Audio Engineering Society

Conference Paper 68

Presented at the Conference on
Immersive and Interactive Audio
2019 March 27 – 29, York, UK

This paper was peer-reviewed as a complete manuscript for presentation at this conference. This paper is available in the AES E-Library (http://www.aes.org/e-lib) all rights reserved. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Journal of the Audio Engineering Society.

Efficient Encoding and Decoding of Binaural Sound with Resonance Audio

Marcin Gorzel[1], Andrew Allen[1], Ian Kelly[1], Julius Kammerl[1], Alper Gungormusler[1], Hengchin Yeh[1], and Francis Boland[1,2]

[1]*Google Inc.*
[2]*Trinity College Dublin*

Correspondence should be addressed to Marcin Gorzel (gorzel@google.com)

ABSTRACT

Resonance Audio is an open source project designed for creating and controlling dynamic spatial sound in Virtual & Augmented Reality (VR/AR), gaming or video experiences. It also provides integrations with popular game development platforms and digital audio workstations (as a preview plugin). Resonance Audio binaural decoder is used in YouTube to provide binaural rendering of 360/VR videos. This paper describes the core sound spatialization algorithms used in Resonance Audio and can be treated as a companion to the Resonance Audio C++ / MATLAB library source code.

1 Introduction

Resonance Audio is an open source project which allows developers to create and control complex virtual auditory environments in real-time. The Resonance Audio Software Development Kit (SDK) consists of the C++ library, MATLAB library of helper Digital Signal Processing (DSP) functions and also platform integrations which allow for immediate use of Resonance Audio within game engines and audio / video editing software. It supports all major platforms (Android, iOS, Windows, MacOS, Linux) and addresses a range of requirements for efficient high quality audio rendering, especially on mobile platforms, such as: thread-safety, non-blocking APIs (critical in VR/AR to ensure high frame rate and maximum CPU utilization), minimum audio processing latency (no buffered processing of audio data, single-threaded audio processing pipeline) and a small binary footprint.

Resonance Audio users can create virtual sound sources to be rendered in a virtual 3D space with a fine control over sources' directivity patterns, occlusion or near-field settings. They can also simulate specific acoustic environments via early reflections and late reverberation.

The "spatializer" is the core concept of the Resonance Audio system which encapsulates algorithms responsible for efficient encoding of sound events into an intermediate format - Ambisonics [1, 2], manipulation and, finally, binaural decoding of Ambisonic soundfields.

Ambisonics is a compact, efficient, scene-based, spatial representation of auditory events around a listener. One of its advantages is scalability, which means that an arbitrarily complex auditory scene can be represented

at a constant storage and computational cost with an arbitrary level of spatial accuracy, as will be explained in more detail in Section 1.2. This representation is used to render an approximation of the original soundfield using either loudspeakers or headphones. The decoding part only is used in cinematic 360/VR video experiences (like YouTube 360/VR) where the spatial audio content is streamed along with the video content.

This paper describes the core algorithms used in the Resonance Audio's spatializer. It is organized as follows: The reminder of this section explains the conventions used in this paper and provides background information about encoding, decoding and binaural reproduction of spatial audio using Ambisonics. Section 1.2.1 describes a novel method of efficient encoding of sound sources into Ambisonic soundfield representation. It also talks about controlling the angular spread of sound sources. Section 3 describes an optimized method of binaural decoding of Ambisonic soundfields based on pre-computed, spherical harmonic-encoded binaural filters. Finally, Section 4 summarizes the paper and suggest some future research and development directions.

1.1 Conventions Used in Resonance Audio

1.1.1 3D Coordinate System

The 3D coordinate system used in this paper is presented in Figure 1. It uses x, y and z axes pointing to the front, left and up respectively. ϕ is the azimuth angle with an anti-clockwise rotation and θ is the elevation angle calculated from the x-y plane. The conversion between spherical and Cartesian coordinates is defined as:

$$\begin{aligned} x &= \|\vec{r}\|\cos(\phi)\cos(\theta) \\ y &= \|\vec{r}\|\sin(\phi)\cos(\theta) \\ z &= \|\vec{r}\|\sin(\theta) \end{aligned} \tag{1}$$

1.1.2 Ambisonic Channel Sequence and Normalization

Resonance Audio uses the AmbiX Ambisonic convention for encoding and decoding of sound objects, as defined in [4]. It uses Schmidt semi-normalized harmonics (SN3D normalization) and Ambisonic Channel Number (ACN) sequencing of spherical harmonics (SHs) - the basis functions used in Ambisonics.

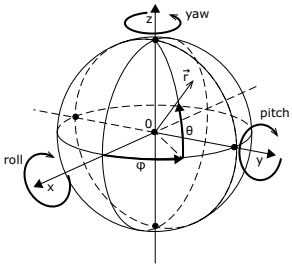

Fig. 1: 3D spherical coordinate system used in this paper, as proposed in [3]

1.1.3 Other Conventions

Traditional maths and physics use the terms *degree* (usually denoted as l) and *order* (usually denoted as m) where $0 \leq l \leq +\infty$ and $-m \leq l \leq m$ to describe SHs. In contrast, in *First Order Ambisonics* (FOA) or *Higher Order Ambisonics* (HOA), the terms *order* (usually denoted as n) and *degree* (usually denoted as m) are used in the opposite meaning [4].

The Resonance Audio source code, whenever we are concerned with spherical harmonics directly, uses the traditional naming convention (for example a spherical harmonic of degree l and order $-m \leq l \leq m$).

However, when it is more natural to use the Ambisonic convention, we use the word *Ambisonic* explicitly, for example *Ambisonic order* and (less often) *Ambisonic degree*.

