected by the phase constancy index; the color corresponds to the demodulated phases.*

This decomposition into stages show how the method transforms large lobes (Fourier standard analysis, in figure 3) into an sharp ones (extraction of the demodulated phases with slow evolution, in figure 5). We observe the marked presence of the fundamental components (harmonics 1) and the harmonics 2 of each note. The point in cyan indicates the tuning fork (here, 440Hz).

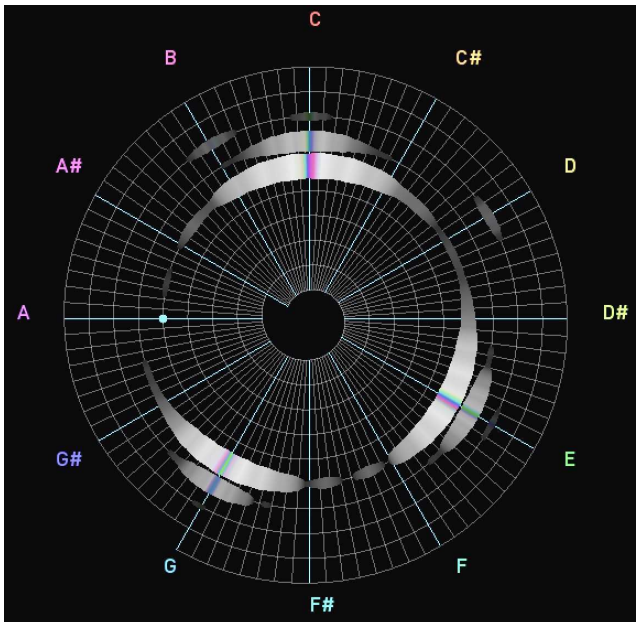

Figure 4: *C major chord (Fender-Rhodes piano): the line thickness depends on the loudness but is not corrected by the phase constancy index; the color saturation corresponds to the phase constancy index. The grey parts are those to reject.*

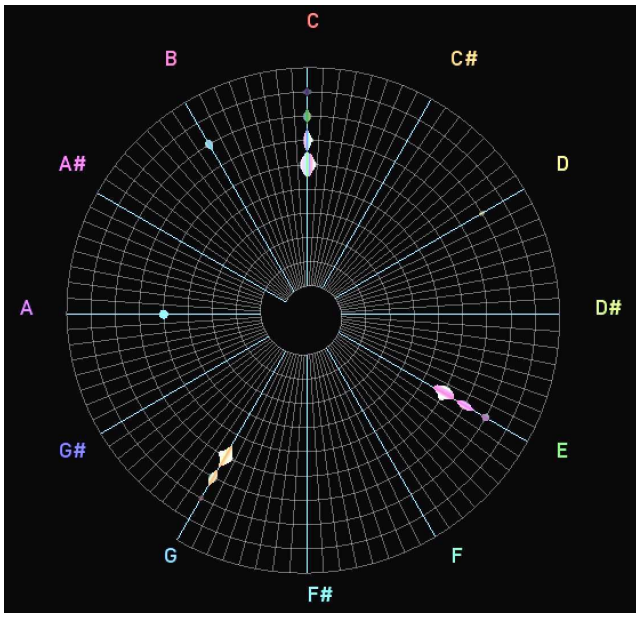

Figure 5: *C major chord (Fender-Rhodes piano): the line thickness depends on the loudness and is corrected by the phase constancy index. The grey parts of figure 4 are rejected.*

4. SOFTWARE DEVELOPMENT

4.1. User Interface

The Snail user interface (figure 6), designed by IRCAM and the Upmitt design studio (Paris, France, [22]), is composed of five parts:

1. The main view, which can be split into two views, that allows a secondary analysis display in the same space.
2. A global menu for the basic application operations and audio I/O configuration.
3. A side bar to access and change the most used display and engine properties.
4. An Advanced Settings (sliding) panel with more options to finely configure the analysis engine, the properties of the snail spiral abacus and the sonogram (other options are available but not detailed here).
5. (standalone only) A sound file player with a waveform display to feed the real-time audio engine.

