EBU

OPERATING EUROVISION AND EURORADIO

Tech 3369

BINAURAL EBU ADM RENDERER (BEAR) FOR OBJECT-BASED SOUND OVER HEADPHONES

SOURCE: EBU AS GROUP

SPECIFICATION

Geneva
March 2023

This page is intentionally blank.

Contents

Binaural EBU ADM Renderer (BEAR) for object-based sound over headphones

EBU Committee	First Issued	Revised	Re-issued
TC	2023		

Keywords: ADM, Audio Definition Model, NGA, Next Generation Audio, Object-based, Binaural, Renderer.

1. Introduction

The BEAR (Binaural EBU ADM Renderer) is a renderer for audio content encoded using the ADM (Audio Definition Model [1]), based on the EAR (EBU ADM Renderer [2]), which produces binaural signals suitable for listening on headphones, rather than loudspeaker signals.

This document aims to describe how the BEAR works. § 2 gives an overview of the signal path and control structures, showing how the components fit together, while § 3 contains a more detailed description of each component.

A reference implementation of the BEAR built using the VISR framework[1] is available at https://github.com/ebu/bear. Where this document refers to file paths, they are files in that repository (version v0.0.1-pre, commit 5eececb at time of writing).

2. Overview

The renderer consists of two main parts: the signal processing, and the control component that calculates the parameters for the signal processing given the metadata inputs.

These are implemented as sets of components with multiple audio and parameter inputs and outputs, connected in a directed acyclic graph[2] structure.

All components process data once per period, which consists of a fixed number of audio frames defined at initialisation time[3]. Components run in order, so the inputs for each component are the outputs of other components within the same period. This structure therefore adds no latency to the system beyond the latencies added by individual components, and the inherent latency of processing audio in blocks of more than one frame.

In the VISR implementation the top-level structure connecting the control and signal processing components is defined in visr_bear/src/top.cpp.

[1] Versatile Interactive Scene Renderer; see https://github.com/s3a-spatialaudio/VISR

[2] For example, see https://en.wikipedia.org/wiki/Directed_acyclic_graph

[3] For example, when running at 48 kHz with a period size of 128 samples, there are 375 periods per second. A frame refers to one sample per channel at one point in time.

2.1 Signal Path

The signal processing structure is shown in Figure 1. The parts of this corresponding to the three supported input types are described below, and the individual components are described in § 3.9. Symbols indicate the exact values used in each block, and are defined in § 3.