To obtain the ACN from a given Ambisonic order and degree, one has to do the following:

$$ACN = n^2 + n + m \tag{2}$$

Conversely, to retrieve the Ambisonic order and degree information from a given Ambisonic channel in an ACN sequence:

$$n = \lfloor \sqrt{ACN} \rfloor \tag{3}$$

$$m = ACN - n^2 - n \tag{4}$$

where $\lfloor \cdot \rfloor$ describes rounding $\cdot$ down to the nearest integer equal or less than $\cdot$.

1.2 Higher Order Ambisonic Soundfields

1.2.1 Encoding Ambisonics

To encode a sound source into an Ambisonic soundfield, we take an outer product of the monophonic pressure signal corresponding to the sound source, $\mathbf{s}$, and the vector $\mathbf{y}$ of SH coefficients $Y_n^m(\phi,\theta)$:

$$\mathbf{B} = \mathbf{s} \otimes \mathbf{y} \tag{5}$$

Spherical harmonic coefficients are defined as:

$$Y_n^m(\phi,\theta) = N_n^{|m|} P_n^{|m|}(\sin(\theta)) \begin{cases} \cos(|m|\phi) & \text{if } m \geq 0 \\ \sin(|m|\phi) & \text{if } m < 0 \end{cases} \tag{6}$$

where ϕ is the source horizontal angle, θ is the source vertical angle, n is the Ambisonic order, m is the Ambisonic degree[1], $P_n^{|m|}$ are the associated Legendre functions (with the Condon-Shortley phase undone). $N_n^{|m|}$ is the SN3D normalization term which is computed according to the following formula:

$$N_n^{|m|} = \sqrt{(2-\delta_m)\frac{(n-|m|)!}{(n+|m|)!}} \tag{7}$$

Note that we skip the $\frac{1}{4\pi}$ term in the normalization which means that we are preserving the original source amplitude in the 0^{th} order Ambisonic channel.

1.2.2 Ambisonic Transformations

An arbitrary Ambisonic transformation can be described as:

$$\widetilde{\mathbf{B}} = \mathbf{B}\mathbf{T} \tag{8}$$

where the new, transformed Ambisonic signal $\widetilde{\mathbf{B}}$ is obtained by applying a matrix $\mathbf{T}$ to the Ambisonic input signal $\mathbf{B}$.

There are different types of transformations that can be applied to an Ambisonic soundfield [5]. For example, rotation can be applied to a single soundfield which can consist of multiple point sources. Another transformation used in Resonance Audio which is applied on a per-sound source basis is angular spread control (see Section 2.2).

Sound sources encoded into an Ambisonic representation are assumed to be dynamic, i.e. their position relative to the listener's head can change at an arbitrary rate and independently of the other sound sources. Hence, to re-position an individual sound source in the 3D space it suffices to re-encode it with updated SH coefficients.

There are situations, however, where rotating an Ambisonic soundfield is a better solution. One such situation is when multiple sound sources are used and when the sources have already been encoded into the Ambisonic representation, for example, using a microphone array.

FOA rotation is trivial and consists in applying a 3 x 3 rotation matrix to the velocity components of the soundfield whereas keeping the pressure component unmodified. This is equivalent to a simple vector rotation in $\mathbb{R}^3$. In Resonance Audio, this transformation is applied by the `FoaRotatorNode` with the underlying DSP implemented by the `FoaRotator` class.

HOA rotation is more complex since it requires rotations of vectors whose dimensionality is higher than 3. Numerous methods have been proposed to compute the required rotation matrices in $\mathbb{R}^4$ and above, for example in [5]. Resonance Audio uses an efficient computation of SH rotation matrices by recursion, used in other fields like physical chemistry or computer graphics [6, 7, 8]. HOA rotation is applied by the `HoaRotatorNode` with the underlying DSP implemented by the `HoaRotator` class.

1.2.3 Decoding Ambisonics

The decoding of an Ambisonic signal $\mathbf{B}$ is done by multiplying it with the inverse of the loudspeaker *re-encoding* matrix $\mathbf{L}$ which encodes each loudspeaker's direction into SHs:

$$\mathbf{L} = \begin{bmatrix} Y_0^0(\Phi_1,\Theta_1) & Y_0^0(\Phi_i,\Theta_i) & \dots & Y_0^0(\Phi_N,\Theta_N) \\ Y_1^{-1}(\Phi_1,\Theta_1) & Y_1^{-1}(\Phi_i,\Theta_i) & \dots & Y_1^{-1}(\Phi_N,\Theta_N) \\ \vdots & \vdots & \vdots & \vdots \\ Y_n^m(\Phi_1,\Theta_1) & Y_n^m(\Phi_i,\Theta_i) & \dots & Y_n^m(\Phi_N,\Theta_N) \end{bmatrix} \tag{9}$$

[1] Please see Section 1.1.3 for explanation of the conventions used.

where Φ_i and Θ_i are azimuth and vertical angles of the i^{th} loudspeaker in the array. The resultant loudspeaker signals $\mathbf{G}$ are obtained by:

$$\mathbf{G} = \mathbf{BD} \tag{10}$$

where $\mathbf{D}$ is a Moore-Penrose pseudo-inverse of $\mathbf{L}$:

$$\mathbf{D} = \mathbf{L}^{\dagger} = \mathbf{L}^{\mathbf{T}}(\mathbf{L}\mathbf{L}^{\mathbf{T}})^{-\mathbf{1}} \tag{11}$$

1.2.4 Frequency-dependent Decoder Optimization

In Ambisonics, the area of correct soundfield reconstruction is located around the so-called “sweet-spot". The size of this area depends on the frequency content of the soundfield and the Ambisonic order of a system. For example, Daniel *et al.* in [3] and [9] evaluated soundfield reconstruction errors for Ambisonic orders of 1 to 5 and proposed cut-off frequencies below which the Ambisonic reconstruction can be considered as accurate.

Gerzon proposed objective criteria for good localization in multi-loudspeaker systems in [10, 11] by evaluating velocity and energy vectors characterizing the soundfield. For example, low frequency localization for humans is mainly based on the interaural phase differences. On the other hand, in the mid- and high-frequency range, interaural level differences start to play a much greater role in localization. That is why, Gerzon suggested to optimize the velocity vectors at low frequencies and the energy vectors at mid- and high frequencies.

