# A Real-Time Cent Sensitive Strobe-like Tuning Software Based on Spectral Estimates of the Snail-Analyser

Thomas Hélie, Charles Picasso, Robert Piéchaud, Michael Jousserand, Tom Colinot

# A REAL-TIME CENT-SENSITIVE STROBE-LIKE TUNING SOFTWARE BASED ON SPECTRAL ESTIMATES OF THE SNAIL-ANALYSER

**Thomas Hélie**
*CNRS*
*STMS laboratory (IRCAM-CNRS-SU)*
Paris, France
thomas.helie@ircam.fr

**Charles Picasso**
*CNRS*
*STMS laboratory (IRCAM-CNRS-SU)*
Paris, France
charles.picasso@ircam.fr

**Robert Piéchaud**
*IRCAM*
*STMS laboratory (IRCAM-CNRS-SU)*
Paris, France
robert.piechaud@ircam.fr

**Michaël Jousserand & Tom Colinot**
*Buffet Crampon*
Mantes-La-Ville, France
[ michael.jousserand , tom.colinot ]@buffetcrampon.com

## ABSTRACT

This paper presents a real time software to tune musical instruments. The visual rendering emulates a tuner commonly used by the music industry, namely, the strobe-tuner. In the classic case, a tuned note is characterised by the immobility of a dial: through a stroboscopic effect, the rotation speed of the dial is made proportional to the deviation in Hertz between the target pitch and the played note. Here, we propose to use spectral estimates of the Snail-Analyser to have a similar rendering with respect to the deviation in cents, with an adjustable sensitivity. These estimates are derived from the demodulated phase calculated by Fourier analysis. The software allows the choice of the targeted frequencies for each note, according to musical considerations such as temperaments, musical modes, but also octave stretching, etc. It has been prototyped using the MAX software and developed with the Juce framework to target both desktop and mobile environments.

Keywords: Fourier analysis, demodulated phase, timbre, harmonic representation

## 1. INTRODUCTION

The ATRIM project [1] aims to design reliable tools with reactive visual renderings adapted to high precision pitch and timbre analysis of musical wind instruments. The objective is to provide musicians with accurate and informative feedback in real time (during performance) that is relevant in several contexts. Some of them are: expert testing and improvement of manufactured instruments, tuning or comparison of instruments, musical practice by helping musicians adjust their motor control to reach their own or a teacher's target (intonation, timbre, vibrato, glissando, playing effects, etc). To this end, the project relies on the patented technology [1] used in the "Snail-Analyser" [2] and on new designs of estimates and visual renderings in collaboration with expert musicians.

In [3], a first result of this project allowed to mimic electro-mechanical strobe-tuners [4, 5] with the same rendering and accuracy (see [3, § 3 in sec.1] for some brief descriptive and historical elements, and [6–8] for more recent non-mechanical versions). Such a tuner is composed of 12 dials (one per chroma) made of tuned rotating wheels with concentric cyclic patterns (one per octave), all enlightened according to the sound wave signal. This stroboscopic effect makes the pattern related to a sound spectral component appear motionless if their respective frequencies are equal: the pattern rotation speed is the frequency deviation. The visual rendering in [3] exploits some analysis signals derived in [1], including the spectrum demodulated phase, as an accurate robust estimate of the pattern angle. Compared to the original strobe-tuners, this technology is better suited to handling temperaments, modes, octave stretching, etc. But, it suffers from the same discomfort for high pitch tuning tasks, as the rotation speed sensitivity follows the frequency scale and not the note scale.

This paper addresses a similar issue (reactive visual renderings for high precision pitch tuning) with respect to the deviation in cents rather than in Hertz. To this end, we derive signals from the process [1], based on the spectrum demodulated phase, and use them in two visual renderings: the Snail viewer (spiral representation of the spectrum) and the strobe-like viewer.

The paper is organised as follows. Section 2 introduces the method. It includes a brief reminder and sequentially presents cent-sensitive estimates adapted to the Snail viewer and for the strobe-like viewer. Section 3 presents the software. It includes considerations on its architecture, elements on design issues and prototyping in Max. It also describes the anatomy of the Strobe-Tuner application. Section 4 ends with conclusive remarks and perspectives.

[1] ATRIM is the French acronym for "Analyseur Temps-Réel haute précision de justesse et de timbre pour Instruments Musicaux" (High precision real-time pitch and timbre analyser for musical instruments). This project is supported by the plan "France Relance" (see acknowledgements at the end of the paper).