Built around the main view, the interface is configured at startup to show the Snail spiral abacus. This abacus is built using the equal temperament but is not related to the engine. It is just used as a grid to help the eye and could be easily substituted by another grid type. Two additional representations of the analysis may also be displayed, either separately or simultaneously to the spiral abacus:

1. the real-time sonogram view, rendering the snail analysis over time. This sonogram may be used in standard (figure 7) or precise mode (figure 8), the latter exactly leading to the same accurate thickness as on the spiral abacus.
2. the tuner view, showing a rectified zoomed region of the spiral (figure 9) and aimed at accurately tuning an instrument when the Snail engine is setup in a Tuner Mode for high precision (figure 10).

Two modes are available in the side bar panel as convenient pre-configurations of the snail engine for two different purposes :

1. the Music mode, aimed at musical visualization, presents a relaxed analysis suitable for the visual tracking of more evolving sounds, like a polyphonic piece of music.
2. The Tuner Mode, a more refined analysis configuration, aimed at the precise visualization of stationary components and the tuning of instruments.

In addition, in order to reflect visually the demodulated phase, an hexagonal spinning shape is drawn above the Tuner View (figure 10) or at the center of the spiral (Snail view). Its angular speed and its color changing both indicate "how close" or "how far" the frequency is from the selected target frequency.

For the user convenience, a "F0 detection" activation switch is also available in the side bar. When set in a Tuner mode configuration, the interface centers the tuner view to the detected fundamental frequency.

Other properties available in the application are the Tuning Reference Frequency, Grid range for the abacus, Visual gain to enhance visually the input signal, and the various color modes, that allows the user to plot only specific parts of the analyzed signal, like the magnitude or the demodulated phase.

For the future version, we plan to integrate the sharable "scala" scale file format for users who want to create and use customized grids based on a tonality different than the equal temperament.

Figure 6: *The Snail User Interface (IRCAM/Upmitt). The Settings panel (normally hidden at startup) is shown visible here (users can switch its visibility on/off).*

Figure 8: *Sonogram in its precise mode: compared to figure 7, only the (colored) refined part are displayed. This mode exactly mirrors the visual representation in the snail display.*

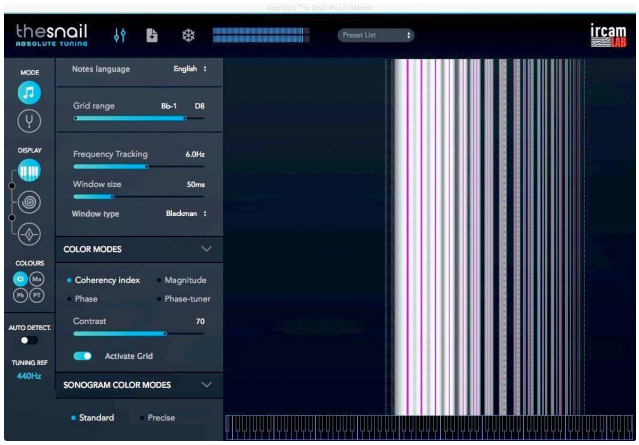

Figure 7: *Sonogram in its standard fft-mode: the brightness is related to the energy (Loudness scale). In this display mode, the central parts of the thick lines are colored according the frequency precision refinement based on the demodulated phases.*

