Real-time 3D Ambisonics using Faust, Processing, Pure Data, and OSC

Pierre Lecomte, Philippe-Aubert Gauthier

REAL-TIME 3D AMBISONICS USING FAUST, PROCESSING, PURE DATA, AND OSC

Pierre Lecomte,

Lab. de Mécanique des Structures et Systèmes Couplés (LMSSC), Conservatoire National des Arts et Métiers
Paris, France
Groupe d'Acoustique de l'Université de Sherbrooke (GAUS), Université de Sherbrooke, Québec, Canada
pierre.lecomte@gadz.org

Philippe-Aubert Gauthier,

GAUS
Université de Sherbrooke, Québec, Canada

ABSTRACT

This paper presents several digital signal processing (DSP) tools for the real-time synthesis of a 3D sound pressure field using Ambisonics technologies. The spatialization of monophonic signal or the reconstruction of natural 3D recorded sound pressure fields is considered. The DSP required to generate the loudspeaker signals is implemented using the FAUST programming language. FAUST enables and simplifies the compilation of the developed tools on several architectures and on different DSP tool format. In this paper, a focus is made on the near-field filters implementation which allows for the encoding of spherical waves with distance information. The gain variation with distance is also taken into account. The control of the synthesis can be made by software controllers or hardware controllers, such as joystick, by the mean of PURE DATA and OPEN SOUND CONTROL (OSC) messages. A visual feedback tool using the PROCESSING programming language is also presented in the most recent implementation. The aim of this research derives from a larger research project on physically-accurate sound field reproduction for simulators in engineering and industrial applications.

1. INTRODUCTION

Ambisonics technologies allow describing 3D sound pressure fields using a projection on a truncated spherical harmonics basis [1]. The resulting 3D encoded sound pressure field can later be manipulated, decoded and reconstructed over several loudspeaker-layouts or even headphones [2]. Ambisonics has two main objectives: the spatialization of virtual sound sources or the reconstruction of recorded sound pressure fields [3]. Several software solutions exist to create, transmit, manipulate, and render sound pressure fields using Ambisonics technologies. See references [4, 5, 6, 7, 8] as examples. Albeit being popular for practical applications in music, sound design and audio context, classical and common Ambisonics implementations suffer from few drawbacks that limit their use for physically-accurate sound field reproduction with applications to environment simulators (vehicles, working environments, etc.) in industrial or engineering context. Indeed, the near-field encoding [2] is rarely provided and the encoding/decoding in 3D context is limited to the first orders, hence limiting the spatial resolution and area size of physically-accurate reproduction. Indeed, if the sound field is controlled up to an order M, the reconstruction area size is frequency-dependent and given by $r = Mc/2\pi$ [9] (where r is the are size radius, c the speed of sound in air, and f the frequency). The near-field support is also detrimental for physically-accurate sound field reproduction as it takes into account the loudspeaker distance from the origin in order to compensate for the loudspeakers spherical waves. In this trend, this work is motivated by the need to develop a practical implementation of Ambisonics for industrial applications such as laboratory reproduction of industrial sound environments, vehicles cabin noise, virtual vibroacoustics models, architectural spaces, etc. In these scenarios, the reproduced sound field must be as close as possible than the target sound field. Some recent examples of such applications potentials are found in Refs. [10, 11, 12, 13]. Typical outcomes are related to listening tests, sound quality testing, perceptual studies and other. On this matter, as mentioned by Vorländer [12] with respect to Ambisonics implementations that often include modifications inspired by auditory perception to increase the listener experience with respect to some expectations, a "generally applicable reproduction system [for sound field simulation] must not introduce any artificial auditory cue which is not part of the simulated sound." [12].

In this context, this paper presents an implementation of Ambisonics technologies for real-time synthesis of 3D sound field up to 5^{th} order. The signal processing is implemented in FAUST[1] (Functional AUdio STream) programming language. This language proposes a functional approach to implement the signal processing and it compiles in efficient C++ code [14]. From this code, DSP tools are provided in several formats such as: VST, LV2, Pure Data, Max/MSP, JACK QT, and others. Thus, from the same FAUST code, one can generate tools working on various operating systems and configurations.

The focus of this paper is on physical encoding and decoding of 3D sound pressure fields with a special attention dedicated to the near field filters development, definition and implementation. After defining the notations in use in Sec. 2, the main equations of Ambisonics are recalled in Sec. 3. In Sec. 4 the implementation in FAUST programming language is addressed with a special attention on near-field filters. Section 5 presents a visual feedback tool using PROCESSING language. This tool helps visualizing in 3D the virtual sources and the loudspeaker levels. Finally, in Sec. 6, the user control interface is addressed, presenting the possibility of interfacing all elements with OSC protocol.

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

2. COORDINATE SYSTEM AND NOTATIONS

In this section, the different notations used throughout the paper are presented and illustrated.

Spherical coordinate system

The following spherical coordinate system is used throughout this paper and shown in Fig. 1:

$$u_1 = r\cos(\theta)\cos(\delta),\ u_2 = r\sin(\theta)\cos(\delta),\ u_3 = r\sin(\delta) \tag{1}$$

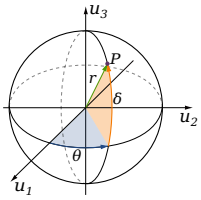

Figure 1: Spherical coordinate system. A point $P(u_1, u_2, u_3)$ is described by its radius r, azimuth θ and elevation δ.

A virtual source position is denoted with its spherical coordinates r_1, θ_1, δ_1. The rendering loudspeakers are arranged on a sphere of radius r_0, as shown in Fig. 2.

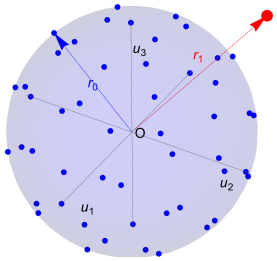

Figure 2: Ambisonics playback layout. Blue: the rendering loudspeakers disposed on a spherical layout of radius r_0. Red: the virtual source at radius position r_1.

