ambitools Documentation

Pierre Lecomte
pierre.lecomte@gadz.org

June 18, 2016

Contents

1 Introduction

1.1 Goals of ambitools

ambitools is a collection of tools for sound-field synthesis using Near-Field Compensated Higher Orders Ambisonics (NFC-HOA). For the rest of this document, the denomination Ambisonics will be used for simplicity. ambitools is developped in the context of my PhD on 3D sound field synthesis. The audio processing is coded in FAUST[1] (Functional AUdio Stream) which allows to produce efficent C++ code and exports in various DSP tools format : VST, standalone applications, LV2, etc. Thus, ambitools is multi-platform (although conceived under Linux/Jack).

The goal of ambitools is mainly to produces several modules to encode, decode and transform 3D synthesized sound field or 3D recordings in a context of physical sound field synthesis.

The project is open-source under GPL licence. ambitools is thought to be very flexible to adapt to your configuration. The FAUST-code for each tools has an header with the required order. So, for example, if

you need an encoder up to order $M = 4$, for $N = 5$ sources, you just has to change this two parameters in `hoa_encoder_N_sources.dsp` and produce de required plug-in using FAUSTLIVE[1] for example.

Don't hesitate to contact me for any suggestions, requirements, critics or even just to tell me you're using ambitools !

Pierre Lecomte

1.2 Ambisonics with ambitools overview

This section presents the basis of Ambisonics for 3D sound field synthesis. It is written with relatively few maths to focus on the concepts instead of the mathematical rigor, which would be out of scope of this document. If you want more insights to Ambisonics refer for example to [Daniel, 2000, Poletti, 2005, Ahrens, 2012].

1.2.1 Principles

Ambisonics are some mathematical approaches to represent a sound field in two or three dimensions over a basis of spherical harmonics. Without giving more mathematical details, Ambisonic main possibilities are displayed on Fig. 1.
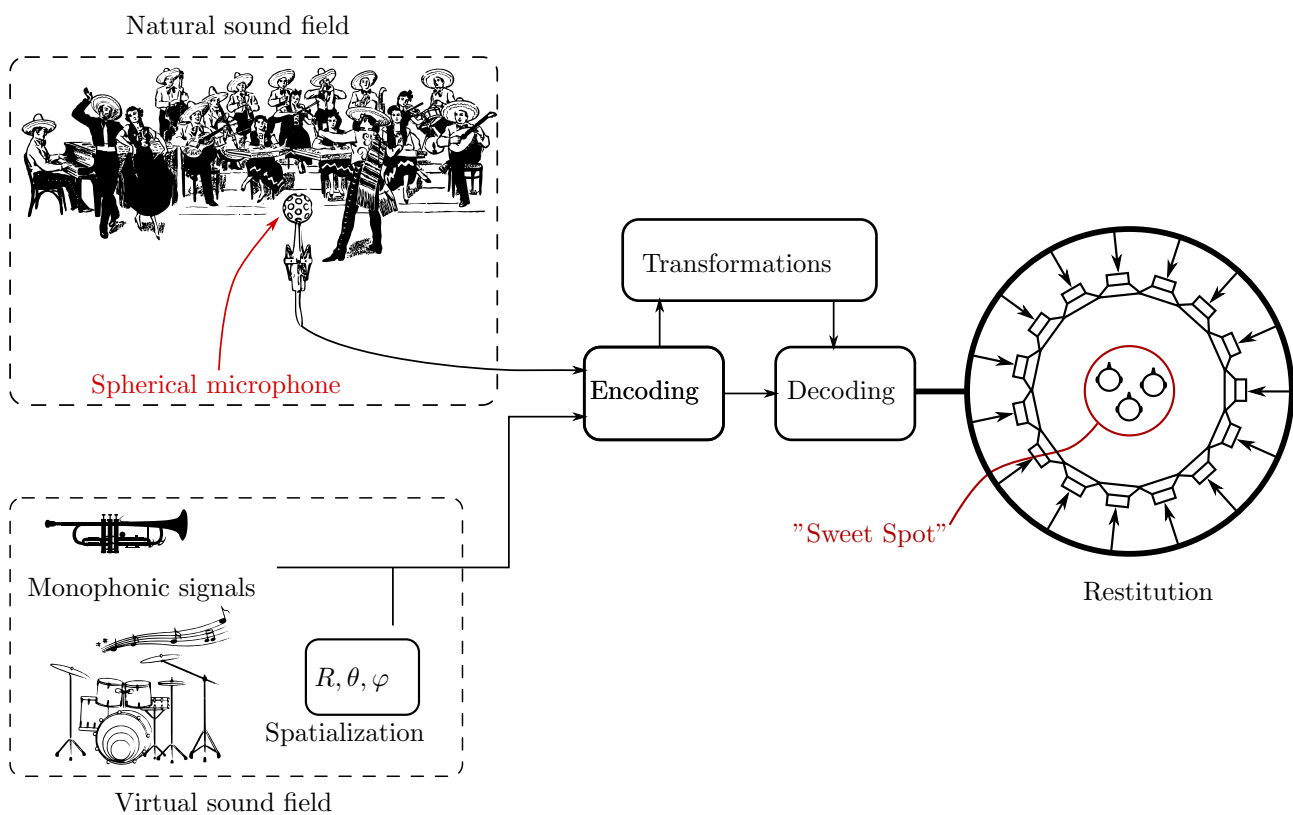