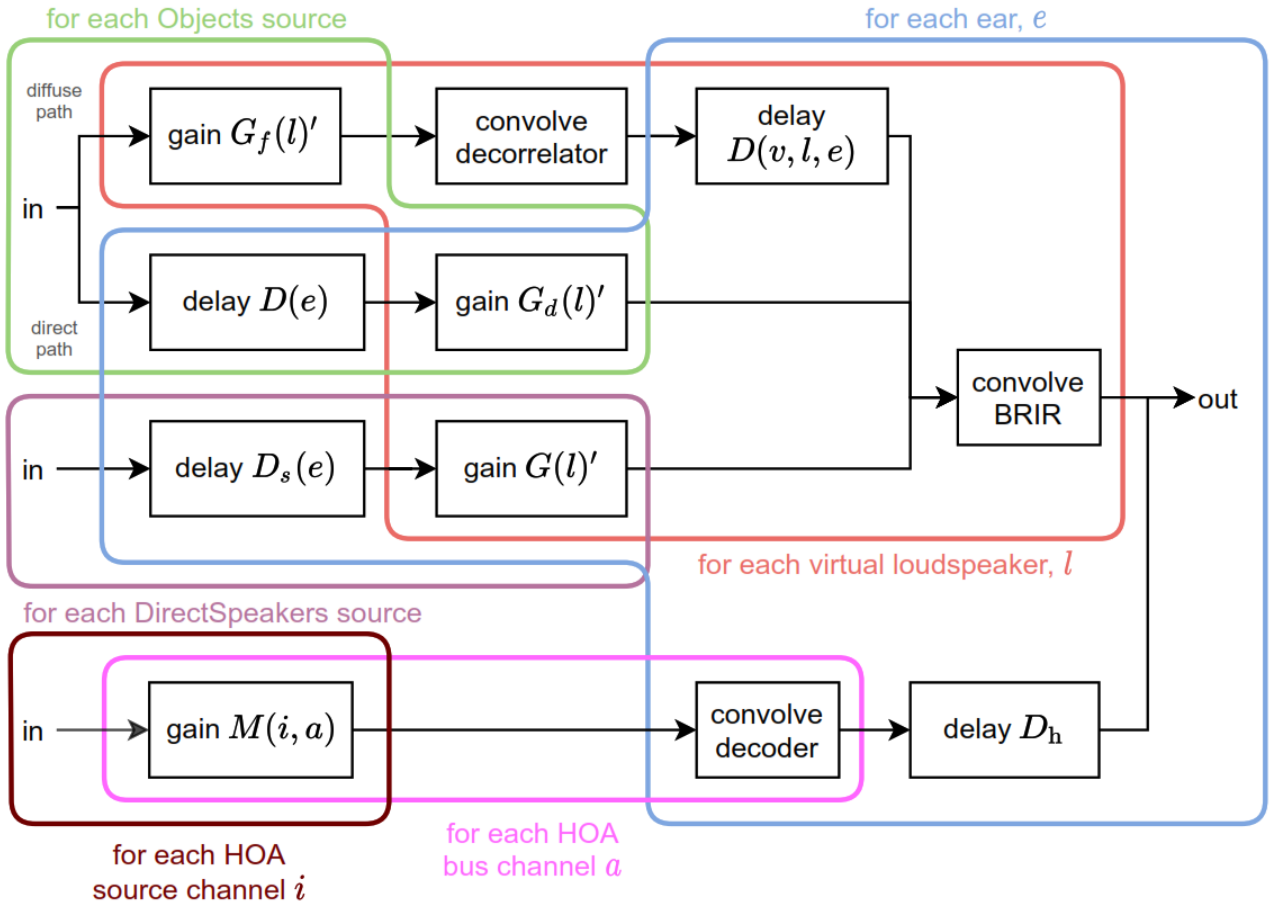

Figure 1: BEAR signal processing

"For each" boxes indicate the dimensionality of the signals and processes which they surround. Signals that enter a box are replicated along the indicated dimension, while signals leaving a box are summed.

In the VISR implementation this structure is defined in visr_bear/src/dsp.cpp.

2.1.1 Objects

The primary structure used to render Object-based audio content is virtual-loudspeaker rendering, in which loudspeaker signals are convolved with Binaural Room Impulse Responses (BRIRs).

In this implementation, delays are extracted from the BRIRs and re-added using fractional delay lines; this reduces comb filtering when Objects are between virtual loudspeaker positions. Object rendering is performed in two paths, direct and diffuse.

In the direct path, a separate per-ear delay is applied for each Object input channel, followed by a gain matrix to produce a set of virtual loudspeaker signals for each ear. These are convolved with BRIRs to produce the output.

In the diffuse path, the input passes through a gain matrix, to produce a set of virtual loudspeaker signals. These pass through a bank of decorrelation filters, and static per-ear delays are applied, before being convolved with BRIRs to produce the output.

Design notes:

- The per-ear delays in the direct path are also used to compensate for the delay introduced by the decorrelation filters.
- In the diffuse path, the per-ear delays are added for each virtual loudspeaker channel, rather than for each Objects channel; this is because in informal listening having a separate ITD for each decorrelated loudspeaker channel was found to be important for the perception of diffuse sources.
- In the VISR implementation, the direct gains are split into a separate gain matrix for the left and right channel, because the left and right gains are the same, and this halves the matrix size.

