HAL
open science

JSAmbisonics: A Web Audio library for interactive spatial sound processing on the web

Archontis Politis, David Poirier-Quinot

To cite this version:

Archontis Politis, David Poirier-Quinot. JSAmbisonics: A Web Audio library for interactive spatial sound processing on the web. Interactive Audio Systems Symposium, Sep 2016, York, United Kingdom. hal-01790246

HAL Id: hal-01790246

https://hal.science/hal-01790246v1

Submitted on 11 May 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

JSAmbisonics: A Web Audio library for interactive spatial sound processing on the web

ARCHONTIS POLITIS

Dept. of Signal Processing and Acoustics, Aalto University, Espoo, Finland
e-mail: archontis.politis@aalto.fi

DAVID POIRIER-QUINOT

IRCAM, Paris, France
e-mail: david.poirier-quinot@ircam.fr

September 23rd 2016.

Abstract

This paper introduces the JSAmbisonics library, a set of JavaScript modules based on the Web Audio API for spatial sound processing. Deployed via Node.js, the library consists of a compact set of tools for reproduction and manipulation of first- or higher-order recorded or simulated Ambisonic sound fields. After a brief introduction to the fundamentals of Ambisonic processing, the main components (encoding, rotation, beamforming, and binaural decoding) of the JSAmbisonics library are detailed. Each component, or "node", can be used on its own or combined with others to support various application scenarios, discussed in Section 4. An additional library developed to support spherical harmonic transform operations is introduced in Section 3.2. Careful consideration has been given to the overall computational efficiency of the JSAmbisonics library, particularly regarding spatial-encoding and decoding schemes, optimized for real-time production and delivery of immersive web contents.

1 Introduction

The emerging technological trends on delivery of audiovisual content are currently targeting increased immersion. After the increase in bandwidth and computational power that made delivery of high quality audio and video content on devices such as smartphones possible, making this content immersive is considered a requirement in order to provide a leap in user experience compared to the traditional modes of enjoying audiovisual content. Virtual and augmented reality technology has also resurfaced, targeting mobile platforms and seemingly closer to large scale deployment. Spatial sound is a fundamental component of these immersive technologies.

Effective spatial sound tools for creation of immersive content are well known from an audio engineering point of view; panning tools for loudspeakers, binaural filters for headphones, reverberation and decorrelation for a sense of space. One approach to spatial scene description and generation is to define all individual sound sources and environment along with their spatialization parameters, an approach termed object-based spatial audio. An alternative is to consider a scene-based description, in which the audio signals describe a full sound scene. Such a representation has certain advantages over the object-based approach, as long as the format is adequate to reproduce the sound scene with high perceptual quality and there is no intention of re-mixing the scene components at the client side. Such advantages are lower transmission requirements, compared to the high number of object channels, efficient implementation of scene effects, such as rotations, and direct mixing with recorded sound scenes.

Ambisonics [1, 2, 3, 4] is such a method with the main advantage that it offers a canonical and hierarchical representation of the spatial sound scene,

and it is computationally efficient. Ambisonics treats synthetic and captured sound scenes in a common framework, which makes them especially suitable for spherical audio recording in conjunction with spherical video. Furthermore, it provides a suitable method for rendering to headphones with a combination of ambisonic theory and binaural filters, and suitable tools for rotations and manipulations of the scene.

This paper presents an Ambisonics audio library that utilizes the Web Audio API (WAA) [5] for interactive spatial sound processing on the web [6]. That makes the library useful for spatial sound creation on any modern browser that supports WAA. The library is written in JavaScript (JS) and is easy to use and incorporate in a web application. Special effort has been given in making the library comprehensive and extensible. The library supports Higher-order Ambisonics (HOA) of arbitrary order and implements most fundamental ambisonic processing blocks for generating and reproducing a sound scene. These operations, their implementation and potential applications are presented below.