One way to apply these optimizations in Ambisonic playback is to use different decoder types in their most optimal frequency bands. Decoders that utilize different decoding schemes in different frequency ranges are known as shelf-filter decoders or dual-band decoders [12, 13, 14]. In such decoders, phase-matched shelf-filters are typically used to split the Ambisonic signal spectrum into low- and mid/high-frequency bands. The cross-over frequency of the filters is usually set so that the *basic* decoder operates only at low frequencies, where the soundfield reconstruction errors do not exist or are minimal. This decoder already maximizes the lengths of velocity vectors. Above the crossover frequency, a different decoding scheme is used which maximizes the lengths of energy vectors and thus, the energy concentration in the direction of an encoded sound source. This is known as the maximum-energy or *MaxRe* decoder.

Different decoding schemes are applied by changing the ratios of the pressure and velocity soundfield components at different Ambisonic orders. The closed-form formulae for obtaining (*MaxRe*) decoder coefficients (as well as other decoder types) have been given, for example, by Daniel in [3].

Resonance Audio implements dual-band decoders where cut-off frequencies are based on the soundfield reconstruction errors proposed by Daniel in [3]. The cross-over frequencies at Ambisonic orders 1 to 5 are set to the following values: 690Hz, 1250Hz, 1831Hz, 2423Hz, 3022Hz. These values can be modified by changing the `ambishelffilter` `MATLAB` function and re-generating the binaural Ambisonic decoders. The binaural Ambisonic decoders in Resonance Audio are pre-emphasized using shelf-filters which greatly reduces computational load at run-time (please see Section 3 for more details).

1.3 Binaural Reproduction of Ambisonic Soundfields

Binaural reproduction of Ambisonic soundfields is typically done using “virtual loudspeakers". This method was first introduced by McKeag and McGrath [15] and examples of its adoption can be found in work by Noisternig *et al.* [16] or Dalenback *et al.* [17].

In this method, loudspeaker signals are first generated by decoding an Ambisonic soundfield. Next, they are convolved with the Head Related Impulse Responses (HRIRs) corresponding to the physical loudspeaker locations. Lastly, left channel signals are summed together to create the left feed of a binaural signal. The same process is repeated for the right feed. For example, to obtain the left and right ear headphone signals $\mathbf{p_l}$ and $\mathbf{p_r}$ we have to perform:

$$\mathbf{p_l} = \sum_{i=1}^{N} \mathbf{h_{Li}} * \mathbf{g_i} \tag{12}$$

$$\mathbf{p_r} = \sum_{i=1}^{N} \mathbf{h_{Ri}} * \mathbf{g_i} \tag{13}$$

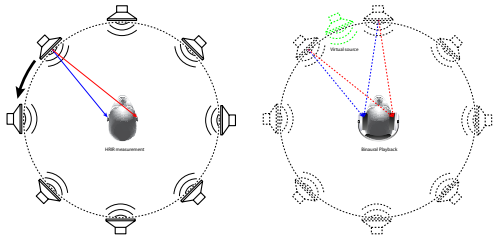

Fig. 2: The “virtual loudspeakers" reproduction concept: (a) measurement of HRIRs corresponding to the spatial locations of all the loudspeakers required by the decoder; (b) two or more loudspeaker signals are convolved with the corresponding HRIRs and summed together, forming a 2-channel binaural output

where $\mathbf{h_{Li}}$ and $\mathbf{h_{Ri}}$ HRIRs for the left and right ear respectively, corresponding to the i^{th} loudspeaker location; $\mathbf{g_i}$ is the i^{th} loudspeaker signal and $*$ denotes the convolution process[2].

In the “virtual loudspeakers" approach, HRIRs are measured at the “sweet spot" and then selected measurements are used by the decoder. Resonance Audio uses measurements provided in the open source SADIE Binaural Measurements database [18]. Decoder configurations used for Ambisonic orders 1 to 5 are: Cube, Dodecahedron, 26-point Lebedev grid, Pentakis Dodecahedron and Pentakis Icosidodecahedron respectively. The concept of forming the “virtual loudspeakers" from an array of loudspeakers is illustrated in Figure 2.

The number of convolutions necessary in the “virtual loudspeakers" method is equal to twice the number of loudspeakers used in the Ambisonic decoder. For example, for the 1^{st} order cube decoder, 16 convolutions would be used; for the 3^{rd} order Lebedev grid decoder, 52 convolutions would be used, etc. In practice, Resonance Audio uses several optimizations that significantly reduce the number of convolutions required and thus minimizes the computational load at run-time. For more information, please see Section 3.

[2]Please note that HRIR convolution in the time domain is equivalent with the HRTF multiplication in the frequency domain. We use HRIR convolution in the below examples although it is implied that these operations are performed more efficiently in the frequency domain.

2 Efficient Encoding of Sound Sources

Real-time computation of SH coefficients is CPU intensive, as can be inferred from Equations 6 and 7. Ideally, the SH coefficients would be pre-computed and stored in a Look Up Table (LUT) to minimize run-time complexity while keeping the memory footprint and pre-computation initialization time at a minimum.

Resonance Audio uses a novel method which exploits symmetries of SH functions to efficiently pre-compute, index and store and then retrieve the coefficients at run-time. This allows a reduction of the memory footprint and pre-computation speed up of $\sim 8x$.