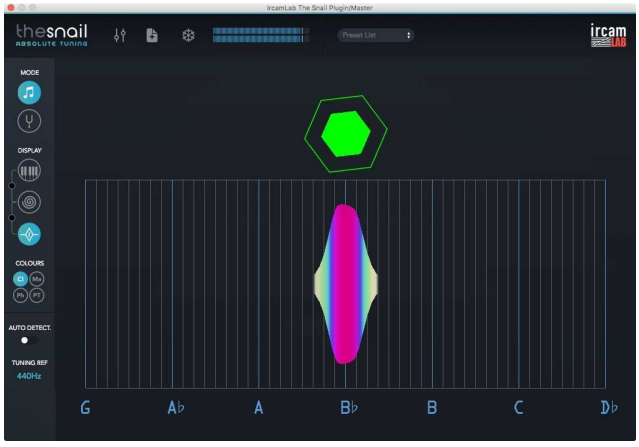

Figure 9: *The Snail tuner view, showing the rectified analysis region with the hexagonal shape on top of it. The Music mode analysis is "on" so that the frequency lobe has not the sharpest size, indicating a more relaxed analysis.*

4.2. Software structure

The Snail real-time application workload is decomposed into:

An Analysis Task performed most of the time directly in the real-time audio thread,

A Visualisation Task usually performed in the main application thread: it may be split, with a dedicated rendering thread for the OpenGL specific drawings depending on the development environment/framework (e.g. the JUCE Framework [23]). We leave it as an internal implementation detail.

The Analysis process (see figure 11ⓐ) is in charge of all the required treatments for the signal spectral analysis, including the production of the demodulated phase and the phase constancy index.

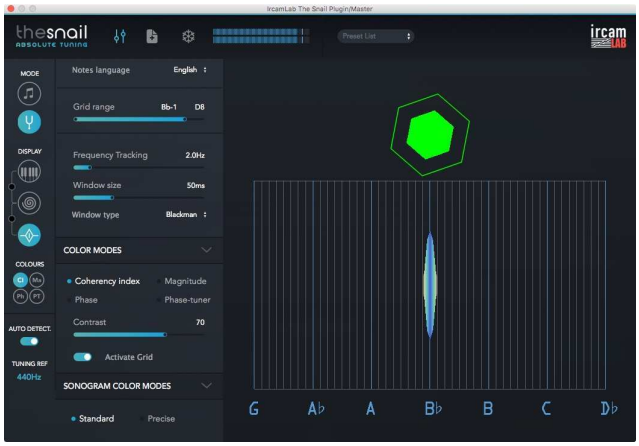

Figure 10: *Same sound and component but with the Tuner mode "on" for a more refined analysis: the size the lobe is sharper and is more adapted to a tuning application). The green color of the spinner still indicates that the current frequency is "in tune".*

The Display process (see figure 11ⓑ) is responsible for the conversion of the spectral output frames to the appropriate geometry and colors for the final displays (sonogram and snail), both rendered using OpenGL [24].

In order to communicate the analysis frames produced to the display task/thread in real-time, a shared FIFO queue (implemented as a Lock-Free FIFO, see figure 11ⓒ) is used and previously allocated with a sufficient amount of space to store the minimal amount of required frames expected for a fluid communication.

4.3. Platforms

The first prototype (research application) of The Snail was developed using the OpenFrameworks library [25] for both standalone application and a first mobile version (thanks to the OpenFrameworks iOS addon [26]).

The final application has been then converted and developed under the JUCE framework [23] in order to simplify the deployment in a standalone and several plugin formats. It is now released as a Standalone application and various plugin formats including VST [27], AudioUnit [28] and AAX [29] for both Mac and PC.

An iOS version [30] (iPhone only for now) is also available, which only offers the tuner display mode (the sonogram is not available as we restricted the mobile usage to a tuner only).

4.4. Examples of use cases

From the tuner perspective, the snail may serve as a high precision tool, with a configurable visual representation of the musical grid. The musician can tune his instrument as a usual tuner, but he may also use it on sounds without a clear pitch, like bells or inharmonic sounds. As the engine does not interpret the incoming sound and does not need the F0 information to adapt its grid on (although it is still possible to do it in the software), the user can focus on particular note (or frequency) and decide accordingly then how to "tune" its sound (let it be on the second harmonic if we wish too). No assumptions are made on how he should proceed, but everything relies on the interpretation of the precise visual feedback. That, by nature, will extend its usability and won't restrict "The Snail" to a specific set of instruments or sounds.