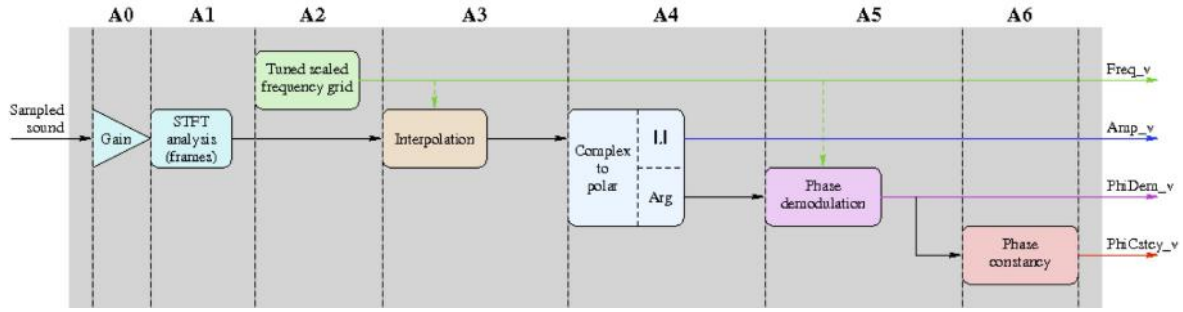



Figure 1. Block diagram of the analysis process of the Snail-Analyser.

## 2. METHOD

This section first presents in § 2.1 a brief reminder on the principle used in the Snail-Analyser [1] and the basic spectral estimates providing a frequency-scaled precision. Second, section 2.2 adapts one of the estimates to render the Snail representation precise to a cent scale. To transpose such a cent-scaled sensitivity to a strobe-like tuner, section 2.3 proposes and exploits an additional spectral estimate.

### 2.1 Reminder on the Snail-Analyser

The analysis process (see figure 1) sequentially applies to the sampled audio input signal (see [2] for more details): a gain (A0), a short time Fourier transform for successive frames (A1), the spectrum complex values of which are interpolated in block (A3) according to a vector `Freq_v` of tuned frequencies (A2), and converted into polar coordinates (spectrum modulus `Amp_v` and phase) in block (A4), followed by a process on the phase (A5-A6).

The process (A4-A5) is applied in the time domain and composed of the two following steps:

(A5) build signal `PhiDem_v` as the demodulated phase

$$\phi_d(t,f) = \phi_{Fourier}(t,f) - 2\pi f t,$$

where $\phi_{Fourier}(t,f)$ denotes the spectrum phase of the frame starting at time $t$ computed for frequency $f$ with value in `Freq_v`,

(A6) build signal `PhiCstcy_v` as the modulus of the output signal of a low-pass filter with cutoff frequency $f_c$, excited by the input complex signal

$$u_f : t \mapsto \exp\big(i\,\phi_d(t,f)\big).$$

In (A6), the unit signal $u_f$ is time invariant if the audio component locally analysed at frequency $f$ exactly oscillates at frequency $f$. Otherwise, it is rotating according to the frequency deviation between $f$ and the frequency of this audio component. As a consequence, the process (A6) returns a unit value if the frequency deviation is lower than $f_c$ and a smaller positive value otherwise. This value is used in the Snail viewer to enhance the frequency precision, which is crucial for musical applications.

The Snail viewer is composed of a spiral skeleton (+1 round from the center is +1 octave, so that one angle corresponds to one chroma): the polar coordinates $(\rho, \theta)$ of the spiral are defined with respect to frequency $f \in [f_{min}, f_{max}]$ as

$$\rho(f) = 1+\log_2(f/f_{min}),\ \ \theta(f) = \theta_{ref}+2\pi\log_2(f/f_{ref}),$$

where $f_{min}$ and $f_{max}$ denote the lowest and highest frequencies to be displayed and $f_{ref}$ is the tuning reference displayed at angle $\theta_{ref}$. A typical cutoff frequency range is from 1Hz (high resolution tuning) to 10Hz (musical play with notes, possibly with a soft vibrato).

On the Snail viewer, the analysed spectrum is represented on the spiral skeleton as follows: for each frequency, the thickness and the brightness are both mapped to the loudness of the spectrum modulus `Amp_v` multiplied by the signal `PhiCstcy_v`, and the color is mapped to `PhiCstcy_v`. The left column of figure 2 illustrates the accuracy improvement resulting from the selection signal `PhiCstcy_v` for $f_c$ = 2Hz (Tuner mode in the lower figure) and for $f_c$ = 6Hz (Music Mode in the upper figure).