2.1 Exploiting Symmetries of Spherical Harmonics

SH functions can be symmetric against x, y and/or z Cartesian axes. For example:

$$Y_n^m(\phi,\theta) = Y_n^m(-\phi,\theta) \tag{14}$$

indicates that $Y_n^m(\phi,\theta)$ is symmetric with respect to the sagittal plane (i.e. left-right symmetry) but at the same time is anti-symmetric with respect to the horizontal plane (e.g. up-down anti-symmetry):

$$Y_n^m(\phi,\theta) = -Y_n^m(\phi,-\theta) \tag{15}$$

An SH which exhibits the above properties is, for example, $Y_1^0(\phi,\theta)$ $(ACN = 2)$:

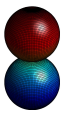

Fig. 3: Front-back symmetric / left-right symmetric / up-down anti-symmetric SH function $Y_1^0(\phi,\theta)$

It is also possible that an SH is left-right and up-down symmetric, but front-back anti-symmetric, for example:

$$Y_n^m(\phi,\theta) = -Y_n^m(\phi,-\theta) \tag{16}$$

This is the case if, and only if, n and m are odd and when $n = m$, for example $Y_3^3(\phi,\theta)$ $(ACN = 15)$:

Fig. 4: Front-back anti-symmetric / left-right anti-symmetric / up-down symmetric SH function $Y_3^3(\phi,\theta)$

In general, and without a formal proof, symmetry/anti-symmetry-related sign flip for a given SH $Y_n^m(\phi,\theta)$ with respect to the y, z, and x axis can be written as:

$$\xi_y(n,m) = \begin{cases} 1 & \text{if } m \geq 0 \\ -1 & \text{if } m < 0 \end{cases} \tag{17}$$

$$\xi_z(n,m) = (-1)^{n+m} \tag{18}$$

$$\xi_x(n,m) = \begin{cases} (-1)^m & \text{if } m \geq 0 \\ -(-1)^{|m|} & \text{if } m < 0 \end{cases} \tag{19}$$

where ξ is a symmetry-related phase coefficient for the relevant axis. Therefore, if we pre-compute and store only one sphere quadrant of the SH coefficients, we can utilize the above symmetry information to retrieve an arbitrary SH coefficient of the same order and degree. Whether to employ a given phase coefficient depends on which quadrant we decide to pre-compute as well as which quadrant the sound source is located in. For example, with the pre-computed front-left-top quadrant ($0 \leq \phi \leq \frac{\pi}{2}, 0 \leq \theta \leq \frac{\pi}{2}$) to retrieve an SH coefficient for an arbitrary angle pair in the rear-right-bottom quadrant ($-\pi \leq \phi \leq -\frac{\pi}{2}, -\frac{\pi}{2} \leq \theta \leq 0$) we can do:

$$Y_n^m(\phi,\theta) = \xi_y(n,m)\xi_z(n,m)\xi_x(n,m)Y_n^m(\pi - |\phi|, |\theta|) \tag{20}$$

Similar logic can be applied to find the SH coefficients for other sound source directions (i.e. in other sphere quadrants), by simply dropping the redundant symmetry-related phase coefficients and constraining the angle look-up to the $[0, \frac{\pi}{2}]$ range.

In Resonance Audio, `AmbisonicLookupTable` implements the above method for fast retrieval of SH coefficients. It first computes a limited size LUT of SH coefficients for angles in the $[0, \frac{\pi}{2}]$ range at 1° resolution (`ComputeEncoderTable`, as well as the symmetry information LUT `ComputeSymmetriesTable` on system start-up. Then, at run-time, the SH encoding coefficients can be rapidly retrieved by calling the `GetEncodingCoeffs` method.

Similarly, `MATLAB` users can utilize the described algorithm by using `getencodertable`, `getshsymmetries` and `ambencodecoeffsymmetric` functions respectively.

Apart from fast encoding of the sound source direction, Resonance Audio also allows to efficiently model its angular spread, which has multiple uses in sound design (for example, to simulate sound source width or create volumetric sound sources). This will be described in more detail in the next section.

2.2 Controlling Sound Source Spread

Angular spread of an Ambisonic sound source can be directly related to the metric of the spread of its acoustic energy across the unit sphere [3, 19]. This relationship can be defined as:

$$\phi_S = \arccos(\|\vec{r_E}\|) \tag{21}$$

where ϕ_S is the angular spread (width) of the source in radians and $\|\vec{r_E}\|$ is the magnitude of the energy vector pointing in the direction of that sound source. An alternative definition was given by Epain *et al.* in [20] (note conversion to radians):

$$\phi_S = 2\arccos(2||\vec{r}_E|| - 1) \tag{22}$$

Also, Frank in [21] proposed another relationship between the energy vector magnitude and perceived spread, based on subjective listening tests (also note conversion to radians):

$$\phi_S = [186.4(1 - ||\vec{r}_E||) + 10.7]\frac{\pi}{180} \tag{23}$$

For the latter, it has to be noted that it was developed for only relatively large values of $||\vec{r}_E||$ [0.75, 1]. Also, the 10.7 bias is assumed to be listening room dependent. For example, it is possible that for anechoic room

conditions this bias could approach the minimum audible angle for humans, but that would require further experimental verification.

Note that the above equations can produce vastly different results, particularly at low $||\vec{r}_E||$ values (i.e. low Ambisonic orders). By default, Resonance Audio utilizes the relationship described in Equation 21, with the exception that the curve is modified for very low values of $||\vec{r}_E||$ (below Ambisonic order 1). This way the sound source spread reaches 2π instead of $\frac{\pi}{2}$ when $||\vec{r}_E|| = 0$.

In Section 1.2.4 we have already established that *MaxRe* decoding maximizes energy concentration in the direction of a sound source, thus minimizing its spread. For example, the maximum theoretical $\|\vec{r}_E\|$ for a 3^{rd} order *basic* Ambisonic decoder is 0.75 ($\phi_S = 0.7227$ or approx. 41.4°), whereas for the same order *MaxRe* decoder $\|\vec{r}_E\| = 0.861$ ($\phi_S = 0.5336$ or approx. 30.6°).