Figure 1: Ambisonic principles: A sound field can be captured in three dimensions using spherical microphone or it can be synthesized with monophonic signals and explicit positions in space. These two approaches result in an encoded sound field in Ambisonic domain up to order M. In this domain, transformations can be done such as: sound field mirroring, rotation, warping, directional filtering, etc. Finally, the decoding step computes the driving signals of the playback loudspeakers to recreate the encoded sound field in a sweet spot region of space. The decoding can be done over various configurations of speakers and even over headphones !

Thus, in accordance to this Fig. 1, ambitools provides tools to:

- Encode a sound field captured by spherical microphone.

- Encode a sound field from monophonic signals and explicit position in space.

- Transform and manipulate the encoded sound field in Ambisonic domain.

- Decode the sound field over spherical grids of speaker or over headphones with binaural convolution.

---

[1] http://faust.grame.fr

1.2.2 Spherical coordinate system

Ambisonics are usually described in spherical coordinate systems. In ambitools, the following spherical coordinate system is used: The following spherical coordinate system is used and is shown in Fig. 2:

$$x = r\cos(\theta)\cos(\delta), \ y = r\sin(\theta)\cos(\delta), \ z = r\sin(\delta). \tag{1}$$

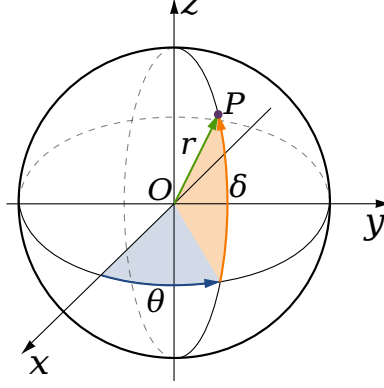The azimuth angle $\theta \in [0 \ \ 360°]$. The elevation angle $\delta \in [-90° \ \ 90°]$.

Figure 2: Spherical coordinate system in use. A point $P(x, y, z)$ is described by radius r, azimuth θ and elevation δ.

1.2.3 Spherical harmonics

As mentionned before, Ambisonics describes a sound field over a spherical harmonics basis. Those functions are directional functions of the variable θ, δ. They are denoted $Y_{mn}(\theta, \delta)$ where subscript m represent the order of the spherical harmonic and n its degree. There is several definition for these functions. In ambitools, the N3D real spherical harmonic definition is used [Daniel, 2000]:

$$Y_{mn}(\theta, \delta) = \sqrt{(2m+1)\epsilon_n \frac{(m-|n|)!}{(m+|n|)!}} P_{m|n|}(\sin(\delta))$$

$$\times \begin{cases} \cos(|n|\theta) & \text{if} \quad n \geq 0 \\ \sin(|n|\theta) & \text{if} \quad n < 0 \end{cases}, \tag{2}$$

where $P_{m|n|}$ are the associated Legendre polynomials of order m and degree $|n|$, $(m, n) \in (\mathbb{N}, \mathbb{Z})$ with $|n| \leq m$, and $\epsilon_n = 1$ if $n = 0$, $\epsilon_n = 2$ if $|n| > 0$. In Eq. (2), m denotes the spherical harmonic order and n its degree.

For each order m, there are $(2m + 1)$ spherical harmonics. Thus a basis truncated at order M contains $(M + 1)^2$ functions. For example, on Fig. 3 are displayed the spherical harmonics up to order $M = 5$ as balloon plots.

Thus, a sound field can be described as a linear combination of spherical harmonics. The weights of this linear combination are called the Ambisonics components or B-Format components. As an example, a sound field described up to order $M = 1$ is described with $(M + 1)^2 = 4$ Ambisonics components. At order $M = 5$ there will be 36 components, etc.

Without giving more details, the concept to retain is that the more components are available to describe the sound field (i.e. the higher the order), the larger the sweet-spot will be on Fig. 1.