Spherical harmonics

The spherical harmonics used in this paper are defined as follow in [1]:

$$Y^{\sigma}_{mn}(\theta,\delta) = \sqrt{(2m+1)\epsilon_n \frac{(m-n)!}{(m+n)!}} P_{mn}(\sin(\delta)) \times \begin{cases} \cos(n\theta) & \text{if} \quad \sigma = 1 \\ \sin(n\theta) & \text{if} \quad \sigma = -1 \end{cases} \tag{2}$$

where P_{mn} are the associated Legendre functions of order m and degree n, m and $n \in \mathbb{N}$ with $n \leq m$, $\sigma = \pm 1$ and $\epsilon_n = 1$ if $n = 0$, $\epsilon_n = 2$ if $n > 0$. The spherical harmonics order are denoted by m and its degree, by n. For each order m there are $(2m+1)$ spherical harmonics. Thus a basis truncated at order M contains $L = (M+1)^2$ functions.

Notations

The Laplace variable is denoted s and the discrete time variable z (discrete domain). A vector is denoted by lowercase bold font $\mathbf{v}$ and a matrix by uppercase bold font $\mathbf{M}$. Superscript ^{T} designates the transposition operation. j is imaginary unit with $j^2 = -1$.

3. AMBISONICS

In this section, the principal Ambisonics equations are recalled. They will later be used for the real-time DSP implementation.

3.1. Encoding

In Ambisonics, the encoding step consists in deriving *B-Format* signals [2] from either monophonic signal (with a spatialization context) or microphone array signals (natural sound field encoding).

3.1.1. Monophonic signal

From a monophonic signal, the encoding can be done for a plane wave with amplitude $a(z)$, azimuth θ_1 and elevation δ_1 direction or a spherical wave, adding a distance information r_1

$$\begin{aligned} B^{\sigma}_{mn}(z) &= a(z) Y^{\sigma}_{mn}(\theta_1, \delta_1) && \text{Plane wave} \\ B^{\sigma}_{mn}(z) &= a(z) F_{m,r_1}(z) Y^{\sigma}_{mn}(\theta_1, \delta_1) && \text{Spherical wave} \end{aligned} \tag{3}$$

In Eq. 3, filters $F_{m,r_1}(z)$ are the forward near-field filters which take into account the finite distance of the virtual source r_1 [2].

3.1.2. Rigid spherical microphone array encoding

For a natural 3D sound pressure field recording made with a rigid spherical microphone array of radius r_a, the B^{σ}_{mn} are given by:

$$B^{\sigma}_{mn}(z) = E_{m,r_a}(z) \sum_{i=1}^{N} Y^{\sigma}_{mn}(\theta_i, \delta_i) w_i p_i(z) \tag{4}$$

$E_{m,r_a}(z)$ are the equalization filters which take into account the diffraction of the rigid sphere [3]. The sound pressure signal at the i^{th} capsule position $(r_a, \theta_i, \delta_i)$ is denoted $p_i(z)$ on the array of N microphones. The B^{σ}_{mn} components are guaranteed to be exact up to order M if the spatial sampling of the spherical microphone array respects the orthonormality criterion of spherical harmonics up to M [15, 16]. Thus, there is possibly a weighting factor w_i for each capsule in Eq. (4) to ensure this condition. The working bandwidth of the array without aliasing is given by: $f \leq Mc/(r_a 2\pi)$ [17], where f is the frequency, c the sound speed, and M the maximum working order for the array.

Equation (4) is for a triplet of indices (m, n, σ). Thus, up to order M, the *B-Format* signals vector is obtained with matrix notation:

$$\mathbf{b}(z) = \mathbf{E}(z) \cdot \mathbf{Y}^T_{\text{mic}} \cdot \mathbf{W}_{\text{mic}} \cdot \mathbf{p}(z) \tag{5}$$

$\mathbf{b}(z)^{(L\times 1)} = [B^1_{00}(z) \cdots B^{\sigma}_{mn}(z) \cdots B^1_{M0}(z)]$. $\mathbf{E}(z)^{(L\times L)}$ is the diagonal matrix of equalization filters with diagonal terms $[E_{0,r_a}(z) \cdots E_{m,r_a}(z) \cdots E_{M,r_a}(z)]$, each $E_{m,r_a}(z)$ term being replicated $(2m+1)$ times on the main diagonal. $\mathbf{Y}^{(N\times L)}_{\text{mic}}$ is the matrix of spherical harmonics up to order M evaluated at each direction (θ_i, δ_i). $\mathbf{W}^{(N\times N)}_{\text{mic}}$ is the diagonal matrix of weightings.

[2] We call here the *B-Format* the signals vector $\mathbf{b}(z)$ $[B^1_{00}(z) \cdots B^{\sigma}_{mn}(z) \cdots B^1_{M0}(z)]$

$\mathbf{p}^{N\times 1} = [p_1(z)\cdots p_i(z)\cdots p_N(z)]$ is the vector of capsule signals.

3.2. Decoding

In this paper, only the basic decoder (mode-matching [18]) is recalled for a full-sphere configuration of radius r_0 respecting the spherical harmonics orthonormality via weighting coefficients [16]. For decoders adapted to irregular loudspeaker layout or other decoding concerns, see for example [6].

