The Spatbox: An Intuitive Trajectory Engine to Spatialize Sound

Pierre Lecomte

THE SPATBOX: AN INTUITIVE TRAJECTORY ENGINE TO SPATIALIZE SOUND

Pierre Lecomte
Univ Lyon, Univ Claude Bernard Lyon 1,
CNRS, Ecole Centrale de Lyon, INSA Lyon,
LMFA, UMR5509, 69622 Villeurbanne France
pierre.lecomte@univ-lyon1.fr

ABSTRACT

This paper presents the software part of the spatbox project [1] : an interface to intuitively generate 3D trajectories and to spatialize them in real-time on a loudspeaker array with Ambisonics. Each trajectory is described by a parametric curve, where each of the coordinates varies according to a configurable LFO. Depending on the LFO parameters, many trajectories can be generated. The similarity of the interface to that of a synthesizer allows the user to quickly get to grips with the tool. In particular, it is possible to create trajectory presets for fixed parameter configurations. An implementation in the Faust language is made and a visual feedback tool is introduced. Some additional features for the trajectory modification are proposed: choice of the coordinate system, trajectory scaling and acceleration, Doppler effect, hold/reset and rotation/translation functionalities.

1. INTRODUCTION

Spatial-, immersive-, or 3D-sound is now being widely spread to the greatest number. Notably in the cinema with commercial system such as Dolby Atmos [2] , or in the gaming sector with binaural sound. More recently, popular music concert are being offered with immersive sound for large venue, for instance with the L-ISA system [3] from L-Acoustics. However, the artistic creation of immersive sound content is still not easily accessible to the "ordinary" musician. This is mainly due to the fact that a lot of equipment is needed when making 3D sound: a dedicated room with many loudspeakers. Consequently, it is rather the professional musical industry or artistic creation centers that occupy this sector. Moreover, the composition often requires experimentation and reflection to choose the position of the sources in space and define their trajectories over time, although a live and improvised spatialization could open up a whole new creative horizon. From a software perspective, the offer is growing and many solutions allow professionals to create spatial sound content. One can mention software such as Sound Particles [4] , Ircam/Flux Spat Revolution [5] [1], GRM Tools [6] , Noise Makers [7] , Blue Ripple [8] , and many others [2]. On the open-source side: IEM-Plugin-suite [9] , ambitools [3] [10] or Sparta [11] [4] for sound generation and IanniX [12] for trajectory sequencing. These software fit well in a 3D audio content creation process with a Digital Audio Workstation (DAW) but are not directly designed for live spatialization where one wishes to improvise the sound trajectories. The use of hardware interfaces can be a relevant choice for this purpose. To the author's knowledge, there exist almost no autonomous interfaces that allow the spatialization of 3D sound on a loudspeaker array in an intuitive way [13] , apart from a mixing console used for the diffusion of sound, notably in the practice of electroacoustic music [5, 6].

It is in this context that the spatbox project takes place. It seeks to make an hardware autonomous interface which allows the user to instantly place sound on trajectories in space with a few loudspeakers, by turning a few knobs. With this instrument, the sounds travel on trajectories generated in an intuitive way, like creating sound patches on a synthesizer. As it is a kind of box to spatialize the sound, the project is called "spatbox". In this paper, the algorithms for generating the trajectories and computing the loudspeaker signals for the spatbox are described. An implementation in the Faust language [14] [7] is made and a software version of the spatbox is presented. The use of the Faust language allows to target an embedded system architecture for the hardware realization of the spatbox.

The paper proceeds as follows: In Sec. 2, the choice of the Ambisonics spatialization approach is argued, the needed equations are reviewed. The trajectory generation engine, two examples and several additional features are presented in Sec. 3. The user interface as well as a visual feedback tool to view the trajectories are introduced

[1] http://www.sekisushai.net/spatbox/
[2] https://www.dolby.com/technologies/dolby-atmos/
[3] https://www.l-isa-immersive.com/

[4] https://soundparticles.com/
[5] https://www.flux.audio/project/spat-revolution/
[6] https://inagrm.com/fr/store
[7] https://www.noisemakers.fr/
[8] https://www.blueripplesound.com/
[9] https://plugins.iem.at/
[10] http://www.sekisushai.net/ambitools/
[11] https://leomccormack.github.io/sparta-site/
[12] https://www.iannix.org/en/
[13] https://blog.bela.io/multichannel-sound-spatialisation/
[14] https://faust.grame.fr/

in Sec.4. Finally, conclusion and future works are done in Sec. 5.