At the moment, ambitools offers the possibility of 3D Ambisonics up to order $M = 5$. Thus for all the tools you need to compile, M SHOULD BE CHOSEN SUCH AS $M \leq 5$.
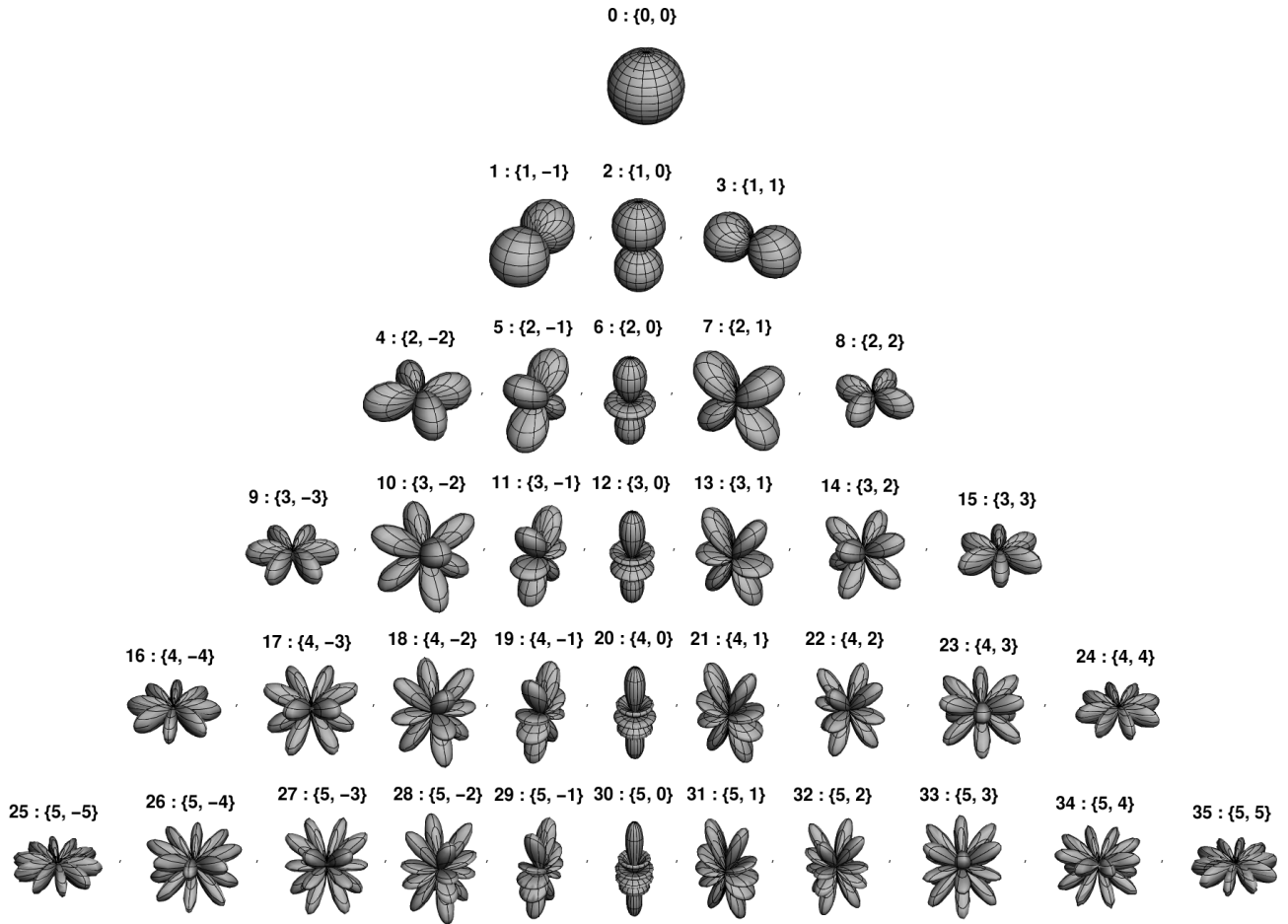
Figure 3: Balloon-plot of the spherical harmonics up to order $M = 5$.

2 Installing ambitools

2.1 Retrieve ambitools repository

To install ambitools, simply go on the github repository[2] and clone it. To do so, open a terminal in the directory you'd like to clone the repository and type the following command:

```
$ git clone https://github.com/sekisushai/ambitools
```

To keep the repository up to date, type the following command at the root of the directory `ambitools/`:

```
$ git pull
```

You can also download a `.zip` file from github[3]
The resulting `ambitools/` folder should have the following structure:

- `Documentation/` Everything concerning the documentation (pdfs, including some scientific papers).

- `Faust/` Everything written in FAUST language (all the ambitools plug-ins + some utilities).

    `bin/` Compiled plug-ins in various formats.

    `src/` Source code of the plug-ins.

    `src/lib/` Shared libraries (spherical harmonics, gui, etc.).

- `FIR/` Finite Impulse Response (FIR) filters banks for binaural rendering and spherical microphone equalization filters, to use with Jconvoler[4], fast convolution software.

    `spherical_microphones/` Equalization filters for rigid spherical microphone, such as mh acoustics EigenMike®[5]

---

[2] https://github.com/sekisushai/ambitools
[3] https://github.com/sekisushai/ambitools/archive/master.zip
[4] http://kokkinizita.linuxaudio.org/linuxaudio/
[5] http://www.mhacoustics.com

> **hrir/** Head Related Impulses Responses (HRIR) of several people to use with binaural rendering over headphones.

- **Processing/** Everything written in Processing language, namely the spherical VU-Meter ((see Sec. 3.2.1).

  > **bin/** Compiled spherical VU-Meter in Java for various architectures.

  > **src/** Processing source code.

- **PureData/** Everything written in Pure Data (a few sounds generator patches, head-tracking patch and PlayStation-like remote patch to drive Faust plug-ins with Open Sound Control protocol, OSC).

2.2 Compile the Faust plug-ins

The Faust plug-ins source codes are in the sub-folder `Faust/src/`.