2 Ambisonics background

2.1 Sound scene description in Ambisonics

Assuming that all sound sources are on the far-field, a general sound scene can be described as a continuous distribution of plane waves with spatio-temporal amplitude $a(t, \boldsymbol{\gamma})$ for a plane wave incident from direction $\boldsymbol{\gamma} = [\cos\phi\cos\theta, \sin\phi\cos\theta, \sin\theta]^{\mathrm{T}}$, with (ϕ, θ) being the azimuth and elevation angle respectively. By taking the spherical harmonic transform (SHT) of the amplitude density, we arrive at the ambisonic description of the sound scene, encoded into the SH coefficients of the amplitude density $\mathbf{a}$, or equivalently, the ambisonic signals

$$\mathbf{a}(t) = \mathcal{SHT}\{a(t, \boldsymbol{\gamma})\} = \int_{\boldsymbol{\gamma}} a(\boldsymbol{\gamma})\mathbf{y}(\boldsymbol{\gamma})\,\mathrm{d}\boldsymbol{\gamma}, \quad (1)$$

where $\int_{\boldsymbol{\gamma}} \mathrm{d}\boldsymbol{\gamma}$ denotes integration over the surface of the unit sphere, and $\mathrm{d}\boldsymbol{\gamma} = \cos\theta \mathrm{d}\theta\mathrm{d}\phi$ is the differential surface element. The basis vector $\mathbf{y}(\boldsymbol{\gamma})$ contains all SHs up to a specified maximum order N. For a SHT of order N, there are $M = (N+1)^2$ SHs and ambisonic signals. Following established HOA conventions, real SHs are used and defined as

$$Y_{nm}(\theta, \phi) = \sqrt{(2n+1)\frac{(n-|m|)!}{(n+|m|)!}} P_{n|m|}(\sin\theta) y_m(\phi), \quad (2)$$

with

$$y_m(\phi) = \begin{cases} \sqrt{2}\sin|m|\phi & m < 0, \\ 1 & m = 0, \\ \sqrt{2}\cos m\phi & m > 0, \end{cases} \quad (3)$$

and P_{nm} the associated Legendre functions of degree n. The SHs are orthonormal with

$$\int_{\boldsymbol{\gamma}} \mathbf{y}(\boldsymbol{\gamma})\mathbf{y}^{\mathrm{T}}(\boldsymbol{\gamma})\,\mathrm{d}\boldsymbol{\gamma} = 4\pi\mathbf{I} \quad (4)$$

where $\mathbf{I}$ is the $M \times M$ identity matrix. Using this power normalization, the 0th order ambisonic signal a_{00} is equivalent to an omnidirectional signal at the origin.

The most commonly used ordering of SHs in most scientific fields and, consequently, the ambisonic signals, is

$$[\mathbf{y}(\boldsymbol{\gamma})]_q = Y_{nm}(\boldsymbol{\gamma}), \quad \text{with } q = 1, 2, ..., (N+1)^2 \text{ and } q = n^2 + n + m + 1. \quad (5)$$

From the index q the mode number (n, m) can be recovered as $n = \lfloor\sqrt{q-1}\rfloor$ and $m = q - n^2 - n - 1$. In HOA literature, the ordering of Eq. 5 is known as *ACN* ambisonic channel ordering, and the normalization of Eq. 2&4 as *N3D* normalization.

2.2 Ambisonic encoding

Encoding of a plane wave source carrying a signal $s(t)$, incident from $\boldsymbol{\gamma}_0$, to ambisonic signals is given by

$$\mathbf{a}(t) = s(t)\mathbf{y}(\boldsymbol{\gamma}_0), \quad (6)$$

so that multiple signals for K sources can be encoded as

$$\mathbf{a}(t) = \sum_{k=1}^{K} s_k(t)\mathbf{y}(\boldsymbol{\gamma}_k). \quad (7)$$

2.3 Ambisonic rotation

Rotation of the sound scene can be conveniently performed in the SHD by applying a SH rotation matrix to the ambisonic signals. More specifically, for a rotation of the coordinate system given by the three Euler angles α, β, γ, the signals of the rotated scene are given by

$$\mathbf{a}_n^{\mathrm{rot}}(t) = \mathbf{M}_n^{\mathrm{rot}}(\alpha, \beta, \gamma)\mathbf{a}_n(t), \text{ with } n = 1, 2, ..., N \quad (8)$$

where $\mathbf{a}_n = [a_{n(-n)}, ..., a_{nn}]^{\mathrm{T}}$ denotes the ambisonic signals of order n, and $\mathbf{M}_n^{\mathrm{rot}}$ is an $(n+1)^2 \times (n+1)^2$ rotation matrix for the certain order. Semi-closed form solutions for the rotation matrices exist only for

complex SHs, and they are too computationally demanding to compute for real-time applications. However, fast recursive algorithms exist for rotation of real SHs, that are efficient and suitable for ambisonic processing. [7, 8].