If one considers a set of N_2 loudspeakers, the input signal of the l^{th} loudspeaker at position $(r_0, \theta_l, \delta_l)$ is obtained from the B^{σ}_{mn} signals by:

$$s_l(z) = w_l \sum_{m=0}^{M} (F_{m,r_0})^{-1}(z) \sum_{n=0}^{m} \sum_{\sigma=\pm 1} Y^{\sigma}_{mn}(\theta_l, \delta_l) B^{\sigma}_{mn}(z) \tag{6}$$

where $(F_{m,r_0})^{-1}(z)$ are the inverse near field filters or near field compensation filters, which take into account the finite distance r_0 of the rendering loudspeakers [2]. Since the reconstructed sound pressure field is the summation of each sound field generated by each loudspeakers, the vector of input signals $\mathbf{s}$ is given by:

$$\mathbf{s} = \mathbf{W}_{\text{spk}} \cdot \mathbf{Y}_{\text{spk}} \cdot \mathbf{F}^{-1}_{r_0}(z) \cdot \mathbf{b}(z) \tag{7}$$

where $\mathbf{s}(z)^{(N_2\times 1)} = [s_1(z)\cdots s_l(z)\cdots s_{N_2}(z)]$. $\mathbf{F}^{-1}_{m,r_0}{}^{(L\times L)}(z)$ is the diagonal matrix of near field compensation filters with diagonal terms $[1/F_{0,r_0}(z)\cdots 1/F_{m,r_0}(z)\cdots 1/F_{M,r_0}(z)]$, each $1/F_{m,r_0}(z)$ term being replicated $(2m+1)$ times on the main diagonal. $\mathbf{Y}^{(N_2\times L)}_{\text{spk}}$ is the matrix of L spherical harmonics up to order M evaluated at each direction (θ_l, δ_l). $\mathbf{W}^{(N_2\times N_2)}_{\text{spk}}$ is the diagonal matrix of weightings w_l. Note that the resulting matrix $\mathbf{M}_{\text{spk}} = \mathbf{W}_{\text{spk}} \cdot \mathbf{Y}_{\text{spk}}$ is the Ambisonics decoding matrix as defined in Ref.[6].

3.3. Equivalent panning-law

In a spatialization context, recalling Eq. (3) and Eq. (6) along with the additivity theorem of spherical harmonics [19], one can directly compute the loudspeaker input signals:

$$\begin{aligned} s_l(z) &= a(z) w_l \sum_{m=0}^{M} \frac{(2m+1)}{F_{m,r_0}(z)} P_m(\gamma_l) && \text{Plane wave} \\ s_l(z) &= a(z) w_l \sum_{m=0}^{M} (2m+1) \frac{F_{m,r_1}}{F_{m,r_0}}(z) P_m(\gamma_l) && \text{Spherical wave} \end{aligned} \tag{8}$$

where $\gamma_l = \cos(\delta_1)\cos(\delta_l)\cos(\theta_1 - \theta_l) + \sin(\delta_1)\sin(\delta_l)$ is relative to the angle between the virtual source in θ_1, δ_1 and the l^{th} loudspeaker.

4. FAUST IMPLEMENTATION

The implementation of Ambisonics tools such as an encoder, basic decoder, near-field filters, and spatialization tools using equivalent panning-law is made using the FAUST language. An overview of the current developped tools is shown in Fig.3. From left to right, the vertical branches correspond to: 1) microphone signal processing, 2) panning of a monophonic signal as virtual point source with encoding and decoding and 3) panning of a monophonic signal as virtual point source with equivalent panning law. The rightmost branch is the control via OSC and visual feedback. Each box corresponds to a module described in the next sections. The common functions of each modules are implemented in library files, which enable the quick design of new modules by re-using existing FAUST code. In the following section, the near-field filters implementation is detailed.

4.1. Near-field filters

4.1.1. Forward filters

The forward filters are given in the Laplace domain [2]:

$$F_{m,r_1}(s) = \sum_{i=0}^{m} \frac{(m+i)!}{(m-i)!i!2^i} \left(\frac{c}{sr_1}\right)^i \tag{9}$$

with $s = j2\pi f$. To convert this analog filter in the digital z domain, the use of a bilinear transform requires precision arithmetic to be efficient, as pointed out by Adriaensen in [20]. In this latter reference, he proposes another s-to-z mapping scheme to obtain a digital realization of the filter which is robust with single precision floating point format:

$$\begin{aligned} \zeta^{-1} = \frac{z^{-1}}{1-z^{-1}} = \sum_{k=1}^{\infty} (z^{-1})^k \quad \text{for} |z| > 1 \\ s = \frac{2F_s}{1+2\zeta^{-1}} \end{aligned} \tag{10}$$

where F_s is the sampling frequency. The implementation of an m^{th} order forward filter is made by product of sections of the form:

$$\begin{aligned} H_{1,r_1}(z) &= g_1(r_1)(1 + d_{1,1}(r_1)\zeta^{-1}) \\ H_{2,r_1}(z) &= g_2(r_1)(1 + d_{2,1}(r_1)\zeta^{-1} + d_{2,2}(r)\zeta^{-2}) \end{aligned} \tag{11}$$

For example a 3rd order filter F_{3,r_1} is realized with the product of a 1st order section and a 2nd order section: $F_{3,r_1}(z) = H_{1,r_1}(z) \cdot H_{2,r_1}(z)$. The computation of each coefficient g_1, g_2, $d_{1,1}$, $d_{2,1}$, $d_{2,2}$ in Eq. (11) is detailed in [20].

4.1.2. Stabilization with inverse filters

The forward filters $F_{m,r_1}(z)$ present an infinite gain at 0 Hz. Thus, they must be stabilized by multiplying with an inverse filter (or near field compensation filters) $1/F_{m,r_0}(z)$. This can be done at the encoding stage as suggested by Daniel [2]:

$$B^{\sigma}_{mn}(z) = a(z) \frac{F_{m,r_1}}{F_{m,r_0}}(z) Y^{\sigma}_{mn}(\theta_1, \delta_1) \quad \text{Spherical wave} \tag{12}$$

It means that to encode a spherical wave, according to Eq. (12), one should know the rendering array radius r_0 *a priori*. However this is not a major concern. Indeed, if the encoded spherical wave is reconstituted on another layout of radius r_2, one can correct by multiplying the $B^{\sigma}_{mn}(z)$ components by $F_{m,r_0}(z)/F_{m,r_2}(z)$.