2. AMBISONICS

There exists many approaches to spatialize sound in space with a loudspeaker array. Among the most popular are Vector Base Amplitude Panning (VPAB) [8], Wave Field Synthesis (WFS) [9] or Higher Order Ambisonics [10], simply called Ambisonics for the rest of this paper. The trajectory engine described in Sec. 3 could be independent of the spatialization technique. However, the main objective of the spatbox is to provide an autonomous interface that takes the sound signals to be spatialized and as inputs and outputs the loudspeaker signals. Therefore the spatialization engine should be embedded.

The choice is made to use Ambisonics in this project for several reasons. First, the Ambisonics signals flow goes through encoding, transformation (i.e. spatial audio effects) and decoding of the sound field [10, Chap. 5]. For the encoding step, it is possible to encode the source distance using near-field filters [11], which is an key feature for a realistic rendering of 3D trajectories. Then, spatial transformations are possible before the decoding step, for example reverberation, spatial blur [12] or warping [13] of the sound scene. These effects can then be integrated into the spatbox spatialization engine with dedicated controllers, as a master effect on a mixing console. Finally, the decoding of the sound scene is flexible and can be done on an irregular loudspeaker array [14] or even in binaural for a headphone rendering [15]. The control of the spatial resolution is easily done with the degree of decomposition on the basis of spherical harmonics and immersive sound results can be obtained with only less than a dozen loudspeakers. Thus, it becomes possible to realize a hardware interface requiring few loudspeakers and thus makes the project accessible to the largest number of musicians.

In the rest of this section, one recalls the main Ambisonic equations which are used to encode, decode or directly pan a virtual source on a loudspeaker array. For a more extensive description, refer to [16] for example.

2.1 Point Source Encoding

Let's consider a point source, located at a position $(r_s(t),\ \theta_s(t),\ \phi_s(t))$ in the spherical coordinate system, where $r \in]0,\ \infty[$ is the radius, $\theta \in]-\pi,\ \pi]$ the azimuth angle, $\phi \in [-\pi/2,\ \pi/2]$ the elevation angle and t represents the time variable. This source carries a signal $s(t)$. The ambisonic encoding of this source produces the signals $b_{l,m}(t)$ of degree l, order m with $(l,m) \in (\mathbb{N} \times \mathbb{Z}), |m| \leq l$, such that:

$$b_{l,m}(t) = s(t - r_s/c) * \text{NF}_l(r_s(t), r_0, t) \frac{r_0}{r_s} Y_{l,m}(\theta_s(t), \phi_s(t)). \quad (1)$$

In Equation (1), $c \simeq 340$ m/s is the sound speed, $Y_{l,m}$ is the real spherical harmonic of degree l and order m, $*$ is the convolution operator and NF_l are the near-field filters of degree l [11]. In the current implementation, these filters are stabilized by Near-Field Compensation (NFC) filters at a fixed radius $r_0 = 1$ m [3]. When $r_s < r_0$, a "bass-boost" effect occurs. To limit the latter, which is combined with an amplification in $1/r_s$ in Equation (1), a minimum radius $r_{\text{min}} = 0.5$ m is introduced as the minimal possible distance for the source. For an encoding up to degree L, the corresponding maximal amplification of the $s(t)$ is $(L+1) \times 20 \log_{10}(r_0/r_{\text{min}}) \simeq 6(L+1)$ dB.

2.2 Sampling Ambisonic Decoder

The decoding step on a loudspeaker array consists in calculating the signals of the loudspeakers from the ambisonic signals $b_{l,m}$. Many strategies exist for this step. A review can be found in [10, chapt. 4] for example.