2.3 Local Faust installation

If you have Faust installed on your machine with the required dependencies, you can run the scripts collection `faust2*` to produce the plug-in of your choice in the desired format.

For example, to compile `hoa_encoder_N_sources.dsp` into a standalone Linux jack-qt application with OSC support, type the following command in a terminal in the folder `\Faust/src`

```
$ faust2jaqt hoa_encoder_N_sources.dsp -osc
```

2.4 FaustLive

To compile the plug-ins to your requirements, load the chosen plug-in in FaustLive[1] and choose `Window/ Export As...` (see Fig. 4).

Figure 4: FaustLive Export Manager

2.5 Compile the Processing VU-Meter

The Processing source code is in the folder `Processing/scr`. You should open the file `Spherical_VU_Meter .pde` in the Processing editor and select "File/Export..." to produce a binary application.

3 The different tools

This section gives a quick presentation of each tool contained in ambitools.

3.1 Faust

The core of the sound processing is written in Faust language. Note that the majority of the figures presented here will be screenshots of the tools compiled as standalone Jack applications for Linux, using `faust2jaqt` script. However, thanks to the versatility of Faust language, remember that each of these tools can be compiled for various architecture, using FaustLive[1] for example.

3.1.1 hoa_N_sources_encoder

- Inputs: N

- Outputs: $(M+1)^2$

This first tool allows to encode N signal inputs to an 3D Ambisonics scene described with Ambisonics components signals up to order M (the B-Format). You can choose N and M at the compilation time in file `hoa_encoder_N_sources.dsp`. The resulting graphical user interface when using `faust2jaqt` script is shown in Fig. 5 for $N = 3, M = 5$. Each input signal can be encoded in B-Format as a plane wave or a spherical wave.

For the plane wave case, the check-box `Spherical Wave` should be unchecked. Consequently, the knob `Radius` and input entry `Speakers Radius` are without effect in this case (a plane wave has no distance information).

For the spherical wave case, the knob `Radius` allows to choose the source radius to origin. Note that to represent a spherical wave in Ambisonic domain, near-field filters are activated [Daniel, 2003, Lecomte and Gauthier, 2015]. Those filters, unstable by nature, are stabilized using the near-field compensation filters from decoding step. Thus in this case, the radius of the spherical loudspeakers grid should be known prior to decoding and given in `Speakers Radius` input entry.

Finally, the outputs of this tool are B-Format signals up to order M. Each order can be muted or each individual Ambisonic components individually.

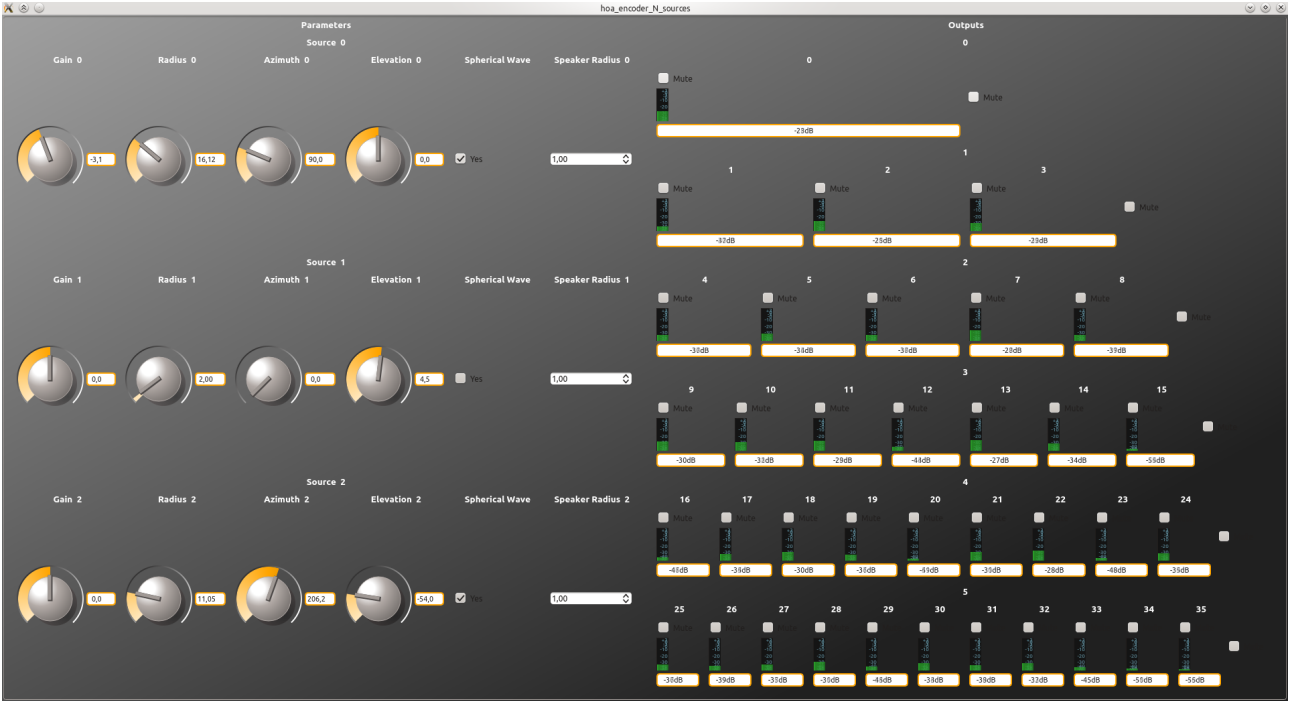Figure 5: `hoa_encoder_N_sources` plug-in compiled using `faust2jaqt` script with $N = 2$ and $M = 5$. The `Gain` knob allows to adjust the input level of the input. `Radius` knob allows to choose the source distance to origin in case of spherical wave encoding. `Azimuth` and `Elevation` knobs allow to choose the source direction. The `Spherical Wave` check-box enables the encoding of spherical wave, using near-field filters. `Speakers Radius` sets the radius for the spherical arrays of loudspeakers at decoding stage in case of spherical wave encoding. Finally, the VU-Meters shows the level of 3D B-Format signals in dBFS up to order M. Each meter as a `Mute` check-box and each order as well.