### 2.2 Cent-sensitive estimate for snail viewer

As expected on the spiral skeleton, a constant frequency accuracy ($\pm f_c$) makes the spectral components appear larger in the low frequency range. An accuracy enhancer adapted to the cent scale ($\kappa$) can be straightforwardly processed, by making the cutoff frequency $f_c$ in (A6) depend on each analysed frequency $f$ as

$$f_c(f) = 2^{\frac{\kappa}{1200}} f.$$

The right column of figure 2 illustrates the resulting visual rendering for $\kappa = 2$ cents (Tuner mode in the lower figure) and for $\kappa = 5$ cents (Music Mode in the upper figure).

### 2.3 Cent-sensitive estimate for strobe-like rendering

The strobe-like viewer proposed in [3] displays the 12 dials (one per chroma) composed of rotating concentric cyclic patterns (one per octave). Each pattern is associated with its target note frequency. For the equal temperament, these pitches are

$$f_m = f_{ref} 2^{(m-m_{ref})/12}$$

where $m$ denotes the integer midi code and $f_{ref}$ is the reference frequency of the midi code $m_{ref}$, e.g. $f_{ref} = 440Hz$ and $m_{ref} = 69$ for the standard note A4.

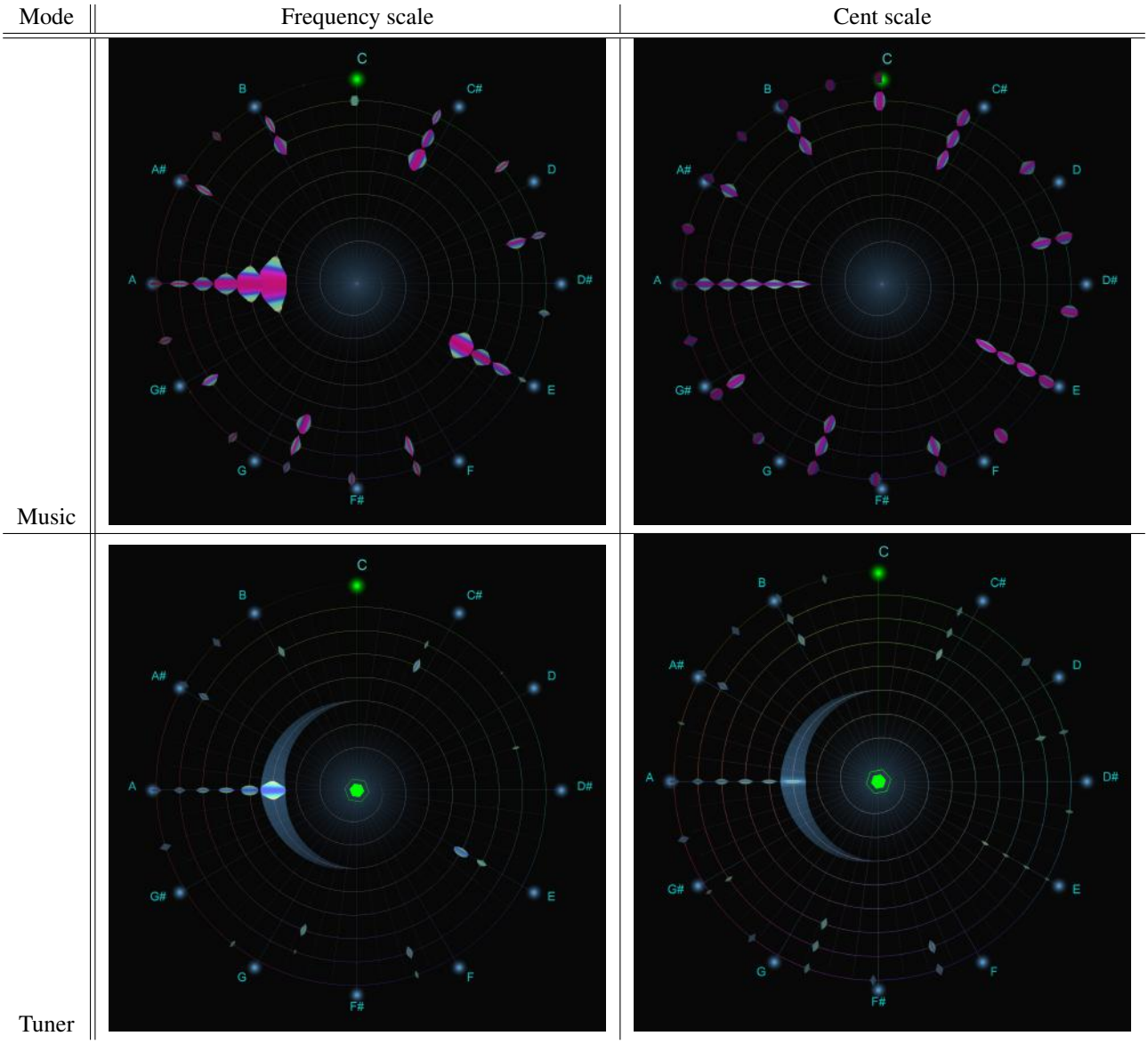



Figure 2. Visual renderings of the Snail-Analyser for several accuracy modes.