In this paper one uses the Sampling Ambisonic Decoder (SAD) technique. The loudspeaker signals are obtained from the simple source continuous formulation [16] sampled in the direction of the loudspeakers as:

$$g_n(t) = \frac{r_n}{r_{\text{max}}} \sum_{l=0}^{L} w_{\text{max-}\mathbf{r}_E,l,L} \text{NFC}_l(r_n, t) * \sum_{m=-l}^{l} Y_{l,m}(\theta_n, \phi_n) b_{l,m} \left(t - \frac{r_{\text{max}} - r_n}{c} \right). \quad (2)$$

In Equation (2), n stands for the $n-$th loudspeaker in an array of N loudspeakers. The ambisonics signals are weighted with max-$\mathbf{r}_E$ weights, denoted $w_{\text{max-}\mathbf{r}_E,l,L}$ in order to maximize the energy vector $\mathbf{r}_E$, improving high frequency localization [17]. Note that the near-field effect of each loudspeaker is compensated with NFC filters. Moreover, the amplitude is scaled to r_n/r_{max} where r_{max} is the maximal distance among the N loudspeakers and the signal is delayed by the corresponding propagation distance relative to r_{max}. This ensures that all loudspeakers have the same gain and delay at origin when playing the same signal. The choice of the decomposition degree L (i.e. Ambisonic order), is generally related to the number of loudspeaker N, such that $N \geq (L+1)^2$.

2.3 Equivalent Panning Law

By combining Equations (1) and (2), and using the addition theorem of spherical harmonics, an equivalent panning law is derived [16] as:

$$g_n(t) = s \left(t - \frac{r_s + r_{\text{max}} - r_n}{c} \right) \frac{r_n}{r_{\text{max}}} \frac{r_0}{r_s} \sum_{l=0}^{L} (2l+1) w_{\text{max-}\mathbf{r}_E,l,L} * \text{NFC}_l(r_n, t) * \text{NF}_l(r_s(t), r_0, t) P_l(\cos(\gamma_n(t))), \quad (3)$$

where P_l is the l-th Legendre polynomial, and $\gamma_n(t) = \cos(\phi_s(t)) \cos(\phi_n) \cos(\theta_s(t) - \theta_n) + \sin(\phi_s(t)) \sin(\phi_n)$ is the angle between the source and the n-th loudspeaker. This formulation allows obtaining the loudspeaker signals with much fewer computation than the formalism with encoding and decoding steps. This is a good point for efficient implementation, especially when making the autonomous hardware interface by using an embedded system with low computing power. However the use of such

panning law no longer allows the use of spatial ambisonic effects before decoding. In addition, if the loudspeakers are not arranged in a "regular way" around the origin, reduced loudness at the location of poor loudspeaker coverage will be noticeable [10, p. 72].

3. TRAJECTORY ENGINE

The aim of the spatbox trajectory engine is to generate the signals $r_s(t)$, $\theta_s(t)$ and $\phi_s(t)$ in the Eqs (1) or (3), i.e. the trajectory that the source follows over time. There are an infinite number of possible trajectories and as many equations to describe them. One therefore constrain this problem in the following way:

- The trajectories are described in a parametric way so that the user can generate a multitude of them by varying the parameters.
- A maximum of trajectories should be generated with a minimum of parameters. Indeed, as the final goal is to make an hardware interface, the minimum number of controllers are desired.

The problem is then to find how to describe the trajectories in a parametric way and which parameters to associate to them. A proposal for a solution to this problem is made in the following Sec. 3.2. The current implementation of this engine is made with the Faust language. The ambisonic layer is made with ambitools[10] [3].

3.1 Parametric Equation

Let be a trajectory that represents all the positions ($r_s(t)$, $\theta_s(t)$, $\phi_s(t)$) that the source takes over time. This trajectory is expressed in a parametric way as follows, where the parameter is the time t:

$$\begin{aligned} r_s(t) &= \sqrt{x_s(t)^2 + y_s(t)^2 + z_s(t)^2} \\ \theta_s(t) &= \arctan(y_s(t)/x_s(t)) \\ \phi_s(t) &= \arcsin(z_s(t)/r_s(t)) \end{aligned} \tag{4}$$

In Equation (4), the functions $x_s(t)$, $y_s(t)$ and $z_s(t)$ are the trajectory coordinates functions in the Cartesian coordinate system. The choice of the coordinate system is left to the user for each trajectory to generate. Thus, either the signals, $r_s(t)$, $\theta_s(t)$, $\phi_s(t)$ in spherical coordinates are directly generated, or the signals $x_s(t)$, $y_s(t)$, $z_s(t)$ in Cartesian coordinates are generated and then converted with Equation (4).