3.1.2 hoa_decoder_*

- Inputs: $(M + 1)^2$

- Outputs: 26 or 50 depending on the decoder.

Two basic decoder by mode-matching [Daniel, 2000, Poletti, 2005] are available at the moment in ambitools: `hoa_decoder_lebedev26` and `hoa_decoder_lebedev50`. Those two decoder allows to decode $(M+1)^2$ B-Format signals on Lebedev grids with respectively 26 and 50 loudspeakers [Lebedev, 1975, Lecomte et al., 2015]. Those grids are able to reconstruct the sound field up to order $M = 3$ and $M = 5$ respectively. If other decoder are required, you should have a look at the ambisonic decoder toolbox from Aaron Heller [Heller et al., 2012], or contact me. The graphical user interface when using `faust2jaqt` script is shown in Fig. 6 for the `hoa_decoder_lebedev50` decoder compiled with $M = 5$. The decoder can be with or without near-field compensation (NFC) filters [Daniel et al., 2003, Lecomte and Gauthier, 2015]. Those filters allow to take into account the finite distance of the loudspeakers: In other terms, if they are disabled, the loudspeakers are modeled as plane-wave generators. In case of spherical wave encoding using the `hoa_N_sources_encoder` plug-in, the `NFC` check-box should be unchecked as the near-field compensation filters are already used at encoding step (see Sec. 3.1.1).
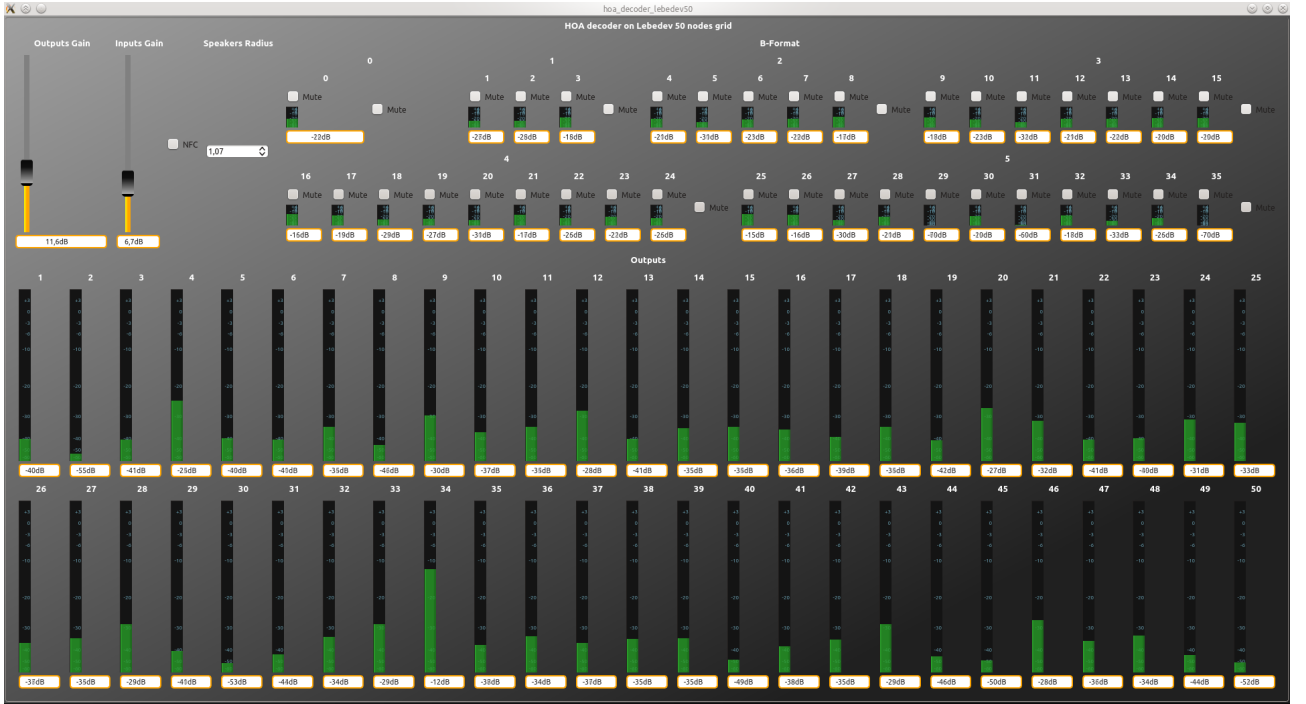