The color and the brightness of each pattern are mapped as in the Snail viewer. The angle $\Theta$ of each pattern is chosen as the demodulated phase `PhiDem_v` in (A5) computed for its own frequency $f_m$. The overall visual rendering (see figure 3 below for illustrative examples) is accurate and the rotation speeds are consistent with electromechanical strobe tuners.

However, the sensivity of these rotation speeds becomes uncomfortable for high pitches as, in this case, even a slight detuning causes rapid spinning. To restore a cent-sensitivity, we propose to update the angle $\Theta$ of each pattern according to the cent deviation, computed from the frequency deviation, itself estimated from the demodulated phase as detailed below.

Indeed, for each Fourier frequency $f$, the demodulated phase `PhiDem_v` in (A5) is constant in time if the frequency of the corresponding audio component is exactly $f$: the signal `PhiCstcy_v` in (A6) estimates such a constancy in a tolerance range parameterized by $f_c$. A more reactive (local-in-time) descriptor proposed in [1] (but not represented in figure 1) is the difference (modulo $2\pi$) between two successive `PhiDem_v`. This descriptor is locally zero for a locally constant demodulated phase and allows the estimation of the average frequency deviation as

$$\delta F(t_{n+1}, f) = \frac{M[\phi_d(t_{n+1}, f) - \phi_d(t_n, f)]}{2\pi T},$$

below the aliasing frequency $1/(2T)$ with $T = t_{n+1} - t_n$, and where $M$ denotes the $2\pi$-modulo centered on $(-\pi, \pi)$. The cent deviation is then given by

$$\begin{aligned}\delta\kappa(t_{n+1}, f) &= 1200\big[\log_2\big(f + \delta F(t_{n+1}, f)\big) - \log_2 f\big] \\ &= 1200 \log_2\left(1 + \frac{\delta F(t_{n+1}, f)}{f}\right),\end{aligned}$$

a good approximation of which is given by

$$\delta\kappa(t_{n+1}, f) \approx \frac{1200}{\ln 2}\,\frac{\delta F(t_{n+1}, f)}{f}.$$

Then, given the cent sensitivity $G$ in cents by turn, updating each pattern angle $\Theta$ as

$$\Theta(t_{n+1}, f) = \Theta(t_n, f) + 2\pi \frac{\delta\kappa(t_{n+1}, f)}{G}$$
$$\approx \Theta(t_n, f) + \frac{1200}{f\,G\,T \ln 2}\, M[\phi_d(t_{n+1}, f) - \phi_d(t_n, f)],$$

provides a reactive spinning for strobe-like rendering, which is cent-sensitive. This solution is the one implemented in the work presented below.

## 3. SOFTWARE

The StrobeTuner software is the first prototype developed for the first stage of the ATRIM project. As more prototypes will come in the future, and in order to reuse and isolate properly the different components, we relied on a specific architecture for the development and the communication model between the DSP kernel of the Snail Analyser and the diverse hosting applications. We will first describe the chosen underlying architecture, then the design process and finally explain more precisely the anatomy of the StrobeTuner application.

### 3.1 Architectural considerations for the ATRIM project

Data visualisation and User Interface design involve a lot of iterations during the design stage. As the User Experience remains at the center of the development process, it is important to have a way to separate both the development of the Snail Analyser DSP kernel, subject to slow changes, and the development of the visual components, subject to faster iterations during the design of the prototypes.
In order to meet these criteria, the Snail Analyser kernel was encapsulated in the form of a C language library, without any dependencies to any graphical libraries or visual considerations. A prior work has been made in that sense for deploying the Snail Analyser in web and embedded environments [9]. This new C library was built on top of these prior works, with an extended separation of the components and a broader access to the kernel parameters through its API.
Although it is still a work in progress, the decoupling allows to integrate the DSP kernel in different hosts independently. More precisely, the communication model used to pass each analysis frame from the library to possible hosts has been standardized for most solutions (real-time or non real-time) and reduced to the use of a single producer-consumer lock-free FIFO (see figure 4). This last solution (although optional) ensures that the kernel can be used in a multi-threaded environment and communicates safely its output frames from a real-time thread to an event thread, which is usually the case for real-time audio applications.