4.1.3. Gain correction for spherical source

In [2], the near field filters are defined with the assumption that the amplitude $a(z)$ of the spherical wave in Eq. (12) is taken at the origin O in Fig. 2. Thus, the propagation term $e^{sr_1/c}/(4\pi r_1)$

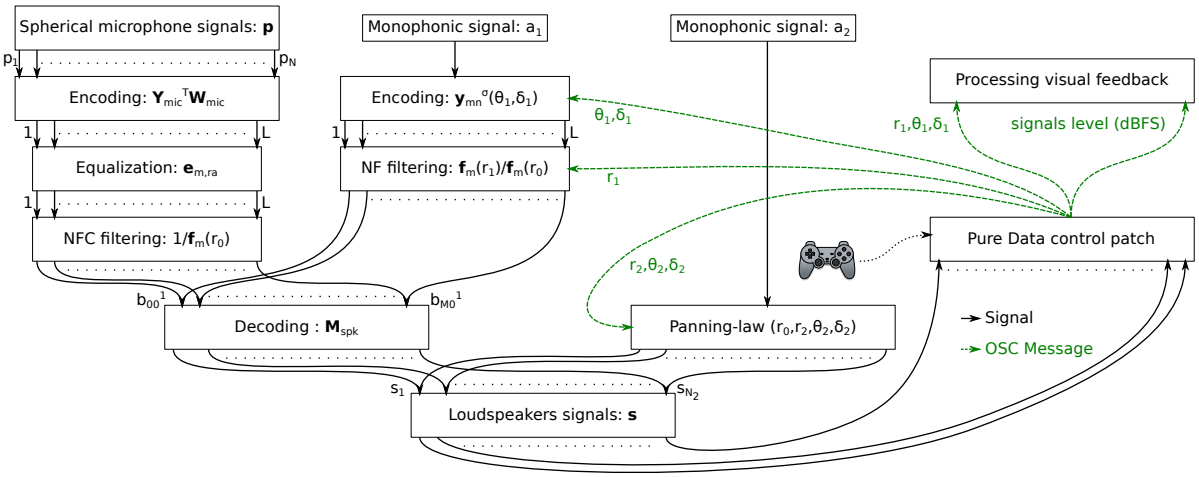

Figure 3: Main features of the current implementation: real-time synthesis of 3D recordings (leftmost vertical branch), spatialization (two centered vertical branches), and control via OSC with visual feedback (rightmost vertical branch). The digital temporal signals are represented by black arrows and the OSC instructions message in green dashed arrows.

(Laplace's domain) is not taken into account. This leads to the definition of Eq. (9) for the forward filters. In the same way, the near-field compensation filters do not take into account the propagation from rendering source to origin O: $e^{sr_0/c}/(4\pi r_0)$. The main consequence is that there is no gain variation with the distance of the virtual source. To correct this, one multiplies the amplitude $a(z)$ for spherical wave in Eq. (12) by: $\frac{r_0}{r_1} z^{-(r_1-r_0)/c}$. The corresponding delay can be fractional depending on r_1 and r_0. In the case of a focused source (i.e. inside the rendering loudspeakers enclosure), the delay is negative and an implementation could require time-reversal method, as in Wave Field Synthesis [21]. However, since this delay is the same for all rendering loudspeakers, according to Eqs. (6) and (8), it is omitted for simplicity. As a result, the pressure sound field of a virtual point source will be reproduced physically in the reproduction zone with correct gain and with a phase shift of $z^{-(r_1-r_0)/c}$ when this latter is omitted. Finally, the encoding and the panning-law of a spherical wave becomes:

$$\begin{aligned} B^{\sigma}_{mn}(z) &= a(z)\frac{r_0}{r_1}\frac{F_{m,r_1}}{F_{m,r_0}}(z)Y^{\sigma}_{mn}(\theta_1,\delta_1) \\ s_l &= a(z)\frac{r_0}{r_1}w_l\sum_{m=0}^{M}(2m+1)\frac{F_{m,r_1}}{F_{m,r_0}}(z)P_m(\gamma_l) \end{aligned} \tag{13}$$

4.1.4. FAUST *implementation*

The near field filters $F_{m,r_1}(z)/F_{m,r_0}(z)$ (as well as the near field compensation filters $1/F_{m,r_0}(z)$) are implemented in a FAUST library `nfc.lib` following Eq. (11) and up to 5^{th} order. The implementation is based on a Direct-Form II [22]. As an example, the FAUST code for the second order section is given as:

```
TFNF2(b0,b1,b2,a1,a2) =
sub~sum1(a1,a2): sum2(b0,b1,b2)
        with {
        sum1(k1,k2,x)=
x:(+~_<:((_':+~_),*(k1))):*(k2),_:+);
        sum2(k0,k1,k2,x)=
x<:*(k0),+~_,_:_,(-<:*(k1),(_':+~_)
*(k2):+):+;
        sub(x,y)= y-x;
};
```

The block diagram for this code is shown in Fig. 4. The coefficients $g_2(r_1), d_{2,1}(r_1), d_{2,2}(r_1)$, $g_2(r_0), d_{2,1}(r_0), d_{2,2}(r_0)$ were precomputed in this figure for simplicity to obtain the corresponding $b_0(r_1), b_1(r_1), b_2(r_1), a_1(r_0), a_2(r_0)$ coefficients, poles an zeroes of the filter. However, `nfc.lib` provides the real-time computation of these coefficients knowing r_1 and r_0, according to [20]. In the current implentation, the near-field filters are provided as independent modules (see Fig. 3), or included in the equivalent panning-law module (see Sec. 4.4)

As an example, the gain frequency response of the filters $r_0/r_1 \times F_{m,r_1}/F_{m,r_0}$ up to order five are shown in Fig. 5 for $r_1 = 1$ m, $r_0 = 3$ m (solid lines) and $r_1 = 3$ m, $r_0 = 1$ m (dashed lines). $c = 340$ m/s and $F_s = 48000$ Hz. For focused sources (i.e $r_1 \leq r_0$, solid lines in Fig. 5), as r_1 decreases, or as r_0 increases and as m increases, the gain of the filters increases at low frequencies. Thus, when encoding a focused spherical source (i.e. $r_1 \leq r_0$), one should be aware of these extreme gains. These "bass-boost" effects create strong artifacts outside the control zone and can easily damage the rendering loudspeakers. A solution could be to impose a minimum r_1 regarding to maximum a $r_0/r_1 \times F_{m,r_1}/F_{m,r_0}$ gain. This maximum gain is then related to the maximum linear excursion of the loudspeakers. Note that another approach for focused sources in Ambisonics can as well be a solution [23].