2.4 Ambisonic reflection

Reflection, or mirroring, of the sound scene along the principal planes of yz (front-back), xz (left-right), or xy (up-down) becomes a trivial operation in the SHD due to symmetry properties of the SHs. As SHs are either symmetric or antisymmetric with respect to these planes, the ambisonic signals either remain the same under reflection (for symmetric SHs) or are inverted (for antisymmetric SHs). Hence, reflection reduces to inverting the polarity of specific sets of ambisonic signals, depending on the reflection plane:

$$(m < 0 \wedge m \text{ even}) \cup (m \geq 0 \wedge m \text{ odd}) : \ \mathbf{yz} \tag{9}$$

$$m < 0 : \ \mathbf{xz} \tag{10}$$

$$(n + m) \text{ odd} : \ \mathbf{xy}. \tag{11}$$

2.5 Ambisonic beamforming

Beamforming in the SHD reduces to a weight-and-sum operation of the SH signals. In ambisonic literature SH beamforming has been traditionally termed a *virtual microphone*. In case the directional pattern of the virtual microphone is axisymmetric, which is usually the case of interest, the virtual microphone signal $x_{\text{vm}}(t)$ is given by

$$x_{\text{vm}}(t, \boldsymbol{\gamma}_0) = \mathbf{w}^{\text{T}}(\boldsymbol{\gamma}_0)\mathbf{a}(t) \tag{12}$$

where $\boldsymbol{\gamma}_0$ is the orientation of the virtual microphone, and $\mathbf{w}(\boldsymbol{\gamma}_0)$ the $(N+1)^2$ vector of beamforming weights. The weight vector follows the ordering of the SHs, and can be expressed as a pattern-dependent part and a rotation-dependent part as

$$[\mathbf{w}(\boldsymbol{\gamma}_0)]_q = w_{nm} = c_n Y_{nm}(\boldsymbol{\gamma}_0). \tag{13}$$

The $(N + 1)$ coefficients c_n are derived according to desired properties of the virtual microphone; some patterns of interest are presented below.

2.6 Ambisonic decoding

2.6.1 Loudspeaker decoding

The ambisonic signals can be distributed to a playback setup through a decoding mixing matrix, a process termed ambisonic decoding. Commonly, this decoding matrix is frequency-independent, especially in HOA. Its design can be performed according to physical or psychoacoustical criteria. The signals $\mathbf{x}_{\text{ls}} = [x_1, ..., x_L]$ for L loudspeakers are then obtained by

$$\mathbf{x}_{\text{ls}}(t) = \mathbf{D}_{\text{ls}}\mathbf{a}(t) \tag{14}$$

where $\mathbf{D}_{\text{ls}}$ is the $L \times (N + 1)^2$ decoding matrix.

Some straightforward designs for the decoding matrix are the following:

$$\text{Sampling} : \mathbf{D}_{\text{ls}} = \frac{1}{L}\mathbf{Y}_L^{\text{T}} \tag{15}$$

$$\text{Mode} - \text{matching} : \mathbf{D}_{\text{ls}} = (\mathbf{Y}_L^{\text{T}}\mathbf{Y}_L + \beta^2\mathbf{I})^{-1}\mathbf{Y}_L^{\text{T}} \tag{16}$$

$$\text{ALLRAD} : \mathbf{D}_{\text{ls}} = \frac{1}{N_{\text{td}}}\mathbf{G}_{\text{td}}\mathbf{Y}_{\text{td}}^{\text{T}} \tag{17}$$