3.2 Low Frequency Oscillators as Trajectory Coordinates Functions

To express the time variations of ($x_s(t)$, $y_s(t)$, $z_s(t)$) or ($r_s(t)$, $\theta_s(t)$, $\phi_s(t)$), i.e. the trajectory coordinates functions in Equation (4), one chooses to use Low Frequency Oscillators (LFO)s. LFOs are commonly used in synthesizers to modulate the parameters of the sound patch. Used here as coordinate generators, they can produce a multitude of different trajectories in three dimensions. A LFO is usually a periodic function that oscillates around 0 and that can be characterized by its frequency f, waveform, amplitude a, phase or duty cycle p:

$$\text{LFO}(t) = a \times \text{waveform}(f, p, t) \tag{5}$$

The phase/duty cycle $p \in [0\ 1]$ is set as a fraction of the waveform unit period. The frequency f of the LFO is kept low as it represents the cyclic variations of the corresponding coordinate signal, and high speed can be reached with only a few Hertz. In the current implementation, $f \in [0\ 1]$ Hz, and the user sets one LFO per coordinate : $\text{LFO}_{x_s}(t)$, $\text{LFO}_{y_s}(t)$, $\text{LFO}_{z_s}(t)$ or $\text{LFO}_{r_s}(t)$, $\text{LFO}_{\theta_s}(t)$, $\text{LFO}_{\phi_s}(t)$.

Bounds When assigned to the Cartesian coordinates $\text{LFO}_{x_s}(t)$, $\text{LFO}_{y_s}(t)$ or $\text{LFO}_{z_s}(t) \in [-a,\ a]$ with $a \in [0,\ 1]$. When assigned to the spherical coordinates, $\text{LFO}_{r_s}(t) \in [-a + 1 + r_{\text{min}},\ a + 1 + r_{\text{min}}]$, $\text{LFO}_{\theta_s}(t) \in [-a\pi,\ a\pi]$ and $\text{LFO}_{\phi_s}(t) \in [-a\pi/2\ , a\pi/2]$.

Waveforms Six different waveforms are proposed in the current implementation, as shown in Tab. 1 over two periods. Note that for the square wave, the user sets the duty cycle instead of the phase. The noise waveform is generated with a sample-and-hold technique where a sample is hold during one period. The phase parameter in this situation sets the instant during the period when a new sample is hold.

ID	Name	Waveform
0	Sawtooth	
1	Sawtooth 2	
2	Sine	
3	Triangle	
4	Square	
5	Noise	

Table 1. The six waveforms for the LFOs used as trajectory coordinates functions. The waveform are shown over two periods.

3.3 Trajectory Generation Principle

To generate a trajectory patch, one proceeds as follows (see Fig. 3 for a graphical user interface view). First, one chooses the coordinate system between Cartesian or spherical. Then one sets each LFO waveform for each coordinates and finally one sets the LFO parameters : amplitude, frequency, phase. Following this procedure, two examples of trajectory are given:

Helix In Cartesian coordinates, a parametric Helix trajectory around the $z-$axis can be expressed with LFOs as follows:

$$\begin{aligned} x_s(t) &= a_0 \sin(2\pi f_0 t + p_0) \\ y_s(t) &= a_1 \sin(2\pi(f_0 t + p_0 + 0.25)) \\ z_s(t) &= a_2 \text{ sawtooth}(f_2, p, t) \end{aligned} \tag{6}$$

The first $\text{LFO}_{x_s}(t)$ is a sinewave with amplitude a_0, frequency f_0 and phase p_0, The second $\text{LFO}_{y_s}(t)$ is a cosine wave with respect to $x_s(t)$, obtained from a sinewave waveform where the phase is set to $p_1 = p_0 + 0.25$. Its amplitude is a_1. With $f_1 = f_0$, the horizontal trajectory is an ellipse of semi-axis a_0 and a_1. For the third $\text{LFO}_{z_s}(t)$, a sawtooth waveform of frequency f_2, amplitude a_2, phase p_2 sets the pitch of the Helix. This trajectory is visible in red in Fig. 1 and the corresponding patch parameters are shown on the left side of Fig. 3.