Figure 6: `hoa_decoder_lebedev50` compiled using `faust2jaqt` script with $M = 5$. The slider `Outputs Gain` apply a global gain on all outputs (loudspeakers signals). The slider `Inputs Gain` apply a global gain on all inputs (B-Format signals). The VU-Meters `Inputs` and `Outputs` give the signals level in dBFS. The check-box `NFC` activate or deactivate the near-field compensation filters. The input entry `Speakers Radius` allows to set the spherical grid radius. Finally, the check-boxes `Mute` above all inputs VU-meters allow to mute some specific B-Format signals. Or, all the signal from an order with `Mute` check-boxes on side of an VU-Meter group.

3.1.3 hoa_panning_*

- Inputs: N

- Outputs: 26 or 50 depending on the configuration.

It is possible to compute directly the driving signals of the loudspeakers without passing by and encoding/decoding process [Lecomte et al., 2015]. This equivalent 3D panning is implemented in the tools `hoa_panning_lebedev26` and `hoa_panning_lebedev50` for 26 and 50 loudspeakers Lebedev grids respectively. The graphical user interface using FAUST Linux jack-qt compiler is shown in Fig. 7 for the `hoa_panning_lebedev50` tool. For this version, two spherical waves are synthesized.
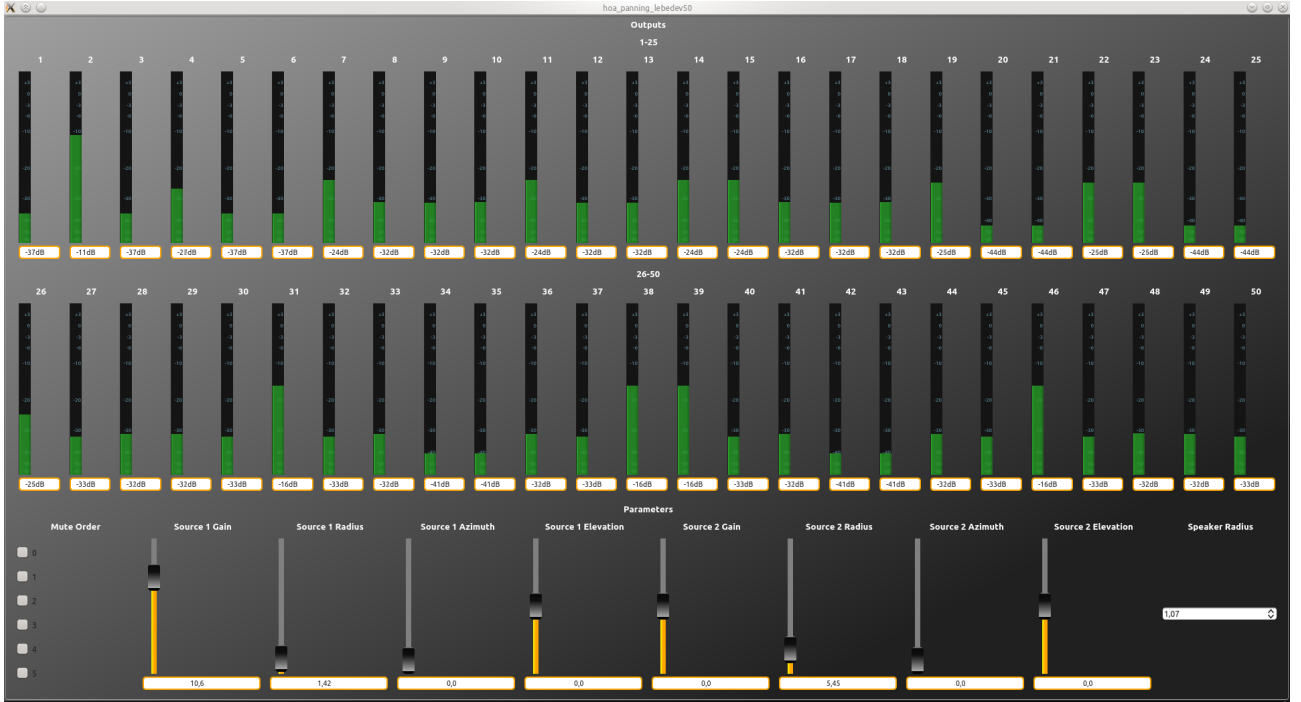

Figure 7: `hoa_panning_lebedev50` plug-in under Linux jack-qt4. The slider 'Outputs Gain' apply a global gain on all outputs (loudspeakers signals). The check-boxes 'Mute Order' allows to mute some Ambisonic orders in the computation of the driving signals. The sliders 'Gain' 'Radius' 'Azimuth' 'Elevation' controls the position and gain of two sources (spherical waves). In order to stabilize the near-field filters for spherical wave synthesis, the input entry 'Speaker Radius' fixes the loudspeaker array radius.