2.1.2 DirectSpeakers

The DirectSpeakers signal path is the same as the Objects direct path.

2.1.3 HOA

HOA (Higher-Order Ambisonics) rendering is performed with a decoder matrix separate from the Objects and DirectSpeakers paths.

The input audio first passes through a gain matrix that routes the input channels to the corresponding bus channels and that performs rotation for head-tracking. It then passes through the decode matrix and is delayed to match the Objects and DirectSpeakers paths and is then mixed into the output.

2.2 Control

The BEAR control architecture is shown in Figure 2. This receives ADM metadata and listener updates and produces the various gain and delay values that control the behaviour of the signal path described above.

In the VISR implementation this structure is defined in visr_bear/src/control.cpp.

2.2.1 Common Components

BRIR Selection

The listener orientation is used to select the closest set of BRIRs available. This results in a BRIR index, and a 'residual' orientation – the orientation relative to the orientation of the head in the chosen BRIR set. See § 3.5.1.

2.2.2 Objects Components

Get Static Delays

The delays used by the Objects diffuse path are looked up based on the chosen BRIR set. See § 3.6.3.

Objects Gain Calc

ADM Objects metadata is modified to account for the listener orientation, and *libear* is used to calculate the corresponding direct and diffuse gains. See § 3.6.1.

Calc Direct Delays

The direct and diffuse gains are used to calculate the per-ear delays in the direct path, based on a weighted average of the delays corresponding to the virtual loudspeakers activated by the gains. See § 3.6.2.