4.2. Encoding of captured sound field and decoding

This case corresponds to the leftmost vertical branch of Fig. 3. The encoding of a 3D sound field captured by a spherical microphone array as described in Eq. (5) requires a matrix operation as the decoding operation of Eq. (7). This matrix operation is done in FAUST as suggested by Heller [6]:

```
// bus with gains
gain(c) = R(c) with {
  R((c,cl)) = R(c),R(cl);
  R(1)      = _;
```

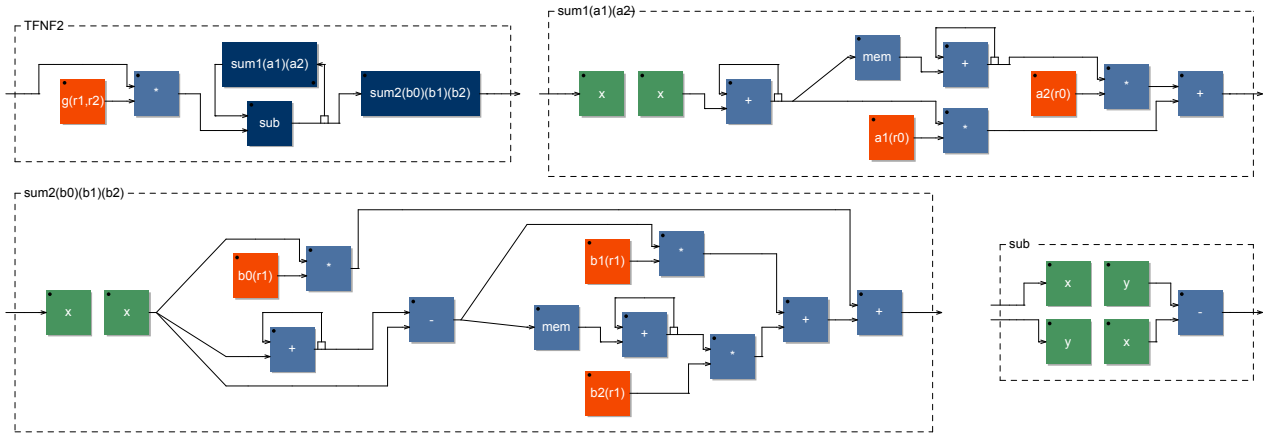

Figure 4: Block diagram representation of the `TFNF2` function. The input signal to be filtered is on the main top left diagram. The others diagrams detail each block in this main diagram. `x` and `y` are input signals in a block. These diagrams where generated using `faust2svg` tool.

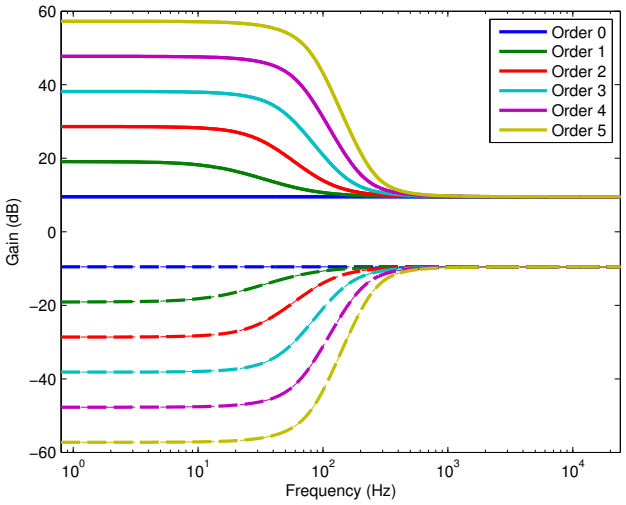

Figure 5: Gain of the frequency response function of filters $\frac{r_0}{r_1}\frac{F_{m,r_1}}{F_{m,r_0}}(f)$ for $r_1 = 1$ m, $r_0 = 3$ m (solid lines) and $r_1 = 3$ m, $r_0 = 1$ m (dashed lines), $m \in \{0, 1, 2, 3, 4, 5\}$, $F_s = 48000$ Hz.

```
  R(0)        = !;
  R(float(0)) = R(0);
  R(float(1)) = R(1);
  R(c)        = *(c);
};

matrix(n,m) = par(i,n,_)
<: par(i,m,gain(a(i)):>_);
// n: number of inputs : column number
// m: number of output : row number
// a(i): row vectors of matrix coefficients
```

In Eqs. (5) and (7) the matrix $\mathbf{Y}_{\text{mic}}^T \cdot \mathbf{W}_{\text{mic}}$ and $\mathbf{W}_{\text{spk}} \cdot \mathbf{Y}_{\text{spk}}$ are pre-computed and then implemented numerically in the FAUST code, row by row, according to the code above. In the current version, the encoding matrix for 2 types of microphone is coded:

- Spherical microphone using Lebedev's grids working up to 1^{st}, 3^{rd} or 5^{th} order as shown in Fig. 9.
- Spherical microphone using Pentakis-Dodecahedron grid working up to 4^{th} order as shown in [3].

In Fig. 3, the obtained modules are sketched under the names "Encoding" and "Decoding".