Flower In spherical coordinates, a flower-shaped trajectory centered at origin in the horizontal plane can be described as follows:

$$\begin{aligned} r_s(t) &= a_0(\sin(2\pi(f_0 t + p_0)) + 1) + r_{\text{min}} \\ \theta_s(t) &= a_1 \text{sawtooth}(2\pi f_1 t + p_1) \\ \phi_s(t) &= 0 \end{aligned} \tag{7}$$

The first $\text{LFO}_{r_s}(t)$, produces a sinusoidal variation of $r_s(t)$ with minimal radius r_{min}, while the second $\text{LFO}_{\theta_s}(t)$ is a sawtooth, which produces a linear increase in azimuth angle. The third $\text{LFO}_{\phi_s}(t)$ is set to null amplitude and phase such that the trajectory stays on the horizontal plane. This trajectory is visible in cyan in Fig. 1 and the corresponding patch parameters are shown on the right side of Fig. 3.

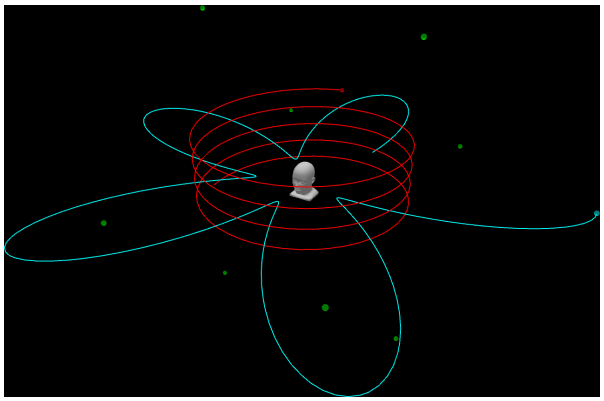

Figure 1. Example of two trajectories generated with the spatbox and visualized on the 3D visualizer: A helix in red and a flower-shaped trajectory in cyan. The green points correspond to the position of the loudspeakers.

Through these two simple examples, one can realize the great variety of trajectory patches that can be made by varying waveforms, frequency, amplitude and phase ratio between each LFOs. Since LFOs are periodic functions (except for the "noise" waveform), the resulting trajectories are also periodic as long as the patch parameters are not moved. As a result, they could fit well with a repetitive sound signal as found in electronic music for instance. Note however that it is quite simple to make the trajectory evolve over a long period of time or to modulate it, by slightly varying a frequency ratio between the LFOs.

3.4 LFOs Synchronization

One can notice, from the Helix trajectory example, that the phase relationship between the LFOs is critical to define the trajectory properly. Since the phase at the origin is defined independently for each LFO, it is crucial to keep a synchronization between the LFOs when the frequency or phase parameters of one of them is changed. To achieve this, in the current implementation, as soon as one of these parameters moves, all LFOs are reset to $t = 0$ to keep their synchronization. Consequently the source returns to the beginning of the trajectory while the user is changing the monitored parameters. Note that a very fast random position change effect can be produced in this way, using the noise waveform: Indeed, as the LFO is reset, a new value for the noise is produced, which results in an immediate change of the assigned coordinates.

3.5 Forbidden Sphere

As explained in Sec. 2.1, a minimum radius is introduced, r_{min}, below which the source cannot go, to avoid excessive signal amplification (or even a singularity if $r_s = 0$). Therefore, the signal $r_s(t)$ is compared to r_{min} and equalized to it if found to be smaller. Consequently, a forbidden sphere of radius r_{min} is defined and it surrounds the origin. The source can never enter inside it. If the trajectory encounters this sphere, it "slides" over it as $\theta_s(t)$ and $\phi_s(t)$ continue to evolve. An example of such situation is shown in Fig. 2.

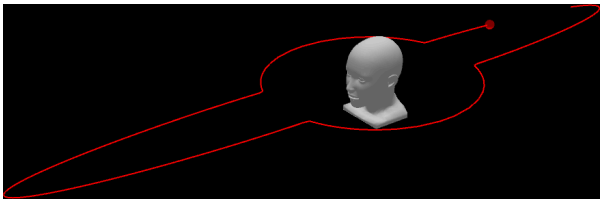

Figure 2. Example of an elliptic trajectory which encounters the forbidden sphere. As the radius $r_s(t) \geq r_{\text{min}}$ for all t, the source slides over the forbidden sphere.

3.6 Extra Features

This section presents some additional features to interact with the sound or the trajectories generated by the LFOs.

Doppler Effect A delay, which corresponds to the propagation time r_s/c to the origin (see Equations (1) and (3)) is eventually applied to the signal $s(t)$. When the source moves, this delay also varies and can produce a Doppler

effect. This results in changing the pitch of the signal $s(t)$, which can be an unwanted effect. Therefore, the application of this delay is left as an option to the user. Note that a playability problem may occur if the propagation delay is too large. For example, if the user places the source at 10 m while playing on a keyboard, it will take about 29 ms before the sound is heard.