where $\mathbf{Y}_L = [\mathbf{y}(\boldsymbol{\gamma}_1), ..., \mathbf{y}(\boldsymbol{\gamma}_L)]$ is the $(N + 1)^2 \times L$ matrix of SHs at the loudspeaker directions. In the mode-matching approach, the least-squares solution is usually constrained with a regularization value β. In the ALLRAD method [4], $\mathbf{Y}_{\text{td}} = [\mathbf{y}(\boldsymbol{\gamma}_1), ..., \mathbf{y}(\boldsymbol{\gamma}_T)]$ is the matrix of SHs at the N_{td} directions of a uniform spherical t-design [9], of $t \geq 2N + 1$, while the $\mathbf{G}_{\text{td}}$ is an $L \times N_{\text{td}}$ matrix of vector-amplitude panning gains (VBAP) [10], with the t-design directions considered as virtual sources.

2.6.2 Binaural decoding

Ambisonics are suitable for headphone reproduction, by integrating head-related transfer functions (HRTFs). As HRTFs are frequency-dependent, so are the decoding matrices in this case. More specifically, the binaural signals $\mathbf{x}_{\text{bin}} = [x_{\text{L}}, x_{\text{R}}]^{\text{T}}$ are given by

$$\mathbf{x}_{\text{bin}}(f) = \mathbf{D}_{\text{bin}}(f)\mathbf{a}(f) \tag{18}$$

with $\mathbf{D}_{\text{bin}}$ being the $2 \times (N + 1)^2$ decoding matrix. In the time domain, Eq. 18 translates to a sum of convolutions as

$$\mathbf{x}_{\text{bin}}(t) = \begin{bmatrix} \sum_{q=1}^{(N+1)^2} d_q^{\text{L}}(t) * a_q(t) \\ \sum_{q=1}^{(N+1)^2} d_q^{\text{R}}(t) * a_q(t) \end{bmatrix} \tag{19}$$

where $(*)$ denotes convolution and $d_q^{\text{L}}(t) = \mathcal{IFT}\{[\mathbf{D}_{\text{bin}}]_{1,q}(f)\}$ is the filter derived from the inverse Fourier transform of the q-th entry of the decoding matrix for the left ear, and similarly for the right. Hence, in the general case $2 \times (N + 1)^2$ convolutions are required for binaural decoding.

There are two ways to derive the decoding matrix coefficients, or equivalently the filters. The direct approach takes advantage of the Parseval's theorem for

the SHT, which for a sound distribution $a(f, \boldsymbol{\gamma})$ and e.g. the left HRTF $h_{\mathrm{L}}(f, \boldsymbol{\gamma})$ states that

$$\begin{aligned} x_{\mathrm{L}}(f) &= \int_{\boldsymbol{\gamma}} a(f, \boldsymbol{\gamma}) h_{\mathrm{L}}(f, \boldsymbol{\gamma}) \, \mathrm{d}\boldsymbol{\gamma} \\ &= \mathcal{SHT}\left\{a(f, \boldsymbol{\gamma})\right\} \cdot \mathcal{SHT}\left\{h_{\mathrm{L}}(f, \boldsymbol{\gamma})\right\} \\ &= \mathbf{h}_{\mathrm{L}}^{\mathrm{T}}(f) \, \mathbf{a}(f). \end{aligned} \tag{20}$$

where $\mathbf{h}_{\mathrm{L}}$ are the coefficients of the SHT applied on the HRTF. The above Eq. 20 states that the binaural signals are the result of the inner product between the ambisonic coefficients and the SH coefficients of the HRTFs. Hence the decoding matrix in this case is $\mathbf{D}_{\mathrm{bin}}(f) = [\mathbf{h}_{\mathrm{L}}(f), \mathbf{h}_{\mathrm{R}}(f)]^{\mathrm{T}}$. Expansion of HRTFs into SH coefficients has been researched extensively, mainly in the context of HRTF interpolation [11, 12, 13].