4.3. Rigid spherical microphone filters

As mentioned in Sec. 3.1.2, the $B_{mn}^{\sigma}(z)$ components derived from a rigid spherical microphone array signals should be filtered by $E_m(z)$ filters to take into account the diffraction of the rigid sphere supporting the capsules. Theses filters present a theoretical infinite gain at 0 Hz and very large "bass-boost" for high orders [24]. They are stabilized by high-pass filters [25] or Tikhonov filters [3], which cut the low frequencies at high orders. Resulting filters are implemented as FIR filters. However, in [26, 27, 24] IIR implementations are proposed with lower orders amplification according to higher orders limitation. These approaches should be investigated in future works for a FAUST implementation since an IIR parametric filters could be interesting to monitor in real-time the performances of a spherical microphone array. For now, in the reported implementation, the filters are implemented as FIR filters. The cut-off frequencies of high-pass filters are chosen with a maximum amplification level. The real-time convolution is made with BRUTEFIR[3] for Linux environment. This module is sketched in Fig. 3 under the name "Equalization".

4.4. Encoding of virtual source and panning law

In the FAUST implementation reported in this paper, the encoding of a virtual source is based on Eq. (3) for a plane wave and Eq. (13) for a spherical wave. In the current state of the implementation, the spherical harmonics are explicitly coded in a `ymn.lib` library up to order five (36 functions). However, they could be computed by recurrence if higher orders would be required. The monophonic

[3] `http://www.ludd.luth.se/~torger/brutefir.html`

signal a_1 is multiplied by corresponding spherical harmonics evaluated at desired direction (θ_1, δ_1) and filtered by near-field filters with desired radii (r_1, r_0) as shown in Fig. 3: Ambisonics signals are then obtained.

The equivalent panning laws of Eq. (8) circumvent the need to encode and decode with matrix operations. Indeed, the computation is reduced to a sum on the orders thanks to the additivity theorem of spherical harmonics. This is of great interest for real-time synthesis. For the moment, the panning-laws are implemented for several Lebedev spheres using $N = 6, 26$, or 50 loudspeakers, as presented in [16]. The Legendre polynomials P_m are explicitely coded in `ymn.lib` up to order five. The weights w_l and angles θ_l, δ_l of the spheres are implemented in a `lebedev.lib` file. This spatialization module is sketched in Fig. 3 under the name "Panning-law": the loudspeakers input signals are computed from a monophonic signal a_2 and spatial parameters $(r_2, \theta_2, \delta_2, r_0)$.

5. VISUAL FEEDBACK WITH PROCESSING

FAUST provides a graphical user interface with VU-Meters. Unfortunately, those meters are organized as classical lines of VU-Meters (see Fig. 7 as an example). Thus, they do not provide an efficient visual cue of where the signal energy is distributed on the loudspeaker spherical layout. Moreover, a 3D representation of the virtual source helps to provide an objective spatial view of the source position. Such type of 3D representation is also much more useful to composers or engineers who are not familiar with under-the-hood DSP. In this context, a visual tool for 3D visual feedback was implemented using PROCESSING[4] language. The code uses an `Atom` class to create a ball representing a loudspeaker. The loudspeakers coordinates are given in a `.cvs` file and are easily adaptable to any configuration. The virtual sources are represented as balls with fading trajectories. The RMS gain (in dBFS) of each loudspeaker given by the Ambisonics solution drives in real-time the size and the color of the loudspeaker's-balls via OSC messages. The position of each source is also received by OSC messages. The `Peasycam` library[5] allows to zoom, move and pan the 3D view. The described visual tool is shown in Fig. 6. The sphere used here is a Lebedev sphere using $N_2 = 50$ loudspeakers as used in [16].

6. USER CONTROL INTERFACE

6.1. Controls by software

The user control interface is provided by FAUST compiled code. Thus, depending on the application it could be a standalone application, a MAX/MSP patch, VST or LV2 plugin, or others. It consists in sliders and entry boxes, to control the parameters $(r_1, \theta_1, \delta_1, r_0)$. Check-boxes are used to mute an order m. VU-Meters give the signals level in dBFS for the $B^{\sigma}_{mn}(z)$ components or s_l.

As an example, a JACK[6] standalone application interface for real-time 3D spatialization using the panning law of Eq. (13) is shown in Fig. 7.

[4] `https://processing.org/`
[5] `http://mrfeinberg.com/peasycam/`
[6] `http://jackaudio.org`

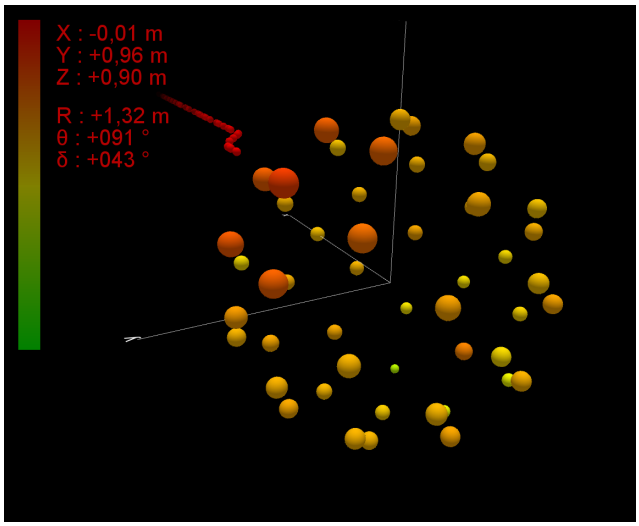

Figure 6: 3D visual feedback tool using PROCESSING language. The virtual source is shown as a red ball with fading trajectory (top left corner). Each loudspeaker is represented by a ball with radius and color driven by loudspeaker signal level (RMS, dBFS). 0 dBFS correspond to the red color on the left color scale and $-\infty$ dBFS to green. All informations are received by OSC messages generated by a joystick and a PURE DATA patch.

6.2. Controls by hardware device and OSC messages

FAUST supports OSC to control the parameters of the DSP tools. For example, with a hardware controller and a PURE DATA patch, it is possible to generate OSC messages which control the plug-in as well as the visual feedback tool of Sec. 5. This correspond to the rightmost branch of Fig. 3. The use of hardware controllers and FAUST generated plug-ins allow for the easy control and recording of the synthesis parameters. As an example, the implemented Ambisonics encoder loaded in ARDOUR[7] Digital Audio Workstation (DAW) as a LV2 plug-in is shown in Fig. 8. With the visual feedback tool described in Sec. 5, this configuration enables to record automation tracks describing "manually-created" trajectories.