Hold and Reset A hold function allows to stop the oscillation of the LFOs and to maintain them at their current value. Consequently, the source pauses on its trajectory. As soon as the hold is released, the source continues moving on its trajectory. In the same way, a reset function brings the source back to the beginning of its trajectory when activated.

Scale and Speed The amplitude of the LFOs is chosen by the user with a parameter $a \in [0\ 1]$ as mentioned in Sec. 3.2. A scale function allows to multiply the LFOs amplitude by an additional scaling factor. This scales up the whole trajectory. For Cartesian coordinates this function is realized by multiplying each LFO by the scale factor. For spherical coordinates, only the LFO assigned to $r_s(t)$ is multiplied by this scale factor. In the same way, a speed function can multiply each LFO frequency by a factor, resulting in an acceleration of the source on its trajectory.

Rotation and Translation The coordinate system in which the trajectories are expressed is centered on the origin. However, a rotation feature is available and allows to rotate the coordinate system around the z, y and x axes with yaw, pitch and roll angles respectively. As well, a translation can be made along the x, y and z axes. As a result, the trajectories can be rotated and translated.

4. USER INTERFACE

This section introduces the spatbox software user interface.

4.1 Graphical User Interface

In its current version, the spatbox is only available as a software, either as a standalone tool or as a plugin, depending on the architecture targeted during the Faust compilation. The number of source to spatialize, it set for the compilation. An example of graphical user interface for 2 sources is visible on Fig. 3 for a Linux Alsa standalone application. For each source, 3 menus, 17 rotary knobs, 2 buttons and 1 checkbox allow creating a trajectory patch. The Doppler effect can be activated and the sound volume be adjusted with a check box and a gain knob respectively. Note that it is to be expected that the volume of the source will naturally increase or decrease as the source moves closer or further away. The controllers labels “0”, “1” and “2” correspond to the coordinates x, y, z or r, θ ϕ respectively, depending on the choice of the coordinate system between Cartesian of spherical. For the LFO bank menu, the correspondence between the identification number and the waveform is shown in Tab. 1. Meters of the coordinate signals allow seeing the current value of each coordinate in the spherical system. The values of these meters are transmitted by Open Sound Control (OSC) message to the 3D visualizer presented in the following section.

4.2 Visual Feedback

Although trajectory patch generation can be intuitive, it is very difficult to rely solely on one’s auditory system to accurately localize the position of sources and monitor the trajectories. Indeed, in addition to the limitation of human spatial perception, the localization performances can be degraded depending on the ambisonic resolution and the loudspeaker array. See for example the listening tests reported in [10] for further reading on this topic. Therefore, it seems essential for the user to have a visual feedback to monitor the trajectories he produces. The latter is made with the Processing [15] language [3]. An illustration is shown in Figs. 1 and 2. It consists of a 3D view of a dummy’s head, centered at the origin and looking in the direction of positive x-axis. The proportions are preserved and the dimensions are in meters. The loudspeakers are represented by green dots whose size and color can change depending on the signal level in dBfs. The successive positions of the sources are received via OSC messages by the Faust application presented in Sec. 4.1. They are kept in a First In First Out (FIFO) memory buffer which keeps only the last 500 positions (this value can be changed). The trajectories are drawn by connecting the successive positions with lines. The current position of the source is materialized by a colored ball. The user can move, zoom and pan the camera to improve its perspective visualization.

5. CONCLUSION AND FUTURE WORKS

In this paper, the spatbox project was introduced as a tool for intuitive spatialization on a loudspeaker array. The LFO-based trajectory generation engine was presented along with several additional features to interact with the sound and the trajectories. The periodic character of the generated trajectories leads to the idea of synchronizing the frequency of the LFOs to a metronome, for example a MIDI clock. This is the topic of future work. An implementation in the Faust language has been made as well as a visual feedback tool to observe the trajectories. Currently, the spatbox exists only in a software form although a MIDI interface can already be connected to associate physical controllers to the interface parameters. A hardware version is currently being designed and built. It targets an embedded system platform such as Teensy [16], Bela [17] or Raspberry Pi [18]. On these platforms, recent audio interfaces are getting multichannel: up to 6 inputs and 8 outputs on the Teensy and Rasberrypi and even up to 16 outputs on the Bela. Thus, this will allow the spatbox to become a musical instrument on its own, capable of receiving audio signals as input and spatializing them in real time on a array of up to 16 loudspeakers. The visual feedback tool seems