### 3.2 Design issues and Prototyping in Max

For a rapid prototyping of the visualization, we used Max, a patching environment for music and multimedia developed by Cycling74 [10]. Two reasons made us converge towards this choice. First, the possibility to easily create a Max object using a static version of the Snail kernel library. Second, Max's Javascript user interface (JSUI) allows for faster development iterations for prototyping and visual design than a compiled language environment such as C or C++.

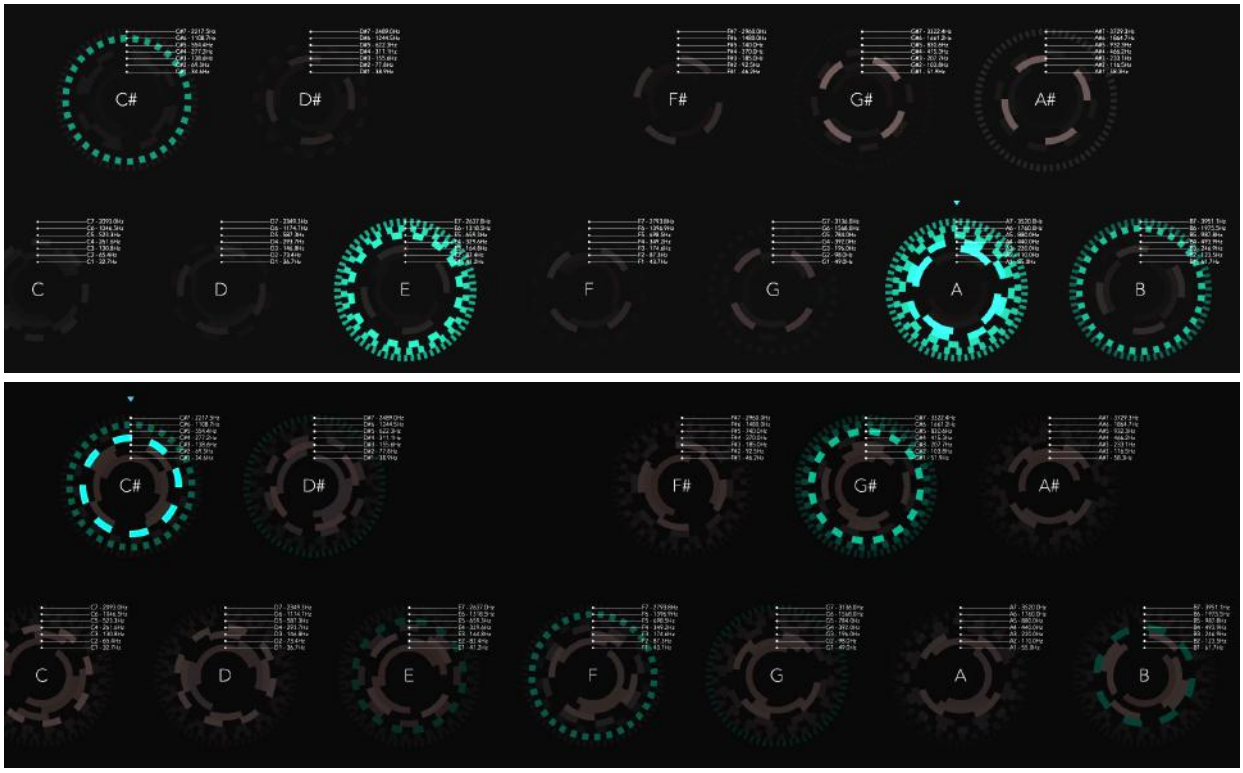
As we aim for a continuous development process, the approach also encourages the exploration of visual possibilities, for example by using different shapes or color cards (see figure 5) before its final implementation in the standalone software.

Once the design stage reached a minimal advancement, the next stage is to implement the results using the JUCE framework, a C++ framework for audio application and plug-in development [11], and deploy the prototypes in the form of Desktop and Mobile applications. The graphical display routines are adapted to the JUCE graphics API, and the necessary analysis parameters to control the Snail kernel used in the MAX prototype are integrated as User Interface controls.
Additionally, a VAMP plugin prototype was developed to evaluate the relevance of using the Snail analysis engine in a non-real time context.
In the future, we look for a possible integration in another cross-platform application development framework called Flutter [12], which allows to design and modify on-the-fly the code of an application thanks to its Virtual Machine hot-reload feature [13]. This option should help us to iterate faster for the visual prototyping and development stage.