The second way, and the one seen more often in literature [14, 15], is the virtual loudspeaker approach, in which plane wave signals are decoded with a decoding matrix of preference $\mathbf{D}_{\mathrm{vls}}$, covering the sphere adequately, and then consequently convolved with the HRTFs for the decoding directions. The number K of decoding directions is selected to be high enough for the order of the available ambisonic signals, with $K > (N+1)^2$. Formulated in the frequency domain, the virtual loudspeaker approach becomes

$$\mathbf{x}_{\mathrm{bin}}(f) = \mathbf{H}_{\mathrm{LR}}(f) \mathbf{D}_{\mathrm{vls}} \mathbf{a}(f) = \mathbf{D}_{\mathrm{bin}}(f) \mathbf{a}(f), \tag{21}$$

where

$$\mathbf{H}_{\mathrm{LR}} = \begin{bmatrix} h_{\mathrm{L}}(f, \boldsymbol{\gamma}_1) & h_{\mathrm{R}}(f, \boldsymbol{\gamma}_1) \\ \dots & \dots \\ h_{\mathrm{L}}(f, \boldsymbol{\gamma}_K) & h_{\mathrm{R}}(f, \boldsymbol{\gamma}_K) \end{bmatrix}^{\mathrm{T}} \tag{22}$$

is the matrix of HRTFs for the decoding directions. Note that the final ambisonic decoding matrix $\mathbf{D}_{\mathrm{bin}} = \mathbf{H}_{\mathrm{LR}} \mathbf{D}_{\mathrm{vls}}$ is again of size $2 \times (N+1)^2$, no matter the number of decoding directions K.

If it is assumed that the left and right HRTFs are antisymmetric with respect to the median plane (termed here as xz-antisymmetry), e.g. when non-personalized HRTFs are applied, then what the right ear would capture is similar to the left ear signal if the sound scene distribution was mirrored with respect to the median plane. Such mirroring corresponds to Eq. 10. In practice, that means that only the $(N+1)^2$ left-ear HRTF filters need to be applied to derive both ear signals. Any of the two methods presented above can be used for computing the filters. Assuming two intermediate signals $M(t)$ and $S(t)$ with

$$\begin{aligned} M(t) &= \sum_{q|m \geq 0} d_q^{\mathrm{L}}(t) * a_q(t) \\ S(t) &= \sum_{q|m < 0} d_q^{\mathrm{L}}(t) * a_q(t) \end{aligned} \tag{23}$$

the binaural signals can be derived simply by

$$\mathbf{x}_{\mathrm{bin}}(t) = \begin{bmatrix} M(t) + S(t) \\ M(t) - S(t) \end{bmatrix}. \tag{24}$$

This formulation is of practical importance for real-time applications since it reduces the required number of convolutions by half. This fact has been noted in literature with the virtual loudspeaker approach, assuming antisymmetric arrangements [15]. It can also be seen, however, from a purely ambisonic perspective as shown above.

3 Implementation

3.1 Web Audio API

WAA contains all signal processing elements that permit the realization of ambisonic processing. More specifically, since they are all either frequency independent or frequency-dependent linear processes, they can be realized with gain factors, convolutions and summations on the ambisonic signals. In WAA fundamental signal processing blocks are called *Audio Nodes*. Three such audio nodes are used in the implementation of all ambisonic processing blocks. The first is the *Gain Node*, a simple signal multiplier with user-controlled gain at runtime. The second is a convolution block, the *Convolver Node*, which performs linear convolution with user-specified FIR filters. This block is utilized for the convolutions in the binaural decoding stage. Finally, the $(N+1)^2$ channels for a specified order are grouped into single streams when sent from an ambisonic block to another, by using the *Channel Merger Node*, and split again into the constituent channels using *Channel Splitter Node* when received from an ambisonic block, to be processed.

Vector and matrix operations on the ambisonic signals are realized with groups of gain nodes and by summing appropriately the resulting channels. An alternative to this can be the *Audio Worker Node*, in which JS code is applied directly on the audio buffers. However, the built-in gain nodes handle the fast updating of values during runtime without artifacts, and the benefit of an audio worker implementation is expected to be small if any. An implementation and comparison of such an approach is planned as future work.