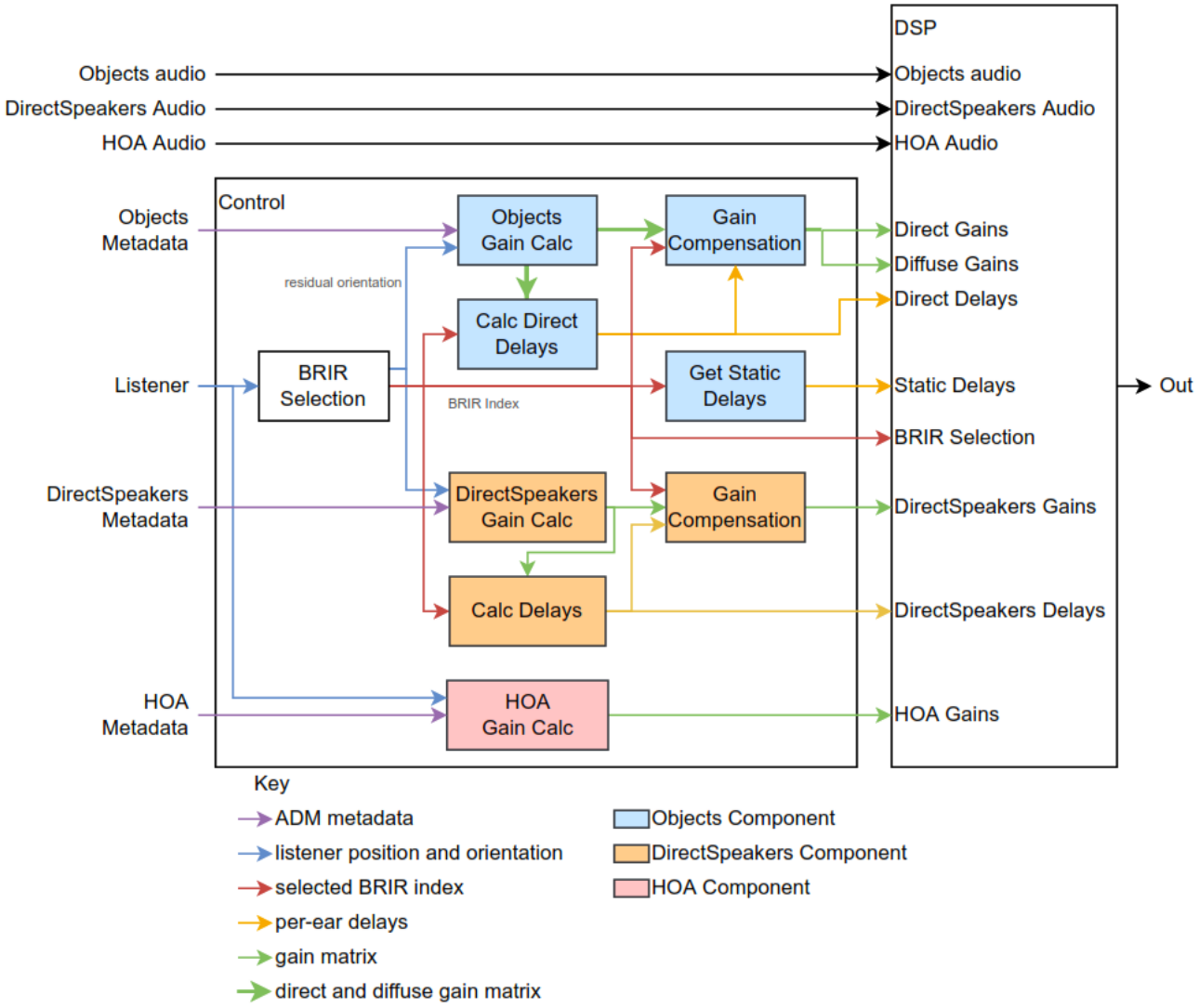

Figure 2: BEAR Control Architecture

Gain Compensation

The direct and diffuse gains, per-ear delay and chosen BRIR set are used to modify the direct and diffuse gains to compensate for changes in overall gain caused by varying correlation between different pairs of BRIRs. See § 3.6.4.

2.2.3 DirectSpeakers Components

DirectSpeakers Gain Calc

Modifies ADM metadata according to the listener orientation, then uses *libear* to calculate loudspeaker gains. See § 3.7.1.

Calc Delays

Performs the same function as the Objects Calc Direct Delays component, but with only direct gains. See § 3.7.2.

Gain Compensation

Performs the same function as the Objects Gain Compensation component, but with only direct gains and no per-ear delay. See § 3.7.3.

2.2.4 HOA

HOA Gain Calc

This calculates the values for the HOA routing matrix, incorporating channel mapping, normalisation conversion and rotation. See § 3.8.

2.3 BRIR Set

The default set of BRIRs are derived from those described in [3], with the following processes applied:

- Time alignment: delays are removed and stored separately.
- Windowing: raised cosine window applied to shorten BRIRs to 60 ms.
- Layout: 9+10+3 with two extra lower-layer loudspeakers at ±135°.

Pre-processed BRIRs are available in the renderer data file; see § 3.1.

3. Detailed Component Description

3.1 Data File

Data required by the renderer is provided in a file which is read when the renderer is initialised. The data in this file are listed in § 3.2.

The reference source code contains a default data file, but other files may be preferred, depending on the application.

3.2 Definition of Terms

The following values are static, provided by the renderer data file:

f_s:	Sampling frequency in Hz.
L:	BRIR filter length
$H(v, l, e, s)$:	BRIR sample s for view v, virtual loudspeaker l, ear e.
$D(v, l, e)$:	Delay in seconds for BRIR with view v, virtual loudspeaker l, ear e.
$F(l, s)$:	Decorrelation filter sample s for virtual loudspeaker l.
N_f:	Index of frontal virtual loudspeaker.
D_f:	Delay in seconds caused by the decorrelation filters.
$V(v)$:	View vector for index v.
$R(v)$:	Head orientation for BRIR set with index v.
$P(d, v, l_1, l_2, e)$:	Gain compensation factor matrix entry for integer sample delay d-1, view v, loudspeaker indices l_1 and l_2, and ear e.
$A(a, e, s)$:	HOA filter matrix sample s for channel a (1 to N_a) and ear e.
D_h:	Static delay in seconds added to HOA path to align it with the other paths.