From a visualizer perspective, the analysis of the snail may also serve the musician, the sound engineer or even the *sound enthusiast* to see how the sound is structured on a musically-relevant abacus. The singer can visualize the harmonics, produced in real-time. The musician can see how his interpretation may clearly affect the produced timbre, still readable at a "note level" too. An interaction takes place as the user changes its "sound production" approach using the visual feedback given by the tool. As opposed to the spectrogram, which may be more relevant to see the global spectral balance of a sound, but which is not appropriate to spot a very specific note, "The Snail" is precise enough to make the user understand what is happening from a "note" perspective and so to spot them.

5. CONCLUSIONS

This paper has presented a real-time application that displays spectral information on a spiral representation such that tones are organized with respect to angles. To reach a frequency accuracy that is usable in a musical context (tuning tasks, work on intonation by instrumentists or singers, etc), the novelty is to complement the standard Fourier analysis by a process before displaying the spectral information. This process applies a contracting contrast factor on the magnitude, that only selects the bins for which the spectrum phase rotates at the bin frequency, in an adjustable tolerance range. Typically, if the maximal tolerance frequency deviation $\pm F_c$ is such that $F_c < 2$ Hz, this results in a very precise tool for tuning tasks, that is robust in noisy environment (non stationary partials being rejected by the process). For $F_c \approx 6$ Hz, the tool is adapted to work on intonation or music visualization. The real-time application has been conceived to run on several platforms (desktop and mobile) and operating systems.

In practice, the Snail has been presented and tested in several musical contexts: (1) with the Choir of Sorbonne Universités, (2) with violinists [31], (3) with piano tuners and manufacturers [32], etc.

Based on their reactions, future possible development is taken into account:

- the modification of the abacus (temperaments, micro-tones, etc.) by the user (e.g. scala format [33]);
- representing the first harmonics in the zoom mode (see figures 9 and 10);
- implementing a new index built on the demodulated phase (included in the patent but not yet implemented) allowing the handling of vibrato, glissando or frequency variations still with a high frequency precision.