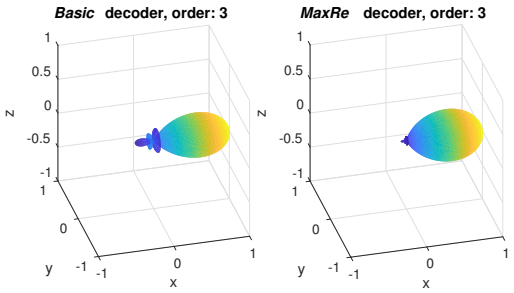

Fig. 5: Virtual microphone patterns produced using third order *basic* and *MaxRe* Ambisonic decoders

Figure 5 shows virtual microphone patterns produced using 3^{rd} order *basic* and *MaxRe* Ambisonic decoders. Although the former looks sharper, the visible side lobes, which correspond to signal components at directions other than the direction of the sound source, result in a wider overall energy spread.

Knowing the desired spread for a given sound source, we can now invert the relationship and figure out the required $\|\vec{r}_E\|$ and thus the minimum required Ambisonic order n to create such a sound source. In Resonance Audio, this relationship is approximated using the following exponential curve:

$$n = \lceil 13.15e^{-2.74\phi_S} \rceil \quad (24)$$

To sum up the discussion so far, by using *MaxRe* gain coefficients described already in Section 1.2.4 we already have a step-wise control over the sound source spread. That can be achieved by multiplying the HOA input signal vector by a lower order set of coefficients. That in turn will a) change the relative energy ratios of channels of different order; b) zero-out higher order channel contributions. This process is visualized in Figure 6.

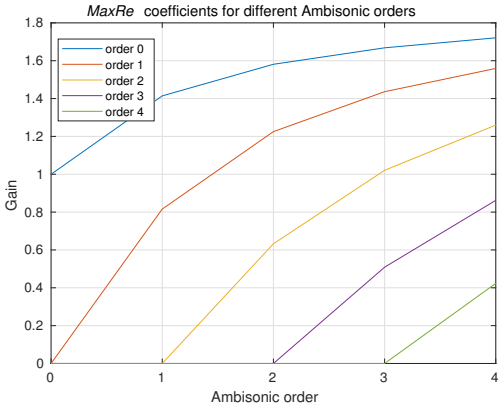

Fig. 6: *MaxRe* gain coefficients for Ambisonic decoders of orders 0 to 4 (horizontal axis). Each color represents gain values (vertical axis) that each Ambisonic channel of a given order should be scaled by in order to obtain a *MaxRe* decoder of a given order.

However, there are two challenges with the above approach. Firstly, smooth transitions between different source spreads are impossible due to Ambisonic orders defined as discrete values. Being able to transition smoothly between different source spreads (as opposed to keeping a constant angular spread) is critical, for example, to simulate a virtual sound source of a constant size in 3D space (for example, a *volumetric* source). Secondly, the total output energy of a sound source would change when changing the spread. Recall that *MaxRe* coefficients derived in Section 1.2.4 ensured energy preservation when transforming between the *basic* and *MaxRe* decoders, but *not* between decoders of different Ambisonic orders.

To deal with the first issue, we numerically find the best polynomial approximation of each *MaxRe* coefficient gain curve from figure 6. This is implemented in the `maxrecoeffs MATLAB` function and the resultant

set of smooth *MaxRe* coefficient curves is shown in Figure 7.

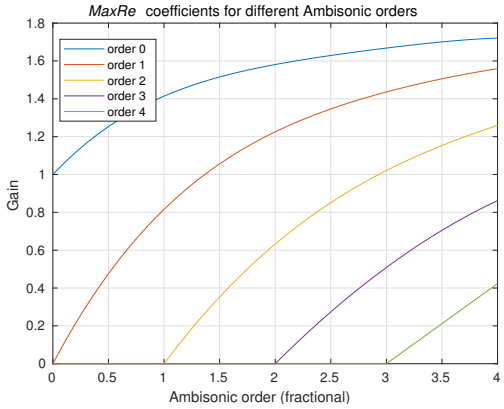

Fig. 7: *MaxRe* gain coefficients for fractional Ambisonic decoder orders ranging from 0 to 4 (horizontal axis). Each color represents gain values (vertical axis) that each Ambisonic channel of a given order should be scaled by in order to obtain a *MaxRe* decoder of a given order.

Then, we can modify the Equation 24 to return a fractional Ambisonic order which will be used by the encoder (please note the lack of the $\lceil \cdot \rceil$ ceiling operators):

$$n = 13.15e^{-2.74\phi_S} \tag{25}$$

The above formula is used in the `spread2ambiorder` MATLAB function which can be further modified if needed (for example, to allow for the relationships defined in Equations 22 or 23).

To deal with the second issue, we must ensure that when widening the sound source, we not only apply the new gain ratio to the Ambisonic channels but also the total energy of the soundfield is preserved. For example, using a maximum Ambisonic order 4 in the system, the *MaxRe* coefficients for Ambisonic order 4 would result in the narrowest source spread with a default amplitude. By reducing the Ambisonic order, we would make the sound source appear wider, but also quieter. To combat that, we must compensate the lower (fractional) order *MaxRe* coefficients so that the sound source is equally loud, no matter how wide it is. The effect of such a compensation is illustrated in Figure 8.

To compute the energy-preserving *MaxRe* coefficients, we first compute the energy of the soundfield at a given

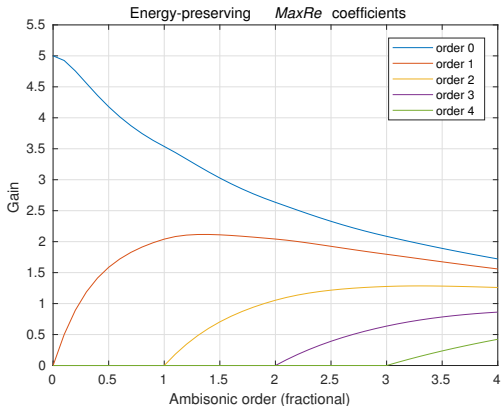

Fig. 8: Energy-preserving *MaxRe* gain coefficients for fractional Ambisonic decoder orders ranging from 0 to 4.