The following indices are used when describing processes to be applied:

l:	Virtual loudspeaker index in range 1… N_l.
v:	View index in range 1… N_v.
e:	Ear index; 1 for left, 2 for right.
a:	HOA channel index from 1 to N_a.

Inputs to renderer processes:

L_p:	Listener head position, in metres and ADM coordinates.
L_o:	Listener head orientation.

3.3 Coordinate System

The renderer uses the ADM coordinate system, described in § 8 of BS.2076-2.

Unless otherwise stated, positions are represented as Cartesian coordinates as column vectors: $(X,Y,Z)^T$.

3.4 Head Position and Orientation

The position and orientation of the listener's head is used by the renderer to give the impression of movement within a sound scene.

The listener's head orientation may be represented using a quaternion (preferred), rotation matrix, or Euler angles (strongly discouraged).

The listener position and orientation have the following properties:

- Given a world position p, the equivalent position relative to the listener p' is:

$$p' = L_o\,(p - L_p)$$

- Given a position p' relative to the listener (for example $p' = (0,1,0)^T$ is 1 metre in front of the listener in the direction that they are looking), the equivalent world position p is:

$$P = L_p + L_o' p'$$

Additionally, orientations can be composed associatively, for orientations Q and R and position p:

$$(Q\ R)\,p = Q\,(R\ p)$$

3.5 Shared Components

3.5.1 BRIR Selection

The BRIR set used is the one closest to the listener view. Given the listener orientation L_o, the listener view L_v is:

$$L_v = L_o'\,(0,1,0)^T$$

The selected BRIR view v is:

$$v = \mathrm{argmax}_i\ V(i)\ \cdot L_v$$

If multiple BRIR sets are closest, the choice is implementation-defined.

The residual listener position L_p' and orientation L_o' are:

$$\begin{aligned} L_o' &= R(v)'\,L_o \\ L_p' &= L_p \end{aligned}$$

This implies that the BRIR set used may change up to once per period. Crossfading over one period is applied to mask these changes, see § 3.9.3.

3.5.2 Loudspeaker Layout

The virtual loudspeaker layout is based on system H defined in BS.2051-2, with the following changes:

LFE1 and LFE2 channels are removed.

The following loudspeakers are added:

Name	Azimuth	Elevation	Elevation Range
B+135	135°	-30°	-30°…-15°
B-135	135°	-30°	-30°…-15°

The loudspeaker positions are adapted to match the loudspeaker positions used to record the BRIR set (which are within the allowed ranges).

3.6 Objects

3.6.1 Calculation of Gains

The EAR Objects gain calculator [2, § 7.3] is used to calculate gains for the virtual loudspeakers, using the loudspeaker layout described in § 3.5.2.

The overall structure is shown in Figure 3. Gains are calculated once per sample period; these are interpolated by the gain matrices over the length of the sample period, so that the calculated gains are used for the last sample in the period.

The input to the gain calculator is a sequence of ObjectsTypeMetadata objects. Given the time of the last sample in the period, two items in this sequence (shown as OTM A and B) and a mixing factor *m* are identified, such that the output gains will match the behaviour specified in [2, § 7.2].

For example, if the input sequence contains 3 ObjectsTypeMetadata objects[4] [b_1, b_2, b_3], and the end of the period is halfway through b_2, then $A = b_2$, $B = b_3$ and $m = 0.5$.