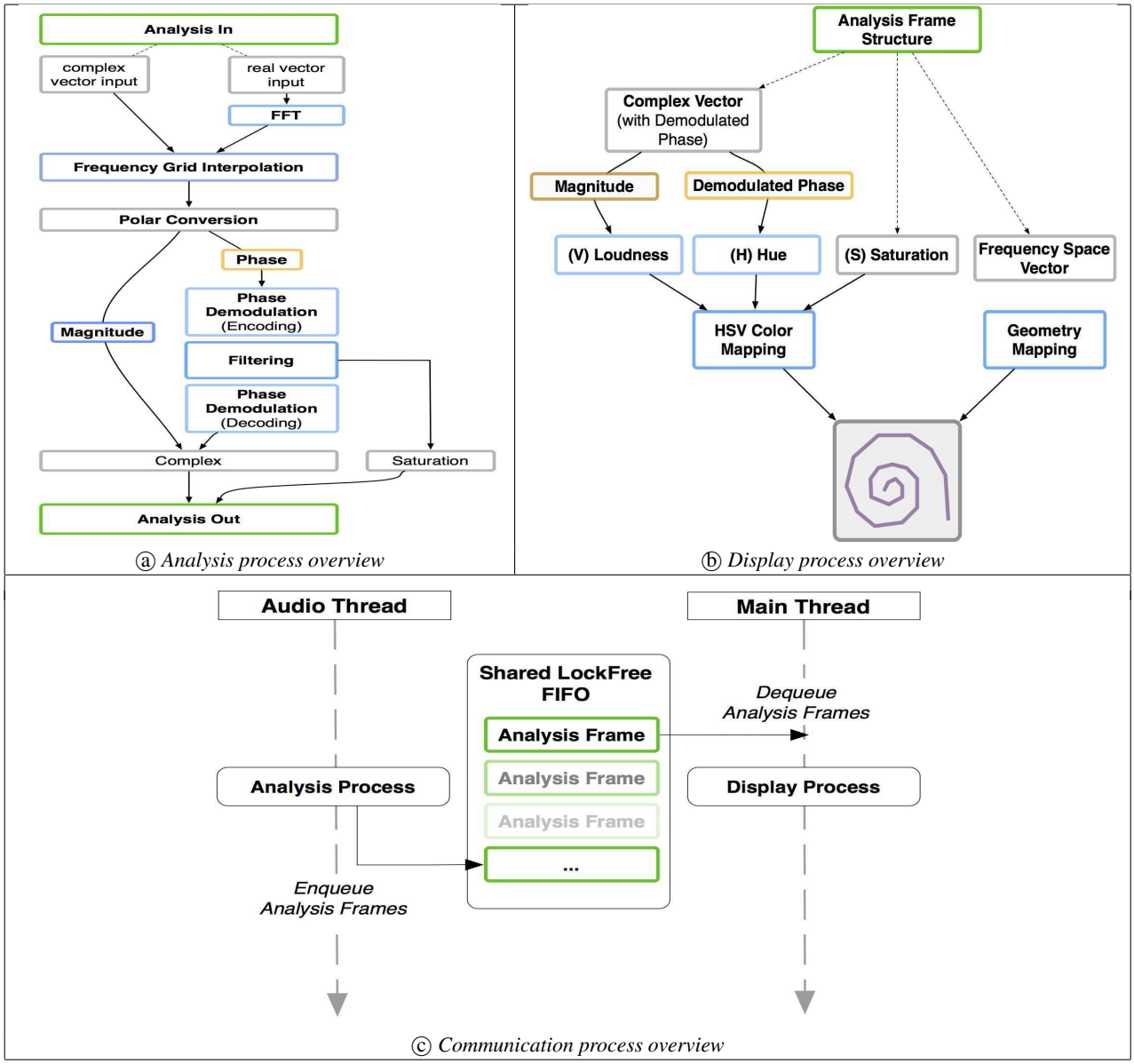

Figure 11: *Overviews of the Snail processes :* ⓐ *Analysis process,* ⓑ *Display process,* ⓒ *Communication process.*

6. REFERENCES

[1] A.G. Storaasli, "Spiral audio spectrum display system," 1992, US Patent 5,127,056.

[2] Michel Rouzic, "Spiral Software Application," http://photosounder.com/spiral/, 2013.

[3] N. Spier, "SpectratunePlus: Music Spectrogram Software," http://nastechservices.com/Spectratune.php, Access: 1 April 2016.

[4] S. D. Freeman, *Exploring visual representation of sound in computer music software through programming and composition*, Ph.D. thesis, University of Huddersfield, 2013, http://phd.samuelfreeman.me.uk.

[5] V. Lostanlen and S. Mallat, "Wavelet Scattering on the Pitch Spiral," in *International Conference on Digital Audio Effects (DAFx)*, 2015, vol. 18, pp. 429–432.

[6] G. Peeters, "Musical key estimation of audio signal based on hmm modeling of chroma vectors," in *DAFx (International Conference on Digital Audio Effects)*, Montréal, Canada, 2006.

[7] M. Mehnert, G. Gatzsche, D. Arndt, and T. Zhao, "Circular pitch space based chord analysis," in *Music Information Retrieval Exchange*, 2008, http://www.music-ir.org/mirex/2008.

[8] R. N. Shepard, *Approximation to Uniform Gradients of Generalization by Monotone Transformations of Scale*, pp. 94–110, Stanford University Press, Stanford, 1965.