Ambisonic order n, which corresponds to the desired source spread, and compare it to the energy of the soundfield at the maximum Ambisonic order N. The compensation gain $\gamma_{n,N}$ is a square root of this ratio:

$$E_{n,N} = \sum_{i=0}^{N} (2i+1) \times MaxRe_{n,N}(i)^2 \tag{26}$$

$$E_{N,N} = \sum_{i=0}^{N} (2i+1) \times MaxRe_{N,N}(i)^2 \tag{27}$$

$$\gamma_{n,N} = \sqrt{\frac{E_{N,N}}{E_{n,N}}} \tag{28}$$

Finally, the energy-preserving set of *MaxRe* coefficients at an arbitrary source order n in an Ambisonic order system of order N, is simply the raw set of $MaxRe_{n,N}$ multiplied by the compensation gain $\gamma_{n,N}$.

$$\hat{MaxRe}_{n,N} = \gamma_{n,N} \times MaxRe_{n,N} \tag{29}$$

Of note, Resonance Audio applies the raw *MaxRe* correction directly to the binaural decoder filters (using shelf-filters) off-line as will be explained in Section 3.2. This allows for significant performance improvements at run-time. However, it means the energy-preserving *MaxRe* coefficients need to be further normalized so that no correction is applied if the sound source spread is set to its nominal (i.e. narrowest) value. This is

achieved by simply dividing the energy-preserving coefficients by the coefficients corresponding to the maximum Ambisonic order N:

$$\hat{\hat{MaxRe}}_{n,N} = \frac{\hat{MaxRe}_{n,N}}{MaxRe_{N,N}} \tag{30}$$

The final set of coefficients is illustrated in Figure 9.

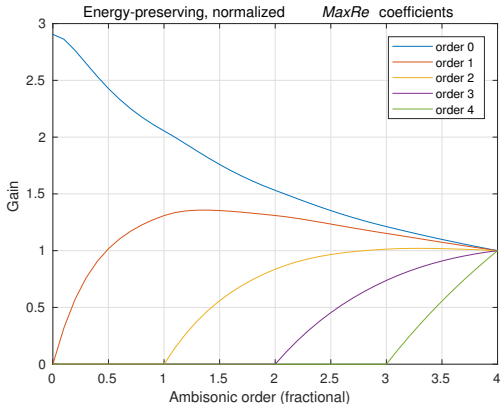

Fig. 9: Energy-preserving, normalized *MaxRe* gain coefficients for fractional Ambisonic decoder orders ranging from 0 to 4.

In Resonance Audio, `maxrespread MATLAB` function is used to generate the energy-preserving, normalized *MaxRe* gain coefficients. The final encoding coefficients are subsequenctly pre-computed and the LUT written into a C++-style header file using a helper `MATLAB` function *generatecoefftable*. The table is the accessed at run-time by the C++ `GetEncodingCoeffs` method of the `AmbisonicLookupTable` class.

The drawback of the above optimization technique is that the sound source widening can result in incorrect coefficients at very low frequencies, where the *basic* decoder is in operation. To combat that, an input soundfield should be split into frequency bands at run-time and a separate set of coefficients applied to each band, which would greatly increase the CPU consumption. After informal listening tests it was decided that this inconsistency is perceptually negligible and the result of the widening process is convincing and with minimal coloration added to the sound source. However, future work should look at evaluating these perceptual differences more formally.

2.3 Mixing in the Ambisonic domain

All sound sources within Resonance audio are mixed exclusively in the Ambisonic domain. This allows the application of all head related rotations simultaneously and ensures that all sound sources, whether point sources encoded into soundfields, or pre-recorded soundfields, are mixed before convolution with the HRTFs. As stated, performing a fixed number of convolutions, regardless of the number of sources, allows a large reduction in computational complexity.

To mix Ambisonic channels efficiently, Resonance Audio takes advantage of Single instruction multiple data (SIMD) processor instructions to add the individual samples of an Ambisonic channel to the resulting mix buffer. On Intel, the Streaming SIMD Extensions (SSE) instruction set extension; and on ARM, the NEON instruction set, allow the execution of 128-bit instructions. This means that one can add four 32-bit single precision floating point samples into a channel of the mix buffer simultaneously.

The related SIMD optimized code can be found in the `AddPointwise` function within `resonance_audio/base/simd_utils.cc`.

2.4 Near-field Sound Sources

Resonance Audio also allows for simulating near-field sound sources(for example, sound sources that appear closer than the HRTF array radius which currently is $1m$). Detailed description of the approximated near-field algorithm is going to be provided in a separate publication.

3 Efficient Binaural Decoding

By assuming symmetry of a human head, the complexity of the binaural decoding can be reduced by a factor of two compared to the approach described earlier in Section 1.3.

Let us assume that all the HRIRs are measured only in the left hemisphere and their symmetric counterparts exist in the right hemisphere simply by swapping the left and right channels. Let us also assume that the decoded input signals are arranged in a way so that the contributions to the left and (symmetrically opposite) right hemisphere loudspeakers alternate. Then, we can rewrite the binaural decoding Equations 12 and 13 as:

$$\mathbf{p_L} = 0.5 \sum_{i=1}^{N/2} [(\mathbf{h_{Li}} + \mathbf{h_{Ri}}) * (\mathbf{g_{2i-1}} + \mathbf{g_{2i}}) + (\mathbf{h_{Li}} - \mathbf{h_{Ri}}) * (\mathbf{g_{2i-1}} - \mathbf{g_{2i}})] \quad (31)$$

$$\mathbf{p_R} = 0.5 \sum_{i=1}^{N/2} [(\mathbf{h_{Li}} + \mathbf{h_{Ri}}) * (\mathbf{g_{2i-1}} + \mathbf{g_{2i}}) - (\mathbf{h_{Li}} - \mathbf{h_{Ri}}) * (\mathbf{g_{2i-1}} - \mathbf{g_{2i}})] \quad (32)$$