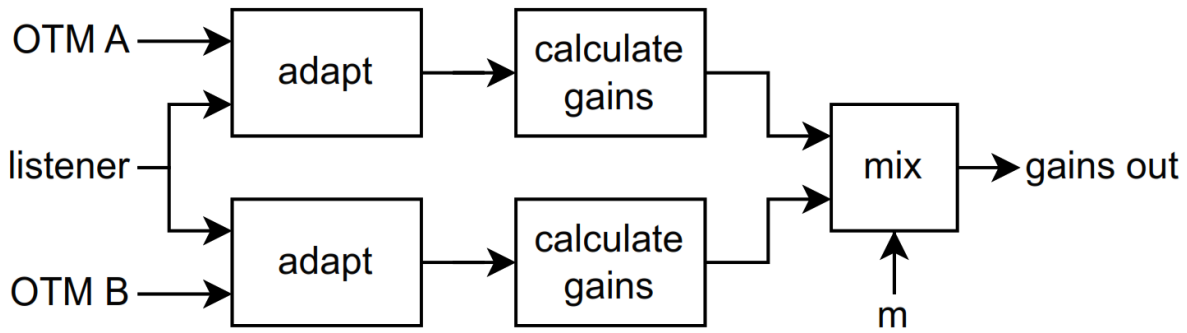

Figure 3: Gain calculator structure

OTM refers to ObjectsTypeMetadata, which primarily contain data from a single audioBlockFormat.

These two ObjectsTypeMetadata objects are modified to account for the listener's head orientation and position, and gains are calculated for each. Finally the two sets of gains are mixed together with the ratio (1 - *m*) : *m*.

[4] Assuming they are adjacent, of non-zero length, and have no jumpPosition flag set.

These steps are described in more detail below.

3.6.1.1 ObjectsTypeMetadata Adaptation

For an ObjectsTypeMetadata object *b*, the following steps are applied:

1. The absolute distance scale is determined: *d* is the value of the absoluteDistance parameter of the last audioPackFormat (closest to the audioChannelFormat) associated with *b* which has one, or None if no absoluteDistance parameter is found, or the found parameter is negative (as in [1, § 5.5.2]).

2. The original position *p* is extracted, and converted to Cartesian if necessary:

$$p = \text{cart}(b.\text{block\_format.position})$$

3. If *d* is not None, the adjusted position *p'* is:

$$p' = \frac{L_o(dp - L_p)}{d}$$

otherwise:

$$p' = L_o p$$

4. *b*.block_format.position is set to $\{\text{azimuth}(p'), \text{elevation}(p'), \| p' \|_2\}$

5. The channelLock flag is cleared.

6. Any zone entries are removed.

3.6.1.2 Calculation of Gains

Gains are calculated using the gain calculator described in [2, § 7.3], using the loudspeaker layout described in § 3.5.2.

3.6.1.3 Gain Mixing

Given the direct and diffuse gains for *A*, $G_{d,A}(l)$ and $G_{f,A}(l)$, and the direct and diffuse gains for B, $G_{d,B}(l)$ and $G_{f,B}(l)$, the resulting gains are:

$$\begin{aligned} G_d(l) &= (1-m)\, G_{d,A}(l) + m\, G_{d,B}(l) \\ G_f(l) &= (1-m)\, G_{f,A}(l) + m\, G_{f,B}(l) \end{aligned}$$

3.6.2 Calculation of Direct Delays

The procedure for calculating the direct delay *D(e)* for ear e given the direct gains $G_d(l)$, diffuse gains $G_f(l)$ and view index *v* is as follows:

The direct and diffuse gains are summed, so that the delays are consistent when the diffuse parameter changes between diffuse = 1 and diffuse <1:

$$G_s(l) = G_d(l) + G_f(l)$$

The delay is either the weighted mean of the delays for each BRIR if the gains are non-zero, or the delay for the front virtual loudspeaker otherwise[5]. The delay caused by the decorrelation filters D_f is added to ensure that the direct and diffuse paths are aligned.