### 3.3 Anatomy of the StrobeTuner

#### *3.3.1 Application Flow Diagram*

As depicted in figure 6, the StrobeTuner application controller uses five parameters to drive the kernel analysis. The output features are then pulled from the engine analysis queue and post-processed to update the visual strobe-like display view. The `Window Size` in milliseconds drives the underlying Fourier analysis window size. The window step used internally for the analysis is fixed internally to a 5 millisecond value in the kernel. The `Frequency Tracking` parameter controls the phase filtering process, either with a global cutoff frequency in Hertz for all filters, or with a value in cents that sets the cutoff frequency of each filter individually if we use the `Cents Filter Mode`.
The `Cents Sensitivity` parameter, exposed earlier in this paper, directly correlates with the `Demodulated Phase Difference` and allows visually to control the amount of rotation per cents deviation (by default set to 20 cents) for each ring in a dial. That value means that, for a frequency whose deviation is exactly 20 cents, the ring will spin at one round per second.
The `Tuning Ref` parameter allows to change the reference frequency (by default A4=440Hz) used internally by in the kernel. Finally, a `Visual Emphasis` acts as a visual gain input in decibels to enhance the overall brightness of the dials. Each output analysis frame of the Snail kernel is then processed such as, for each frequency target (ring) in a dial, an interpolation of the required features (1. Demodulation Phase Difference, 2. Amplitude, and 3. Phase




ⓐ




ⓑ

Figure 3. (a) Sawtooth waveshape (note A3, 220 Hz), (b) Clarinet (note C#4, ≈277Hz).

Constancy) is performed. Those values are then converted in both a rotation and a color to fill the corresponding visual attributes of each rings for the twelve dials.

### *3.3.2 User Interface Description*

The main view of Strobe-Tuner is inspired by the interface and layout of the conventional mechanical stroboscopic tuner (see figure 7).

Twelve dials corresponding to the twelve pitches of the chromatic scale are displayed from left to right and bottom to top. The natural pitches dials (C, D, E, F, G, A, B) are displayed on the bottom line and the altered pitch dials (C#, D#, F#, G#, A#) are arranged on the top line with a half-dial translation to the right. A zoom mode also allows to zoom and view only a specific dial (see figure 8)

Inside each of these 12 dials is a digital version of the visual pattern commonly used in mechanical strobe tuners, that is, 7 concentric rings composed of binary cyclic patterns, with a $2^k$-periodicity from octave $k = 0$ at center to $k = 6$ at the periphery (using the midi convention where A4=440Hz). In a standard strobe tuner, these rings are attached on a wheel that rotates at the frequency $f$ of the targeted note (at octave 1). When enlightened according to a sound wave signal that includes an active spectral component of frequency $2^k f$, the stroboscopic effect makes the cyclic pattern of the ring $k$ appear like motionless.
In the present visualization, the digital version emulates a rendering similar to the legacy strobe-tuner, with one addition: the rings on the dials are no longer interlocked. Each ring inside a dial have its own target frequency rotation retrieved from the analysis results transmitted by the snail processing library.

This extra information, although it may feel quite surprising the first time, displays an additional visual information on the sound characteristics. The user should be able to identify the fundamental note of the sound and its harmonics (or partials) as long as their frequency deviation is close enough to the target frequency of the nearest ring (and does not exceed the detection range, in cents, of the Snail kernel).
In a default mode, the rings on the dials are tuned using the twelve-tone (12-TET) equal temperament. This means that the additional tuning (and timbral) information conveyed by the display will be dependent on the actual timbre of the input sound (harmonic/inharmonic) and the used temperament.
In the example at the top of the figure 3, we can see the display of digital Sawtooth wave playing a A3 (220Hz) and its visible harmonics 2 (440Hz, A4), 3 (660Hz, ∼E5), 4 (880Hz, A5), 5 (1100Hz, ∼C#6), 6 (1320Hz, ∼E6), 8 (1760Hz, ∼A6), 9 (1980Hz, ∼B6), 12 (2640Hz, ∼E7), 16 (3520Hz, ∼A7), 18 (3960Hz, ∼B7). The other harmonics of the sound do not lie close enough to the targeted frequencies to be visible in the display.
The second example, at the bottom of the figure 3, shows a clarinet playing a C#4 (∼277.2Hz) and the visible harmonics 3 (∼831Hz, ∼G#5), 4 (∼1109Hz, C#6) and 5 (∼1386Hz, ∼F6).