[15] https://processing.org/
[16] https://www.pjrc.com/teensy/
[17] https://bela.io/
[18] https://www.raspberrypi.com/

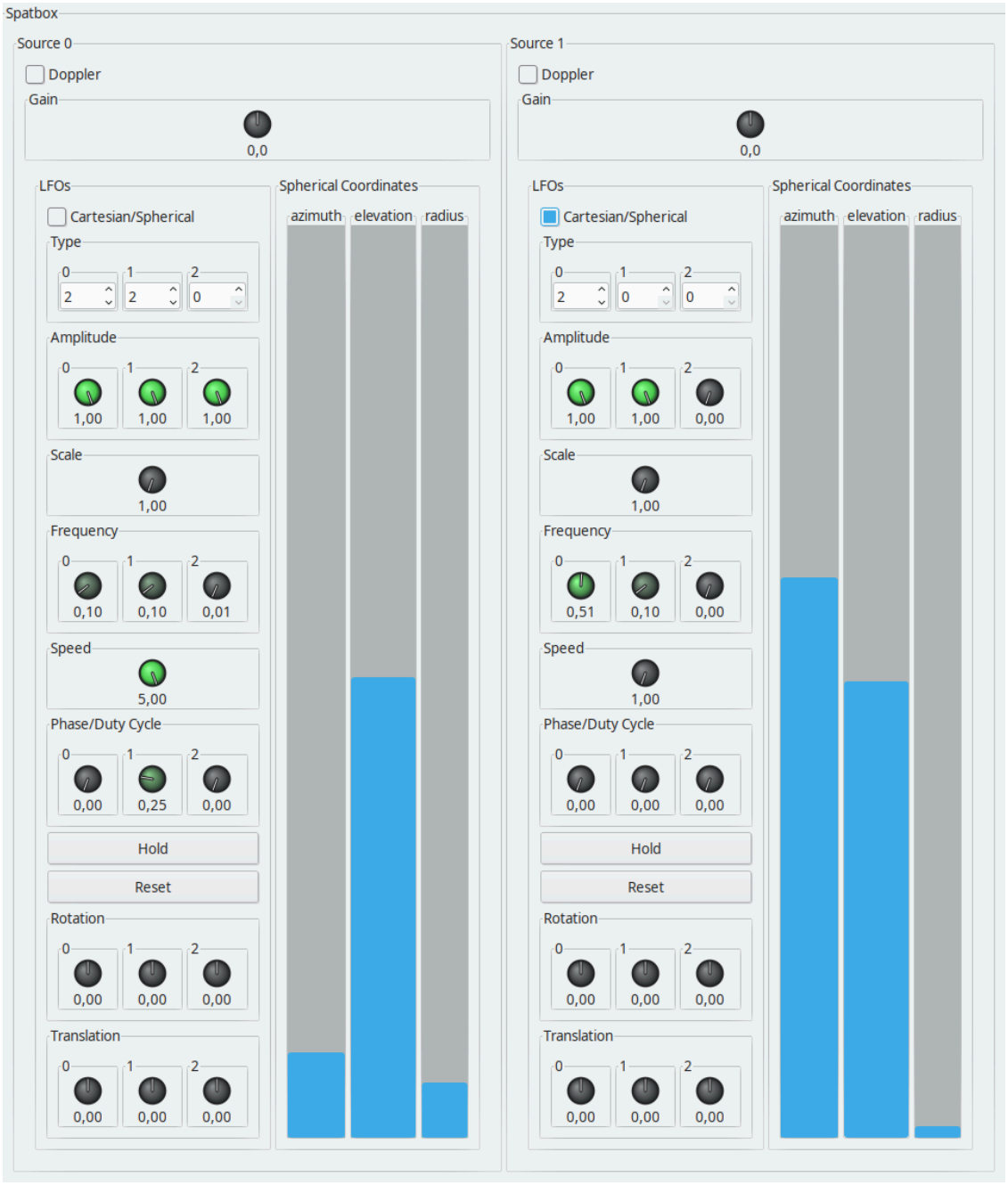

Figure 3. Graphical user interface for a standalone Linux Alsa application. Two sources are spatialized and their trajectory patch is set with the different controllers. The corresponding trajectories are visible on Fig. 1: The red trajectory corresponds to Source 0 and the cyan trajectory to Source 1.

however essential to the user and an embedded version of it is currently under study, although it is already possible to drive it on a computer with OSC messages. Once the hardware interface is available, it is planned to conduct a feedback study with musicians. Code and documentation for this project are available online [1] .