7. EXPERIMENTAL SETUP

An experimental setup is presented briefly in this section as an application of the tools presented above.

7.1. 3D sound pressure field capture

The recording of natural sound pressure fields is made with a rigid spherical microphone "MemsBedev"[8] shown in Fig. 9. This microphone was made by 3D printing and uses $N = 50$ microphone capsules on a Lebedev grid [28]. Each capsule is made of 4 MEMS[9] microphones to reduce the background noise. The microphone works up to 5^{th} order, as explained in Ref.[16]. The analog signals are digitalized with a commercial front-end. The tools presented in this article can provide in real-time the Ambisonics components $B^{\sigma}_{mn}(z)$ up to 5^{th} order according to Eq. (5) and Fig. 3.

[7] `http://ardour.org/`
[8] `http://www.cinela.fr/`
[9] MEMS: Micro Electro-Mechanical System

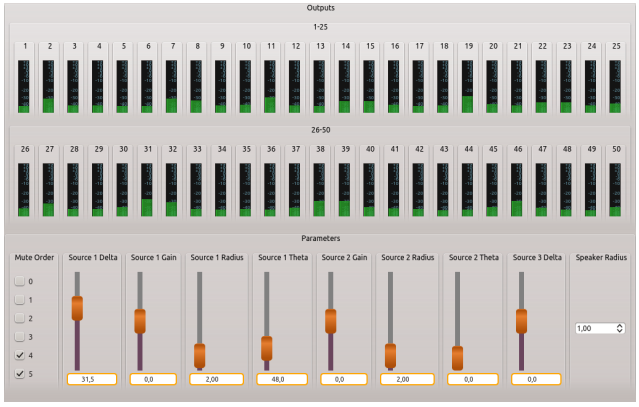

Figure 7: Standalone application interface running under JACK. The sliders allow controlling the gains and positions of two virtual sources (bottom, grey and yellow). The check-boxes (bottom-left) mute and unmute a specific order at the synthesis stage. The entry box allows giving the r_0 radius of the rendering spherical loudspeakers layout (bottom right). The VU-meters give the signals levels in dBFS for each loudspeaker. In this case there are $N_2 = 50$ loudspeakers according to a Lebedev grid.

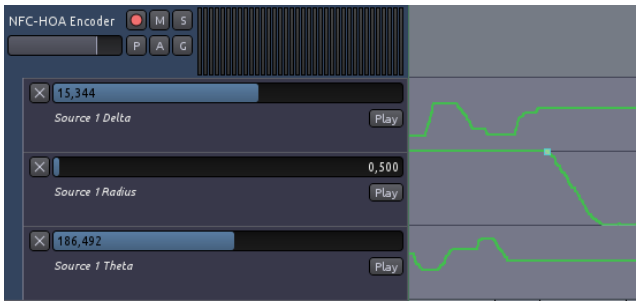

Figure 8: Ambisonics encoder loaded in ARDOUR as a LV2 plug-in. The positions parameters are controlled with the mouse or by an hardware joystick via OSC messages. Resulting automation tracks are shown in green in this figure. The LV2 plug-in is generated from FAUST code using `faust2lv2`.

7.2. 3D sound pressure field synthesis

The decoding of the pressure sound field is made in real-time with a decoder made with FAUST according to Eq. (7) and Fig. 3. The Lebedev spherical loudspeaker layout used for the sound field rendering is shown in Fig. 10. For recording rendering purposes, it is possible to record in 3D in one place and render in real-time the 3D sound pressure field with the loudspeaker sphere located at another location. For spatialization purposes, the user takes place in the sphere and drives the position of the virtual sources using an hardware controller. The hardware controller is linked to a PURE DATA patch and generates OSC messages for the synthesis using FAUST, as summarized in Fig. 3. This experimental setup can thus help to mix 3D audio contents *in situ*.

Figure 9: MemsBedev microphone arrays with $N = 50$ MEMS microphones

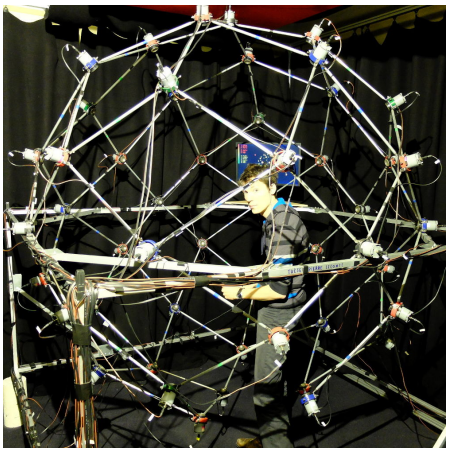

Figure 10: Lebedev spherical loudspeakers layout with $N_2 = 50$ loudspeakers.

8. CONCLUSION

Several integrated tools for real-time 3D sound pressure field synthesis using Ambisonics were developed and presented in this paper. The signal processing was implemented using FAUST, the visual feedback with PROCESSING and the communication between tools with OSC protocol. The near-field filters were implemented in a FAUST library and allow synthesizing spherical waves. The gain correction with distance is also implemented, which is of great importance for engineering applications or simulators. In the current version of the implementation, the synthesis is controlled up to 5^{th} order in 3D. Some future ameliorations could include transformations in Ambisonics domain [29], recent advances in radial filters implementation [27] or inclusion of the Doppler effect for moving sources [30]. The code of this project is freely available online under GPL license[10]. In a near future, the described implementation and experimental setup will be used to reproduced various recorded environments and achieve physical evaluation of the reproduced sound fields.