3.1.4 hoa_mirroring

The plug-in `hoa_mirroring` allows a sound field transformation in Ambisonic domain. Thus, it should be inserted somewhere in between encoding and decoding steps. The sound field is here flipped upside-down, left-right or front-back, or any combination of the above. The tool changes the sign of particular Ambisonic components to realize the transformation [Kronlachner and Zotter, 2014]. The graphical user interface using Faust Linux jack-qt compiler is shown in Fig. 8. An example of plug-in insertion under Linux jack is shown in Fig. 9.
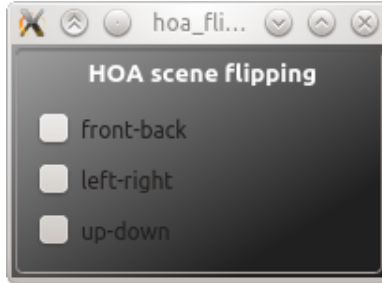

Figure 8: `hoa_mirroring` plug-in under Linux jack-qt4. The check-boxes allows to select the mirror transformation : 'left-right', 'front-back' or 'up-down'.
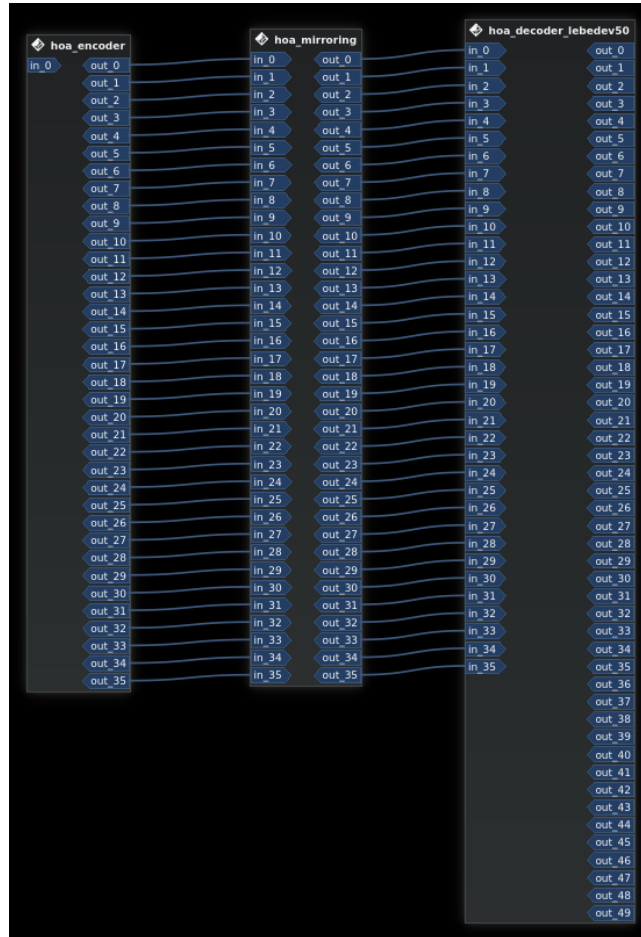

Figure 9: Example of `hoa_mirroring` insertion between encoding and decoding steps, i.e. in Ambisonic domain.