[9] Elaine Chew, *Towards a Mathematical Model for Tonality*, Ph.D. thesis, Massachusetts Institute of Technology, MA, USA, 2000.

[10] L. Cohen, *Time-Frequency Analysis*, Prentice-Hall, New York, 1995, ISBN 978-0135945322.

[11] J.C. Brown and M.S. Puckette, "An efficient algorithm for the calculation of a constant Q transform," *JASA*, vol. 92, no. 5, pp. 2698–2701, 1992.

[12] S. Mallat, *A Wavelet Tour of Signal Processing*, Academic Press, 3rd edition edition, 2008.

[13] T. Hélie, *Modélisation physique d'instruments de musique et de la voix: systèmes dynamiques, problèmes directs et inverses*, Habilitation à diriger des recherches, Université Pierre et Marie Curie, 2013.

[14] F. Auger and P. Flandrin, "Improving the readability of time-frequency and time-scale representations by the reassignment method," *IEEE Transactions on Signal Processing*, vol. 43, no. 5, pp. 1068–1089, 1995, doi:10.1109/78.382394.

[15] P. Flandrin, F. Auger, and E. Chassande-Mottin, *Time-frequency reassignment: From principles to algorithms*, chapter 5, pp. 179–203, Applications in Time-Frequency Signal Processing. CRC Press, 2003.

[16] S. Marchand, "Improving Spectral Analysis Precision with an Enhanced Phase Vocoder using Signal Derivatives," in *Digital Audio Effects (DAFx)*, Barcelona, Spain, 1998, pp. 114–118.

[17] B. Hamilton and P. Depalle, "A Unified View of Non-Stationary Sinusoidal Parameter Estimation Methods Using Signal Derivatives," in *IEEE ICASSP*, Kyoto, Japan, 2012, doi: 10.1109/ICASSP.2012.6287893.

[18] B. Hamilton, P. Depalle, and S. Marchand, "Theoretical and Practical Comparisons of the Reassignment Method and the Derivative Method for the Estimation of the Frequency Slope," in *IEEE WASPAA*, New Paltz, New York, USA, 2009, pp. 345–348.

[19] T. Hélie (inventor) and CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE (assignee), "Procédé de traitement de données acoustiques correspondant à un signal enregistré," French Patent App. FR 3 022 391 A1., 2015 Dec 18,
(and Int. Patent App. WO 2015/189419 A1. 2015 Dec 17).

[20] Website, "Midi association," https://www.midi.org/specifications/item/table-1-summary-of-midi-message.

[21] Technical Committee ISO/TC 43-Acoustics, "ISO 226:2003, Acoustics - Normal equal-loudness-level contours," 2003-08.

[22] Website, "Upmitt," https://www.behance.net/upmitt.

[23] Website, "JUCE," https://www.juce.com.

[24] Website, "OpenGL," https://www.opengl.org.

[25] Website, "OpenFrameworks," http://openframeworks.cc.

[26] Website, "iOSaddon," http://ofxaddons.com/categories/2-ios.

[27] Website, "VST (Steinberg)," https://www.steinberg.net/.

[28] Website, "AudioUnit," https://developer.apple.com/reference/audiounit.

[29] Website, "AAX," http://apps.avid.com/aax-portal/.

[30] Website, "iOS10," http://www.apple.com/ios/ios-10/.

[31] T. Hélie and C. Picasso, "The snail: a new way to analyze and visualize sounds," in *Training School on "Acoustics for violin makers", COST Action FP1302*, ITEMM, Le Mans, France, 2017.

[32] T. Hélie, C. Picasso, and André Calvet, "The snail : un nouveau procédé d'analyse et de visualisation du son," *Pianistik, magazine d'Europiano France*, vol. 104, pp. 6–16, 2016.

[33] Website, "Scala Home Page," http://www.huygens-fokker.org/scala/.