[10] `https://github.com/sekisushai/ambitools`

9. REFERENCES

[1] Jérôme Daniel, *Représentation de champs acoustiques, application à la transmission et à la reproduction de scènes sonores complexes dans un contexte multimédia*, Ph.D. thesis, Université Paris 6, Paris, 2000.

[2] Jérôme 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, 2003, pp. 1–15, AES.

[3] Sébastien Moreau, Jérôme Daniel, and Stéphanie Bertet, "3d sound field recording with higher order ambisonics-objective measurements and validation of spherical microphone," in *Audio Engineering Society Convention 120*, Paris, 2006, pp. 1–24, AES.

[4] Matthias Kronlachner, "Plug-in Suite for mastering the production and playback in surround sound and ambisonics," in *Linux Audio Conference*, 2013.

[5] Christian Nachbar, Franz Zotter, Etienne Deleflie, and Alois Sontacchi, "Ambix - A suggested ambisonics format," in *Ambisonics Symposium, Lexington*, 2011.

[6] Aaron J. Heller and Eric M. Benjamin, "The Ambisonic Decoder Toolbox: Extensions for Partial-Coverage Loudspeaker Arrays," in *Linux Audio Conference*, 2014.

[7] Julien Colafrancesco, Pierre Guillot, Eliott Paris, Anne Sèdes, and Alain Bonardi, "La bibliothèque HOA, bilan et perspectives," in *Journées d'Informatique Musicale*, 2013, pp. 189–197.

[8] Matthias Geier and Sascha Spors, "Spatial Audio with the SoundScape Renderer," in *27th Tonmeistertagung–VDT International Convention*, 2012.

[9] Darren B. Ward and Thushara D. Abhayapala, "Reproduction of a plane-wave sound field using an array of loudspeakers," *IEEE Transactions on Speech and Audio Processing*, vol. 9, no. 6, pp. 697–707, 2001.

[10] Philippe-Aubert Gauthier, Cédric Camier, Thomas Padois, Yann Pasco, and Alain Berry, "Sound Field Reproduction of Real Flight Recordings in Aircraft Cabin Mock-Up," *J. Audio Eng. Soc*, vol. 63, no. 1/2, pp. 6–20, 2015.

[11] Anthony Bolduc, Philippe-Aubert Gauthier, Telina Ramanana, and Alain Berry, "Sound Field Reproduction of Vibroacoustic Models: Application to a Plate with Wave Field Synthesis," in *Audio Engineering Society Conference: 55th International Conference: Spatial Audio*, 2014.

[12] Michael Vorländer, *Auralization: fundamentals of acoustics, modelling, simulation, algorithms and acoustic virtual reality*, Springer Science & Business Media, 2007.

[13] Estelle Bongini, *Modèle acoustique global et synthèse sonore du bruit d'un véhicule: application aux véhicules ferroviaires*, Ph.D. thesis, Université de Provence-Aix-Marseille I, 2008.

[14] Yann Orlarey, Dominique Fober, and Stéphane Letz, "FAUST: an efficient functional approach to DSP programming," *New Computational Paradigms for Computer Music*, vol. 290, 2009.

[15] Boaz Rafaely, "Analysis and design of spherical microphone arrays," *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 1, pp. 135–143, Jan. 2005.

[16] Pierre Lecomte, Philippe-Aubert Gauthier, Christophe Langrenne, Alexandre Garcia, and Alain Berry, "On the use of a Lebedev grid for Ambisonics," in *Audio Engineering Society Convention 139*, New York, 2015.

[17] Boaz Rafaely, Barak Weiss, and Eitan Bachmat, "Spatial aliasing in spherical microphone arrays," *IEEE Transactions on Signal Processing*, vol. 55, no. 3, pp. 1003–1010, 2007.

[18] Mark A. Poletti, "Three-dimensional surround sound systems based on spherical harmonics," *Journal of the Audio Engineering Society*, vol. 53, no. 11, pp. 1004–1025, 2005.

[19] George B. Arfken and Hans J. Weber, *Mathematical methods for physicists - sixth edition*, Elsevier, 6th edition, 2005.

[20] Fons Adriaensen, "Near field filters for higher order Ambisonics," http://kokkinizita.linuxaudio.org/papers/hoafilt.pdf, 2006, Accessed: 2013-10-28.

[21] Sascha Spors, Rudolf Rabenstein, and Jens Ahrens, "The theory of wave field synthesis revisited," in *Audio Engineering Society Convention 124*, Amsterdam, 2008, pp. 1–19, AES.

[22] Alan V. Oppenheim and Ronald W. Schafer, *Digital signal processing*, Prentice-Hall, Englewood Cliffs, NJ, 1975.

[23] Jens Ahrens and Sascha Spors, "Focusing of virtual sound sources in higher order Ambisonics," in *Audio Engineering Society Convention 124*, Amsterdam, 2008, pp. 1–9, AES.

[24] Stefan Lösler and Franz Zotter, "Comprehensive radial filter design for practical higher-order Ambisonic recording," *Fortschritte der Akustik, DAGA*, 2015.

[25] Jérôme Daniel and Sébastien Moreau, "Further study of sound field coding with higher order ambisonics," in *Audio Engineering Society Convention 116*, Berlin, 2004, pp. 1–14, AES.

[26] Hannes Pomberger, "Angular and radial directivity control for spherical loudspeaker arrays," 2008.

[27] Robert Baumgartner, Hannes Pomberger, and Matthias Frank, "Practical implementation of radial filters for ambisonic recordings," *Proc. of ICSA, Detmold*, 2011.

[28] V.I. Lebedev, "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*, vol. 15, no. 1, pp. 44–51, 1975.

[29] Matthias Kronlachner and Franz Zotter, "Spatial transformations for the enhancement of Ambisonic recordings," in *Proceedings of the 2nd International Conference on Spatial Audio, Erlangen*, 2014.

[30] Gergely Firtha and Peter Fiala, "Sound Field Synthesis of Uniformly Moving Virtual Monopoles," *Journal of the Audio Engineering Society*, vol. 63, no. 1/2, pp. 46–53, 2015.