3.2 JS Spherical Harmonic Transform library

Since there is no existing JS library for the spherical harmonic operations involved in ambisonic processing, a custom made one was created for this project [16]. The library performs the following basic operations:

- Computation of all associated Legendre functions up to a maximum degree N, for sets of points, using fast recursive formulas [17].
- Computation of all real SHs up to a specified order N, for sets of directions.
- Computation of the forward SHT, using either a direct weighted-sum approach of the data points, or by a least-squares approach. The transform returns a vector of SH coefficients.
- Computation of the inverse SHT at an arbitrary direction, using the SH coefficients from the forward transform.
- Computation of rotation matrices in the SHD, using the fast recursive solution of [7] for real SHs .

Applications of JSHT are not limited only to Web audio and ambisonics. Graphics and scientific applications that benefit from a spherical spectral representation can use it for demonstrative purposes deployed on the Web. Spherical interpolation of directional data is such an example.

3.3 JS Ambisonics library

The WAA Ambisonics library implements a set of audio processing blocks that realize most of the fundamental operations presented in Sec. 2. SH computations are performed internally using the JS SHT library described above. All ambisonic processing follows the ACN/N3D convention. However, a number of blocks are provided for converting other channel and normalization conventions to this specification. All ambisonic blocks expose an *in* and *out* node, that can be used for WAA style of connecting audio blocks. Furthermore, they expose some properties and methods that can be updated during real-time, for interactive operation. For a detailed documentation of the object properties the reader is referred to [6].

3.3.1 Encoding, Rotation & Mirroring

The *monoEncoder* object takes a monophonic sound stream and encodes it in an ambisonic stream of a user-specified order, and at a user specified direction, using Eq. 6. The source direction can be updated interactively at runtime.

The *sceneRotator* object takes an ambisonic stream of a certain order and returns the stream of the same order for a rotated sound scene. The scene rotation is given in *yaw-pitch-roll* convention. To avoid redundant computations, the ambisonic signals of each order n are multiplied only with the rotation matrix $\mathbf{M}_n^{\text{rot}}$ of that order, as shown in Eq. 8. The *sceneMirror* object implements mirroring through the polarity inversions of Eq. 9–11. Both rotation and mirroring can be updated interactively.

3.3.2 Virtual Microphones

The *virtualMic* object implements an ambisonic beam former of a user-specified type and orientation. The block implements Eq. 12, with the following options controlling the type of a virtual microphone of order N through the coefficients c_n of Eq. 13:

$$\text{cardioid}: \quad c_n = \frac{N!N!}{(N+n+1)!(N-n)!} \tag{25}$$

$$\text{hypercardioid}: \quad c_n = \frac{1}{(N+1)^2} \tag{26}$$

$$\text{max}-\text{rE}: \quad c_n = \frac{P_n(\cos\kappa_N)}{\sum_{n=0}^{N}(2n+1)P_n(\cos\kappa_N)} \tag{27}$$

with $\kappa_N = \cos(2.407/(N+1.51))$ as given in [4]. Higher-order cardioids are defined as a normal cardioid raised to the power of N. Higher-order hypercardioids maximize the directivity factor for a given order; in spherical beamforming literature also known as *regular* or *plane-wave decomposition* beamformers. The max-rE pattern originates from ambisonic literature and maximizes the acoustic intensity vector in an isotropic diffuse field. Apart from the above, higher-order supercardioids are also implemented up to 4th order with the coefficients converted appropriately from [18]. Supercardioids maximize the front-to-back power ratio for a given order.

3.3.3 Conversion between formats

All operations are internally performed using the ACN/N3D specification. However, the vast majority of recorded ambisonic material is first order, and it follows the traditional B-format specification of *WXYZ* channel ordering. Conversion from this specification to ACN/N3D can be expressed by the con-

version matrix

$$\mathbf{x}_{\mathrm{ACN/N3D}} = \begin{bmatrix} \sqrt{2} & 0 & 0 & 0 \\ 0 & 0 & \sqrt{3} & 0 \\ 0 & 0 & 0 & \sqrt{3} \\ 0 & \sqrt{3} & 0 & 0 \end{bmatrix} \mathbf{x}_{\mathrm{WXYZ}} \quad (28)$$

Regarding HOA, the first existing specification is the *Furse-Malham* (FuMa) one [19], defined up to third order. Conversion from WXYZ or FuMa to ACN/N3D can be performed with the *converters.bf2acn* and *converters.fuma2acn* objects respectively. Note that the first-order specification of FuMa is the same as the traditional WXYZ one.