3.1.5   hoa_azimuth_rotator

3.1.6   hoa_beamforming_hypercardioid_to_mono

3.1.7   hoa_beamforming_hypercardioid_to_hoa

3.1.8   hoa_beamforming_dirac_to_hoa

3.2 Processing

3.2.1 Spherical_VU_Meter

ambitools provides a Spherical VU-Meter developped with PROCESSING language. A screen-shot is shown on Fig. 10. This tool allows to "see" where the sound energy is in space, instead of using classical in-line VU-Meter. You can zoom, pan and rotate the view of the meter while running. The FAUST tools `hoa_panning*` and `hoa_decoder*` emit OSC messages on port UDP 5511. Those messages drive the spherical VU-Meter rendering in real-time: loudspeakers size and color for the meter, and source position.
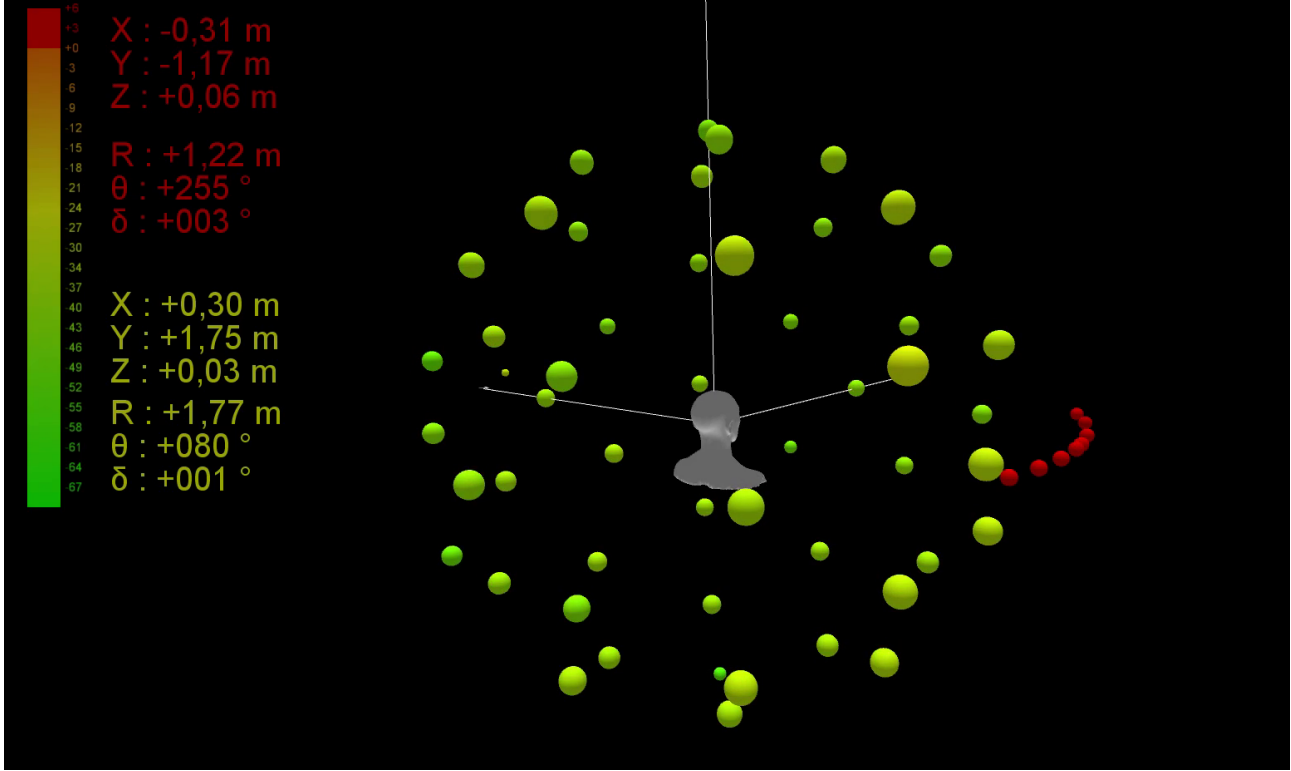

Figure 10: `Spherical_VU_Meter` for a 50 loudspeaker Lebedev array and two virtual sources. Each loudspeaker is represented as a color ball with size and color proportional to RMS Level in dBFS. A scale in dBFS is displayed on the left of the screen. The virtual sources are represented as red an yellow dots. Their coordinates are displayed in Cartesian and spherical coordinates on the left. A grey manikin is standing in the middle of the array to indicates the front direction.

3.3 Jconvolver

3.3.1 jconvolver_mic*

3.3.2 hrir_lebedev50

References

[Ahrens, 2012] Ahrens, J. (2012). *Analytic Methods of Sound Field Synthesis*. Springer.

[Daniel, 2000] Daniel, J. (2000). *Représentation de champs acoustiques, application à la transmission et à la reproduction de scènes sonores complexes dans un contexte multimédia*. PhD thesis, Université Paris 6, Paris.

[Daniel, 2003] Daniel, J. (2003). 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*, pages 1–15, Helsingør. Audio Engineering Society.

[Daniel et al., 2003] Daniel, J., Moreau, S., and Nicol, R. (2003). Further investigations of high-order ambisonics and wavefield synthesis for holophonic sound imaging. In *Audio Engineering Society Convention 114*, pages 1–18, Amsterdam. AES.

[Heller et al., 2012] Heller, A. J., Benjamin, E. M., and Lee, R. (2012). A toolkit for the design of ambisonic decoders. *Linux Audio Conference*.

[Kronlachner and Zotter, 2014] Kronlachner, M. and Zotter, F. (2014). Spatial transformations for the enhancement of Ambisonic recordings. In *2nd International Conference on Spatial Audio*, Erlangen.

[Lebedev, 1975] Lebedev, V. (1975). Values of the nodes and weights of quadrature formulas of Gauss-Markov type for a sphere from the ninth to seventeenth order of accuracy that are invariant with respect to an octahedron. *USSR Computational Mathematics and Mathematical Physics*, 15(1):44–51.

[Lecomte and Gauthier, 2015] Lecomte, P. and Gauthier, P.-A. (2015). Real-Time 3D Ambisonics using Faust, Processing, Pure Data, And OSC. In *15th International Conference on Digital Audio Effects (DAFx-15)*, Trondheim.

[Lecomte et al., 2015] Lecomte, P., Gauthier, P.-A., Langrenne, C., Garcia, A., and Berry, A. (2015). On the use of a Lebedev grid for Ambisonics. In *Audio Engineering Society Convention 139*, New York.

[Poletti, 2005] Poletti, M. A. (2005). Three-dimensional surround sound systems based on spherical harmonics. *Journal of the Audio Engineering Society*, 53(11):1004–1025.