6. REFERENCES

[1] T. Carpentier, M. Noisternig, and O. Warusfel, "Twenty years of Ircam Spat: Looking back, looking forward," in *41st International Computer Music Conference (ICMC)*, 2015, pp. 270–277.

[2] J. D. Mathew, S. Huot, and B. F. Katz, "Survey and implications for the design of new 3D audio production and authoring tools," *Journal on Multimodal User Interfaces*, vol. 11, no. 3, pp. 277–287, 2017.

[3] P. Lecomte, "Ambitools: Tools for Sound Field Synthesis with Higher Order Ambisonics - V1.0," in *International Faust Conference*, Mainz, 2018, pp. 1–9.

[4] L. McCormack and A. Politis, "SPARTA & COMPASS: Real-time implementations of linear and parametric spatial audio reproduction and processing methods," in *Audio Engineering Society Conference: 2019 AES International Conference on Immersive and Interactive Audio*. York: AES, 2019, pp. 1–12.

[5] J. Harrison, "Sound, space, sculpture: Some thoughts on the 'what','how'and 'why'of sound diffusion," *Organised Sound*, vol. 3, no. 2, pp. 117–127, 1998.

[6] J. Prager, *L' Interprétation Acousmatique - Fondements Artistiques et Techniques de l'interprétation Des Œuvres Acousmatiques En Concert (in French)*, unpublished ed., 2012.

[7] Y. Orlarey, D. Fober, and S. Letz, "FAUST: An efficient functional approach to DSP programming," *New Computational Paradigms for Computer Music*, vol. 290, 2009.

[8] V. Pulkki, "Virtual sound source positioning using vector base amplitude panning," *Journal of the Audio Engineering Society*, vol. 45, no. 6, pp. 456–466, 1997.

[9] A. J. Berkhout, D. de Vries, and P. Vogel, "Acoustic control by wave field synthesis," *The Journal of the Acoustical Society of America*, vol. 93, no. 5, pp. 2764–2778, 1993.

[10] F. Zotter and M. Frank, *Ambisonics - A Practical 3D Audio Theory for Recording, Studio Production, Sound Reinforcement, and Virtual Reality*, ser. Springer Topics in Signal Processing. Cham, Switzerland: Springer, 2019.

[11] J. Daniel, "Spatial sound encoding including near field effect: Introducing distance coding filters and a viable, new ambisonic format," in *Audio Engineering Society Conference: 23rd International Conference: Signal Processing in Audio Recording and Reproduction*. Helsingør: AES, 2003, pp. 1–15.

[12] T. Carpentier, "Ambisonic spatial blur," in *Audio Engineering Society Convention 142*. Berlin: AES, 2017, pp. 1–7.

[13] H. Pomberger and F. Zotter, "Warping of 3D ambisonic recordings," in *Proc. of the 3rd Int. Symp. on Ambisonics & Spherical Acoustics*, Lexington, 2011.

[14] F. Zotter and M. Frank, "All-round ambisonic panning and decoding," *Journal of the Audio Engineering Society*, vol. 60, no. 10, pp. 807–820, 2012.

[15] M. Noisternig, A. Sontacchi, T. Musil, and R. Holdrich, "A 3d ambisonic based binaural sound reproduction system," in *Audio Engineering Society Conference: 24th International Conference: Multichannel Audio, The New Reality*. AES, 2003, pp. 1–5.

[16] P. Lecomte, P.-A. Gauthier, C. Langrenne, A. Berry, and A. Garcia, "A Fifty-Node Lebedev Grid and Its Applications to Ambisonics," *Journal of the Audio Engineering Society*, vol. 64, no. 11, pp. 868–881, 2016.

[17] J. Daniel, J.-B. Rault, and J.-D. Polack, "Ambisonics encoding of other audio formats for multiple listening conditions," in *Audio Engineering Society Convention 105*. San Francisco: AES, 1998, pp. 1–29.