Recent HOA research and technology uses the ACN ordering scheme as the standard. However, in terms of SH normalization there are two popular schemes, the orthonormal N3D, which is used throughout this library, and the *Schmidt semi-normalized* one, known as *SN3D* in ambisonic literature. Conversion between the two is trivial and given by

$$x_{nm|\mathrm{SN3D}} = x_{nm|\mathrm{N3D}}/\sqrt{2n+1} \quad (29)$$

$$x_{nm|\mathrm{N3D}} = \sqrt{2n+1}x_{nm|\mathrm{SN3D}}. \quad (30)$$

Conversion between the two specifications can be performed with the blocks *converters.n3d2sn3d* and *converters.sn3d2n3d*.

3.3.4 Acoustic Visualization

It is possible to extract information from the ambisonic signals about the directional distribution of sound in the scene. One such approach is based on the acoustic active intensity, expressing the net flow of energy through the notional center of the sound scene, and the diffuseness, expressing the portion of energy that is not propagating due to either modal or diffuse behavior. These parameters require only the first-order ambisonic signals, which correspond to acoustic pressure and velocity, see for example [20]. Examples of how diffuseness and intensity may be used for visualizations sound sources in the scene can be found in the code examples [6]. Their broadband version can be extracted using the *intensityAnalyzer* block, computed at each processing block of WAA. More refined visualizations can be obtained if the intensity and diffuseness is computed in frequency bands, e.g. using the biquad filter structures of WAA.

3.4 Decoding filter generation and SOFA integration

Binaural decoding is implemented in the *binDecoder* block, paired with both *hoaLoader* and *hrirLoader* blocks that handle user-defined binaural decoding filters loading.

Using the *hoaLoader*, users can choose both HRIR set and decoding approach. An additional Matlab script based on the Higher-Order-Ambisonics library [21] is available for offline generation of HOA decoding filters. Some decoding filters are already included in the repository, based on LISTEN HRTF sets [22], derived using the ALLRAD method of Eq. 17. Both decoding approaches mentioned in Sec. 2.6.2 were tested for derivation of decoding filters. The virtual loudspeaker approach was found superior in terms of preserving timbre than the direct approach of Eq. 20, which suffered from severe high-frequency loss at lower orders. Note that an approximate timbre correction can be applied to counteract this effect, as proposed in [23].

The *hrirLoader* block on the other hand allows for on-the-fly HRIR filters loading, internally converted to HOA decoding filters to be used by the *binDecoder* block. The *hrirLoader* implementation is based on the *HrftSet* class of the binauralFIR library [24], featuring server-based HRIR loading, granting access to an extensive choice of HRTF sets without cluttering the library itself. At the time of writing, the *hrirLoader* relies on local JSON embedded HRTF set loading, awaiting for the IRCAM OpenDAP SOFA server [25] publication.

4 Applications

The library is relevant to any web application that delivers or involves immersive content. Some examples of special interest are highlighted below:

- Reproduction of spherical audio and video for telepresence. In this scenario an ambisonic audio stream is delivered to the client along with a spherical video. The audio part is rendered binaurally at the target platform including head-rotation information, giving a convincing sense of presence.

- Reproduction of audio-only or audiovisual compositions, with the sound part encoded into a few ambisonic channels using the provided tools, and broadcasted to multiple clients with binaural rendering done independently on each one of them.

- Web VR/AR applications in which the audio components are updated in real-time and encoded into ambisonic streams, avoiding costly

binaural rendering of multiple sources and reverberation, while still peforming rotation of the sound scene.

- Web video games with immersive spatial sound.
- Interactive visualization driven by spatial properties of the sound scenes, for extracting acoustic information or for artistic uses.

Some basic examples highlighting some of these applications are included in the code repository [6].