## 3.4 Future developments

The StrobeTuner software is mostly suited for the tuning of harmonic sounds. In that sense, the possibility to set an external target frequency map based on a different temperament, for example using the Just Intonation (see video 1)

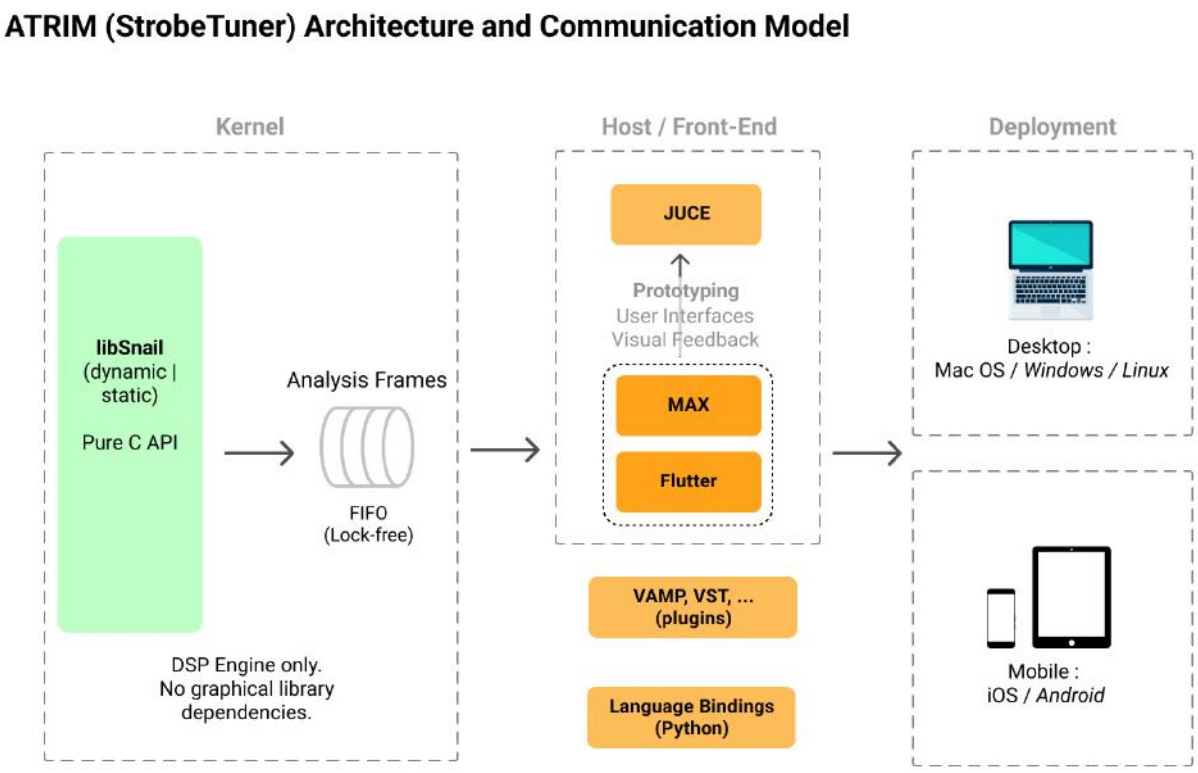



Figure 4. Architecture of the StrobeTuner.

or the Indian Svara scale (see video 12), or even to build a completely specific frequency map for exotic instruments, is already possible and should be accessible from the user interface soon.

Future development includes an architectural change of the output features of the kernel in order to refine the visual feedback according to the demodulated phase difference.

## 4. CONCLUSION

The Snail analysis process appear to be relevant for strobe-tuner viewers. The design of cent-sensitive spinner provides a comfortable visual rendering for musical use.

Further work will focus on real-time visual renderings of timbre and new features such as recording/playback modes to allow accurate and easy-to-read comparison between two instruments or musicians.

### Acknowledgments

The authors thank the French Ministry of Higher Education and Research and the National Research Agency for the financial support of the ATRIM project, through the plan "France Relance". This project is collaborative work between Buffet Crampon and the STMS laboratory.

## 5. REFERENCES

[1] T. Hélie, "Analyse fréquentielle par démduolation de phase d'un signal acoustique," French Patent App. FR1455456A (2014), Int. Patent App. WO2015/189419 A1 (2015), Assignee: Centre National de la Recherche Scientifique, 2014.

[2] T. Hélie and C. Picasso, "The snail: a real-time software application to visualize sounds," in *Proc. DAFx*, 2017.

[3] T. Hélie, C. Picasso, R. Piéchaud, M. Jousserand, and T. Colinot, "Variations on the Snail-Analyser and its spectral estimates for the accurate tuning of musical sounds: a strobe tuner like display," in *31st International Electrotechnical and Computer Science Conference ERK 2022*, Portoroz, Slovenia, Sep. 2022. [Online]. Available: https://hal.science/hal-03782993

[4] R. W. Young and L. Allen, "Stroboscope," 1938, uS2286030A. [Online]. Available: https://patents.google.com/patent/US2286030A/en

[5] G. E. Arends and R. S. Dobrose, "Strobe tuner," 1997, uS5877443A. [Online]. Available: https://patents.google.com/patent/US5877443A/en