$$s = \sum_l G_s\,(l)$$

If $s > \epsilon$:

$$D(e) = \frac{\sum_l G_s(l) D(v, l, e)}{s} + D_f$$

Otherwise:

$$D(e) = D\big(v, N_f, e\big) + D_f$$

3.6.3 Calculation of Static Delays

In the diffuse path, the delay for view *v*, loudspeaker *l* and ear *e* is just the BRIR delay:

$$D(v, l, e)$$

3.6.4 Gain Compensation

Given the direct and diffuse gains $G_d(l)$ and $G_f(l)$, the BRIR view index v and the direct delays $D(e)$, the sets of gains $G_d(l)'$ and $G_f(l)'$ are calculated, based on a compensation gain g_c derived from the desired gain g_d and expected gain g_r, described below:

$$g_c = \frac{g_d}{g_r}$$

$$\begin{aligned} G_d(l)' &= g_c\, G_d(l) \\ G_f(l)' &= g_c\, G_f(l) \end{aligned}$$

In these descriptions, $G_c(l_i)$ and $G_c\big(l_j\big)$ refer to the concatenated direct and diffuse gains:

$$G_c(l_i) = \begin{cases} G_d(l_i) & l_i \le N_l \\ G_f(l_i - N_l) & l_i > N_l \end{cases}$$

3.6.4.1 Expected Gain

The expected gain is a sum over both ears:

$$g_r = \sqrt{\sum_e g'_{r,e}}$$

[5] Possible improvement: use the gains calculated before the gain parameter is applied.

The per-ear contribution is:

$$g'_{r,e} = p \sum_{l_i} \sum_{l_j} G_c(l_i) G_c(l_j) P(d_a, v, l_i, l_j, e) + (1-p) \sum_{l_i} \sum_{l_j} G_c(l_i) G_c(l_j) P(d_b, v, l_i, l_j, e)$$

Where p, d_a and d_b are chosen so that d_a and d_b are valid indices into the factor matrix P (described in § 3.2) and:

$$d = p\, d_a + (1-p)\, d_b \quad \text{and} \quad 0 \le p \le 1$$

Where *d* is the delay in samples relative to the minimum possible delay:

$$d = f_s(D(e) - D_f)$$

3.6.4.2 Desired Gain

The desired gain is similar to the expected gain, but does not take the interactions between BRIRs into account:

$$g_d = \sqrt{\sum_e \sum_{l_i} G_c(l_i)^2 P(0, v, l_i, l_i, e)}$$

3.7 DirectSpeakers

3.7.1 Calculation of DirectSpeakers Gains

The input to the gain calculator is a sequence of DirectSpeakersTypeMetadata objects. Given the time of the last sample in the period, the gain calculator finds a DirectSpeakersTypeMetadata block b which contains this time, according to [2, § 6.5].

If no DirectSpeakersTypeMetadata block containing the time is found, then the output gains are zero, otherwise the following process is used:

1. If *b* is an LFE channel according to [2, § 8.2], then the channel is discarded.
2. Screen edge lock is applied according to [2, § 8.4].
3. The position relative to the listener, *p'* is determined.

   $$p' = L_o p$$

   where *p* is the nominal position of *b*.
4. The gains are calculated using the polar point-source panner as described in [2, § 6.1], using the loudspeaker layout described in § 3.5.2.

3.7.2 Calculation of DirectSpeakers Delays

Given gains $G(l)$ and view index v, $D_s(e)$ is calculated as $D(e)$ in § 3.6.2, except that $G_s(l) = G(l)$.

3.7.3 Gain Compensation

Given gains $G(l)$, view index v and per-ear delay $D(e)$, compensated gains $G(l)'$ are calculated as in § 3.6.4, except that:

$$\begin{aligned} G(l)' &= g_c\, G(l) \\ G_d(l) &= G(l) \\ G_f(l) &= 0 \end{aligned}$$

3.8 HOA

HOA rendering is performed using a static matrix of filters; the HOA gain calculator produces a matrix which maps from the input channels of the renderer to a bus matching the expected input format of the filters, while compensating for head rotation.