References

[1] M. A. Gerzon, "Periphony: With-height sound reproduction," *Journal of the Audio Engineering Society*, vol. 21, no. 1, pp. 2–10, 1973.

[2] S. Moreau, S. Bertet, and J. Daniel, "3D sound field recording with higher order ambisonics – objective measurements and validation of spherical microphone," in *120th Convention of the AES*, (Paris, France), 2006.

[3] M. 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.

[4] 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.

[5] W3C, "Web Audio API," 12 2015. https://www.w3.org/TR/webaudio/.

[6] A. Politis and D. Poirier-Quinot, "JSAmbisonics: A Web Audio library for interactive spatial sound processing on the web." https://github.com/polarch/JSAmbisonics.

[7] J. Ivanic and K. Ruedenberg, "Rotation matrices for real spherical harmonics. direct determination by recursion," *The Journal of Physical Chemistry*, vol. 100, no. 15, pp. 6342–6347, 1996.

[8] M. A. Blanco, M. Flórez, and M. Bermejo, "Evaluation of the rotation matrices in the basis of real spherical harmonics," *Journal of Molecular Structure: THEOCHEM*, vol. 419, no. 1, pp. 19–27, 1997.

[9] R. H. Hardin and N. J. Sloane, "McLaren's improved snub cube and other new spherical designs in three dimensions," *Discrete & Computational Geometry*, vol. 15, no. 4, pp. 429–441, 1996.

[10] 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.

[11] M. J. Evans, J. A. S. Angus, and A. I. Tew, "Analyzing head-related transfer function measurements using surface spherical harmonics," *The Journal of the Acoustical Society of America*, vol. 104, no. 4, pp. 2400–2411, 1998.

[12] D. N. Zotkin, R. Duraiswami, and N. A. Gumerov, "Regularized HRTF fitting using spherical harmonics," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, (New Paltz, NY, USA), 2009.

[13] G. D. Romigh, D. S. Brungart, R. M. Stern, and B. D. Simpson, "Efficient real spherical harmonic representation of head-related transfer functions," *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, no. 5, pp. 921–930, 2015.

[14] M. Noisternig, T. Musil, A. Sontacchi, and R. Höldrich, "3D binaural sound reproduction using a virtual ambisonic approach," in *IEEE Int. Symposium on Virtual Environments, Human-Computer Interfaces and Measurement Systems (VECIMS)*, (Lugano, Switzerland), 2003.

[15] B. Wiggins, I. Paterson-Stephens, and P. Schillebeeckx, "The analysis of multi-channel sound reproduction algorithms using HRTF data.," in *In 19th Int. Conf. of the AES*, 2001.

[16] A. Politis, "A JavaScript library for the Spherical Harmonic Transform." https://github.com/polarch/Spherical-Harmonic-Transform-JS.

[17] E. W. Weisstein, "Associated legendre polynomial." http://mathworld.wolfram.com/AssociatedLegendrePolynomial.html.

[18] G. W. Elko, "Differential microphone arrays," in *Audio signal processing for next-generation multimedia communication systems*, pp. 11–65, Springer, 2004.

[19] Blue Ripple Sound, "HOA Technical Notes – B-format." http://www.blueripplesound.com/b-format.

[20] A. Politis, T. Pihlajamäki, and V. Pulkki, "Parametric spatial audio effects," in *Int. Conf. on Digital Audio Effects (DAFx)*, (York, UK), 2012.

[21] A. Politis, "Higher Order Ambisonics library," 2015. https://github.com/polarch/Higher-Order-Ambisonics.

[22] O. Warusfel, "Listen HRTF database," *online, IRCAM and AK, Available: http://recherche. ircam. fr/equipes/salles/listen/index. html*, 2003.

[23] J. Sheaffer, S. Villeval, and B. Rafaely, "Rendering binaural room impulse responses from spherical microphone array recordings using timbre correction," in *EAA Joint Symposium on Auralization and Ambisonics*, (Berlin, Germany), 2014.

[24] T. Carpentier, "Binaural synthesis with the Web Audio API," in *1st Web Audio Conference (WAC)*, 2015.

[25] IRCAM, "IRCAM OpenDAP Server. (to be published soon).."

View publication stats