From the above, we can define the convolution terms as:

$$\mathbf{c_{sum,i}} = (\mathbf{h_{Li}} + \mathbf{h_{Ri}}) * (\mathbf{g_{2i-1}} + \mathbf{g_{2i}}) \quad (33)$$

$$\mathbf{c_{diff,i}} = (\mathbf{h_{Li}} - \mathbf{h_{Ri}}) * (\mathbf{g_{2i-1}} - \mathbf{g_{2i}}) \quad (34)$$

which simplify the binaural decoding equations to:

$$\mathbf{p_L} = 0.5 \sum_{i=1}^{N/2} (\mathbf{c_{sum,i}} + \mathbf{c_{diff,i}}) \quad (35)$$

$$\mathbf{p_R} = 0.5 \sum_{i=1}^{N/2} (\mathbf{c_{sum,i}} - \mathbf{c_{diff,i}}) \quad (36)$$

Since the convolution terms $\mathbf{c_{sum,i}}$ and $\mathbf{c_{diff,i}}$ in Equations 35 and 36 are identical, they are computed once. Thus, we can see that the total number of convolutions is reduced from $2N$ in Equations 12 and 13 to N.

Of note, for HRIRs in the sagittal plane, we also assume left-right symmetry and hence the virtual loudspeaker contribution can be obtained by convolving the input with only one of the HRIR channels and duplicating the output to the other channel. This way, the assumption that the maximum number of convolutions equals the number of virtual loudspeakers still holds.

3.1 Symmetric HRIR convolution in the SH domain

By representing the HRIRs using SH functions and by taking advantage of the inherent symmetries and anti-symmetries of the SHs, we can further reduce the number of convolutions and make them independent of the number of HRTFs used. By doing so, we can guarantee that the maximum number of convolutions will always equal the number of SHs used in the spatializer. For example, for the 3^{rd} order Ambisonic binaural decoder using the 26-point Lebedev grid, that means reduction from 26 to 16 convolutions ($\sim 62\%$).

Also, the HRTF positions in principal no longer need to conform to any symmetry. That said, the performance of the decoder is still expected to be better for regular layouts (i.e. when the loudspeaker positions are at the vertices of a platonic solid). In addition, the generation of the SH HRIRs is greatly simplified if that is the case.

In order to encode the HRIRs into SH representation, we first compute the decoding matrix $\mathbf{D}$ as in equation 10, where the ordering of HRTF angles follows the left-right symmetry rule: if the first HRTF is at $+45^\circ$ azimuth angle and -35.26° elevation angle the next one should be at -45° azimuth angle and -35.26°, etc. The HRTFs at the sagittal plane can be added at an arbitrary order.

Next, we compose a HRIR signal matrix $\mathbf{H}$, by concatenating HRIRs in the same order that was used to compute the decoding matrix $\mathbf{D}$. Following our example above, the first two columns of the matrix would be channels left and right of the HRIR corresponding to the $+45^\circ$ azimuth angle and -35.26°. The symmetry assumption means that the same HRIR is used to represent virtual loudspeakers at angles $+45^\circ$ azimuth / -35.26° elevation and -45° azimuth / -35.26° elevation.

Finally, our SH HRIR decomposition $\hat{\mathbf{H}}$ is computed as:

$$\hat{\mathbf{H}} = \mathbf{H}\mathbf{D}^\mathbf{T} \quad (37)$$

and the binaural decoding equations for an arbitrary Ambisonic order simplify to:

$$\mathbf{p_L} = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} \mathbf{b_n^m} * \hat{\mathbf{h}}_\mathbf{n}^\mathbf{m} \quad (38)$$

$$\mathbf{p_R} = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} \begin{cases} \mathbf{b_n^m} * \hat{\mathbf{h}}_\mathbf{n}^\mathbf{m} & \text{if } m \geq 0 \\ -\mathbf{b_n^m} * \hat{\mathbf{h}}_\mathbf{n}^\mathbf{m} & \text{if } m < 0 \end{cases} \tag{39}$$

In Equation 39 the first sum over n,m corresponds to symmetric terms with respect to the forward axis, while the second sum over n,m corresponds to anti-symmetric terms with respect to the forward axis. That is, the symmetric terms maintain their sign upon reflection about the forward axis, while the anti-symmetric terms change their sign upon reflection about the forward axis. In practical terms, it means that the convolution is performed only once per Ambisonic channel, and when summing the channels to create the right ear output, channels corresponding to anti-symmetric SHs (i.e. $m < 0$) have their phase inverted. This is illustrated in figure 10 for the 1^{st} order Ambisonic case.

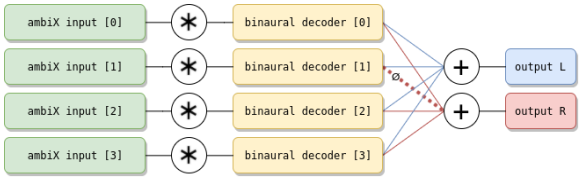

Fig. 10: Optimized binaural decoding of Ambisonic soundfields

3.2 Pre-filtering of SH HRIRs

The improved binaural decoding technique described above also allows for zero run-time cost *MaxRe* soundfield optimization (see Section 1.2.4). *MaxRe* SH weights are typically applied using shelf-filters at the soundfield decoding stage, before the loudspeaker signals are obtained. However, when using binaural decoding via SH HRIRs the decoding stage is performed off-line when creating the binaural decoding filters. Hence, the shelf-filtering can also be preformed off-line once the HRIRs are represented in the SH domain. Removing both decoding matrix and shelf-filtering from the real-time processing pipeline results in significant CPU savings and allows for efficient binaural decoding of HOA even on low-end or mobile devices.