For a HOAInput object *b* with N_b channels, the following process is used:

- An N_b by N_a matrix M' is created; initially containing all zeros.
  For each input channel *i* in *b* with order *n*, degree *m* and normalisation norm, set:

$$M'\big(i,\mathrm{ACN}(n,m)\big) = \frac{N_{\mathrm{SN3D}_n^{|m|}}}{N_{\mathrm{norm}_n^{|m|}}}$$

  where *N* … are the normalisation functions defined in § 11.2 of BS.2076-2 and:

$$\mathrm{ACN}(n,m) = n^2 + n + m + 1$$

- The N_a by N_a HOA rotation matrix R corresponding to L_o is determined. This may be calculated using any method, but must have the property:

$$R\,\mathbf{Y}(p) = \mathbf{Y}(L_o\,p)$$

  for all unit vectors p, where $\mathbf{Y}(p)$ is a column vector containing the HOA encoding of p, i.e.

$$\mathbf{Y}(p)_{\mathrm{ACN}(n,m)} = Y_n^m(p)$$

  Where $Y_n^m(p)$ is as defined in § 11.1 of BS.2076-2, with an appropriate Cartesian to polar conversion.

  The matrix *M* is used to map from channels in b to the input of the static decode filters:

$$M = RM'$$

3.9 DSP Components

Rather than exactly specifying the signal processing to be applied, general specifications are given to allow for some implementation flexibility.

The components are specified in terms of single channels, but where inputs are shared between components, or outputs are mixed, the efficiency may be improved by implementing multi-channel components. For example, the BRIR convolution is best implemented as a 2 × N_l-input 2-output convolution matrix.

3.9.1 Fractional Delay Lines

Fractional delay lines accept one channel of audio samples and a delay parameter (in seconds) and produce one channel of audio samples with the applied delay.

The delay is specified once per period and is linearly interpolated over one period. Delay lines must use high quality fractional sample interpolation, sufficient to avoid zipper noise; for example 3rd order Lagrange interpolation according to [4].

The specified delays are greater than or equal to zero seconds, but high-quality delay lines have some inherent delay, so zero delay is not achievable. Additional delay may be added, provided that the same additional delay is included in all paths through the renderer.

3.9.2 Gains

Gain blocks accept one channel of audio samples and a gain parameter, and result in one channel of audio samples with the specified gain applied.

The gain is to be linearly interpolated over one period.

3.9.3 Convolution

Convolution blocks apply discrete convolution with a specified FIR filter.

In some convolution blocks the filters are updated dynamically. This must be implemented in a way which avoids steps in the output, by crossfading at the inputs or outputs of the filters, or some other method. Crossfading should take one period[6] to match the gains and delays.

It is assumed that convolution blocks will not add any delay (beyond that present in the filter); if a method is used which adds delay, this must be accounted for, so that the delay is the same in all paths through the renderer.

3.9.3.1 BRIR Convolution

Given the view selection v calculated in § 3.5.1, the BRIR convolution process for ear e and virtual loudspeaker l convolves the input signal with an impulse response given by $H(v, l, e, s)$ at sample s.

3.9.3.2 Decorrelator Convolution

The decorrelator convolution process for virtual loudspeaker l convolves the input signal with an impulse response given by $F(l, s)$ at sample s.

3.9.3.3 HOA Decoder Convolution

The HOA decoder convolution process for HOA channel a and ear e convolves the input signal with an impulse response given by $A(a, e, s)$ at sample s.

[6] The effects of crossfading will be visible for longer than one period if crossfading is applied at the inputs of the filters.

4. References

[1] ITU-R, “Recommendation ITU-R BS.2076-2 – Audio Definition Model.” International Telecommunication Union, 2019.

[2] ITU-R, “Recommendation ITU-R BS.2127 – Audio Definition Model renderer for advanced sound systems.” 2019.

[3] C. Pike and M. Romanov, “An impulse response dataset for dynamic data-based auralisation of advanced sound systems,” in 142nd AES convention, 2017.

[4] A. Franck, “Efficient algorithms and structures for fractional delay filtering based on lagrange interpolation,” Journal of the Audio Engineering Society, vol. 56, no. 12, pp. 1036–1056, 2009.