[6] M. J. Skubic, "Electronic strobe tuning aid," 2001, uS6580024B2. [On-

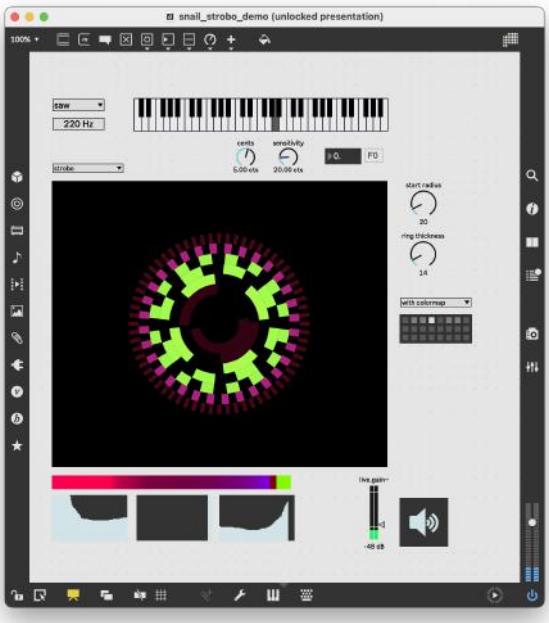



Figure 5. Max prototype.

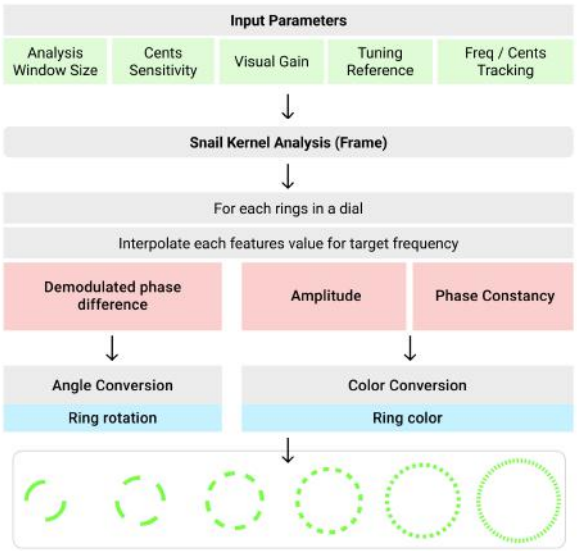



Figure 6. StrobeTuner Application Flow Chart, with displayed ring pattern (see bottom) ranging from octave 2 to 7.


Figure 7. StrobeTuner User Interface.

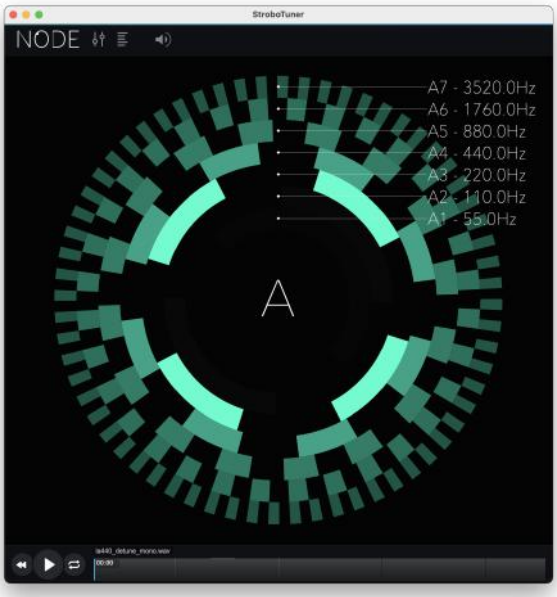



Figure 8. The StrobeTuner Zoomed Dial Mode.

line]. Available: https://patents.google.com/patent/US6580024B2/en?inventor=Michael+J.+Skubic

[7] "StroboStomp HD Tuner (see also iStroboSoft)," 2019. [Online]. Available: https://www.petersontuners.com/products/strobostomphd/

[8] Linotune, "User's guide (v1.14 - Oct. 2018)," linotune LLC, Farmington MO 63640, USA, 2011, https://linotune.com/download/guide.pdf.

[9] R. Piéchaud and Q. Lamerand, "iMuSciCA project," http://www.imuscica.eu/wp-content/uploads/2018/11/imuscica-press-kit-web.pdf.

[10] Cycling74, "Max home page," https://cycling74.com/products/max.

[11] Website, "JUCE Home Page," https://www.juce.com.

[12] "Flutter Framework," website: https://flutter.dev/.

[13] "Flutter Hot Reload documentation," website: https://docs.flutter.dev/development/tools/hot-reload.