4 Summary

This paper described the core algorithms used by the Resonance Audio's spatializer and, at the time of writing, may be used as a reference to the source code contained within the Resonance Audio C++ and `MATLAB` libraries. Note however, that future algorithms within the Resonance Audio might change, as new methods and DSP algorithms are constantly being developed. It is intended to record such changes as separate documents referencing this paper. Also, note that not all the algorithms used by Resonance Audio have been described here, but only the ones directly pertaining to the Resonance Audio's spatializer. For example, room effects algorithms (early reflections and late reverberation) are mostly documented in the code itself. However, some critical parts of the late reverberation algorithm are published in [22]. Also, the near-field effect and directivity control algorithms are intended to be documented separately. Future work should focus on, for example: practical ways of individualization of SH HRIRs in the Resonance Audio binaural decoder, adding efficient decoding for regular and irregular, physical loudspeaker setups, simulation of advanced sound source radiation patterns and API methods for creating volumetric sound sources, as well as further quality (for example, better modelling of spread functions at low frequencies) and efficiency improvements to existing algorithms.

5 Acknowledgments

The authors gratefully acknowledge Jan Skoglund, Damien Kelly, Dillon Cower, Wayne Jackson and Brian O'Toole, for their input into the project and help in preparation of this manuscript. We would also like to thank the reviewers for their insightful comments and suggestions.

References

[1] Gerzon, M. A., "Periphony: With-height sound reproduction," *Journal of the Audio Engineering Society*, 21, pp. 2–10, 1973.

[2] Malham, D., "Higher order Ambisonic systems for the spatialisation of sound," in *Proceedings of the International Computer Music Conference'99*, Beijing, China, 1999.

[3] Daniel, J., *Acoustic field representation, application to the transmission and the reproduction of complex sound environments in a multimedia context (English translation)*, Ph.D. Thesis, University of Paris, Paris, France, 2001.

[4] Nachbar, C., Zotter, F., Deleflie, E., and Sontacchi, A., "AMBIX - A Suggested Ambisonics Format," in *Proceedings of the 3rd Ambisonics Symposium*, Lexington, KY, USA, 2011.

[5] Kronlachner, M., *Spatial Transformations for the Alteration of Ambisonic Recordings*, Master's thesis, University of Music and Performing Arts, Graz, Austria, 2014.

[6] Ivanic, J. and Ruedenberg, K., "Rotation Matrices for Real Spherical Harmonics. Direct Determination by Recursion," *The Journal of Physical Chemistry*, 100(15), pp. 6342–6347, 1996, doi: 10.1021/jp953350u.

[7] Ivanic, J. and Ruedenberg, K., "Additions and Corrections: Rotation Matrices for Real Spherical Harmonics. Direct Determination by Recursion," *The Journal of Physical Chemistry*, 102(45), pp. 9099–9100, 1998, doi:10.1021/jp953350u.

[8] Green, R., "Spherical harmonic lighting: The gritty details," *Archives of the Game Developers Conference*, 2003.

[9] Daniel, J., Rault, J.-B., and Polack, J.-D., "Ambisonics Encoding of Other Audio Formats for Multiple Listening Conditions," in *Proceedings of the 105th Audio Engineering Society Convention*, San Francisco, CA, USA, 1998.

[10] Gerzon, M. A., "General Metatheory for Auditory Localisation," in *Proceedings of the 92nd Audio Engineering Society Convention*, Vienna, Austria, 1992.

[11] Gerzon, M. A., "Panpot Laws for Multispeaker Stereo," in *Proceedings of the 92nd Audio Engineering Society Convention*, Vienna, Austria, 1992.

[12] Lee, R., "Shelf Filters for Ambisonic Decoders," Technical report, http://ambisonic.info/info/ricardo/shelfs.html, 2008.

[13] Heller, A., Lee, R., and Benjamin, E., "Is My Decoder Ambisonic? (Revision 2)," in *Proceedings of the 125th Audio Engineering Society Convention*, San Francisco, CA, USA, 2008.

[14] Benjamin, E., Heller, A., and Lee, R., "Design of Ambisonic Decoders for Irregular Arrays of Loudspeakers by Non-Linear Optimization," in *Proceedings of the 129th Audio Engineering Society Convention*, San Francisco, CA, USA, 2010.

[15] McKeag, A. and McGrath, D., "Sound Field Format to Binaural Decoder with Head-Tracking," in *Proceedings of the AES 6th Australian Regional Convention*, Melbourne, Australia, 1996.

[16] Noisternig, M., Sontacchi, A., Musil, T., and Höldrich, R., "A 3D Ambisonic Based Binaural Sound Reproduction System," in *Proceedings of the Audio Engineering Society 24th International Conference on Multichannel Audio*, Banff, Alberta, Canada, 2003.

[17] Dalenbäck, B. and Strömberg, M., "Real time walkthrough auralization - the first year," *Proceedings of the Institute of Acoustics*, 28(2), 2006.

[18] Kearney, G. and Doyle, T., "An HRTF Database for Virtual Loudspeaker Rendering," in *Proceedings of the 139th Audio Engineering Society Convention*, 2015.

[19] Bertet, S., Daniel, J., Parizet, E., and Warusfel, O., "Investigation on Localisation Accuracy for First and Higher Order Ambisonics Reproduced Sound Sources," *Acta Acustica united with Acustica, Hirzel Verlag*, 99, pp. 642–657, 2013.

[20] Epain, N., Jin, C. T., and Zotter, F., "Ambisonic Decoding With Constant Angular Spread," *Acta Acustica United with Acustica*, 100, pp. 928–936, 2014.

[21] Frank, M., "Source Width of Frontal Phantom Sources: Perception, Measurement, and Modeling," *Archives of Acoustics*, 38(3), pp. 311–319, 2013.

[22] Kelly, I. J., Gorzel, M., and Güngörmüsler, A., "Efficient externalized audio reverberation with smooth transitioning," in *Technical Disclosure Commons*, 2017.