UNIVERSITY OF CALIFORNIA, SAN DIEGO

PhD Qualifying Examination in Computer Music

A qualifying exam submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Music

by

Gabriel Zalles Ballivian

Committee in charge:

Shahrokh Yadegari, Chair
Professor Erbe, Tom
Professor Smyth, Tamara
Professor Roberts, Justin
Professor Steiger, Rand

2021

The qualifying exam of Gabriel Zalles Ballivian is approved, and it is acceptable in quality and form for publication on microfilm:

_____

_____

_____

_____

_____
Chair

University of California, San Diego

2021

TABLE OF CONTENTS

LIST OF FIGURES

QUESTIONS FROM COMMITTEE

PhD Qualifying Examination in Computer Music

by

Gabriel Zalles Ballivian

Doctor of Philosophy in Music

University of California San Diego, 2021

Shahrokh Yadegari, Chair

Question one (Tom Erbe): In our meetings we have talked about the need for an accessible spatial audio toolkit for composition, performance, sound design and instruction. Discuss what is needed for such a toolkit, and what makes this toolkit. Compare at least 3 tool-kits, with an analysis of function, technical framework, and personal assessment of sound quality. Add to this a description of your use of one of these spatial audio tool-kits.

Question two (Tamara Smyth): Toward the aim of making spatial audio and ambisonics more accessible, i.e. more affordable, comprehensible, usable and available to the average user, discuss the design and development considerations, along with possible trade-offs, of low-cost sound-field microphone designs and accompanying (open-source) processing software for signal encoding/decoding/formatting. How have your own experiments and implementation of an existing design informed your future work and goals? What are your impressions of the viability of this tool for musicians wanting to incorporate spatialized audio into their work?

Question three (Shahrokh Yadegari): Your various research projects during the past years have revolved around capture, production, development, and presen-

tation of 3D audio and music. A recent important topic has been providing access to communities with less resources to produce and consume multi-channel music and audio (usually in ambisonic format). Present your research on the history of distribution of surround and 3D audio and music in general, and examine and compare contemporary technological infrastructures which may serve your aspirations of wide distribution of such music without the extensive equipment requirement on the consumer side. Pay special attention to the quality of the algorithms of such approaches. Finally present your plans for the use of some of these approaches in your own work or the work of other artists whose works you have produced or presented.

Oral Examination

Friday, June 11, 2021

Time: 8:30 AM

Zoom Meeting ID: 2024413074

Chapter 1

Free and Open Source Spatial Audio Tool-kits

Question: *In our meetings we have talked about the need for an accessible spatial audio toolkit for composition, performance, sound design and instruction. Discuss what is needed for such a toolkit, and what makes this toolkit. Compare at least 3 tool-kits, with an analysis of function, technical framework, and personal assessment of sound quality. Add to this a description of your use of one of these spatial audio tool-kits.*

1.1 Introduction

Over the last century a number of composers have explored the use of space as a key parameter of musical composition. Recently, with the growth of *Extended Reality* (XR), an even greater number of artists have began exploring this medium, as a way to push this musical domain to new extremes. XR environments, such as *Virtual Reality* (VR) or *Augmented Reality* (AR), constitute a new field rich with creative possibilities towards art-making. In this chapter, we explore the latest technological

developments vis-à-vis tools that can be used to create spatial music.

In our research we found that state-of-the-art development in this domain has shifted towards *Virtual Studio Technologies* (VSTs). Two institutions in particular, Aalto and IEM, have been developing *Free and Open Source Software* (FOSS) solutions, which are intended for *Digital Audio Workstation* (DAWs) like Ardour or Reaper[1]. Other computer music languages, such as MAX/MSP[2] also support VSTs, and offer support for multi-channel music, making them ideal for incorporating these tools into real-time compositional performances.

While there are also useful tool-kits in the commercial sector, in this chapter we will focus on FOSS which engineers may adapt, and musicians with limited resources may freely acquire. There are also frameworks[3] for the creation of new spatial audio VSTs, which will be omitted from the conversation; our goal is to take a deep look into already compiled tool-kits composers may implement in their future works. We hope this deep dive into the mechanics of these effects will inspire sound engineers to develop their own plug-ins, as well as inform composers about the internal nature of these applications.

1.1.1 Requirements

What is needed for a toolkit to be accessible for composition, performance, sound design and instruction?

1. Cross-platform e.g. runs on all major *Operating Systems* (OSs).

2. Free, so composers and students with limited resources can use them.

3. Open source, so students with technical backgrounds can modify them.

[1]These particular DAWs have multi-track support and are free or low-cost.
[2]By Cycling 74.
[3]Programming libraries such as the Spatial Audio Framework by Aalto.

4. Well documented, so the user can easily set-up the system and start composing.
5. Well supported (i.e. regularly re-compiled as new versions of OSs emerge).
6. Intuitive e.g. relatively easy to understand and use without formal technical training.

1.1.2 Composition

What is an accessible spatial audio tool-kit made up of?

As we will see, in this chapter the focus will be on *ambisonics*, which we believe constitutes the most accessible and standardized method to create spatial music today. *Object-Based Audio* (OBA), another popular technology, is generally created using proprietary software, which features closed-source code, and is often very expensive. OBA is often used in big-budget cinematic productions. In this method, the audio source is encoded with associated meta-data, describing the spatial properties of the sound object, which a decoder applies at the reproduction side. *Vector Based Amplitude Panning* (VBAP) is a common method used in OBA for spatializing sound sources. VBAP is an extension of traditional stereo panning algorithms, which calculates gains in three dimensional space for a given speaker layout. One can also use VBAP without the need for an OBA encoding process.

In contrast to many other spatial audio methods (i.e. VBAP, *Wave Field Synthesis* (WFS), or Surround Sound), ambisonic tool-kits generally feature binaural synthesis methods which make it possible to compose spatial music without loudspeaker arrays. This, we believe, is one of the key features of ambisonics which make it very attractive financially compared to other techniques. A sensor, such as a gyroscope, can be used to send data to the computer, which determines the user's head orientation. A linear rotational transform is then applied to the ambisonic audio, which is subsequently decoded binaurally using a set of *Head Related Transfer*

Functions (HRTFs)[4].

This means that using very cheap external electronic components, the user can simulate a speaker array at home via a pair of headphones. The resulting sound-field, however, is not limited to binaural reproduction. It can be decoded over *High Density Loudspeaker Arrays* (HDLAs), if one has access to one, allowing larger audiences to listen to these works together. Alternatively, these files can be decoded binaurally online, using a variety of tools discussed in Chapter 3.

1.2 FOSS Spatial Audio Tool-kits

The following three tool-kits we will describe correspond to those we believe are the most accessible in terms of price and ease-of-use. In addition, these plug-ins have extensive documentation and publications, which provide a deeper insight into the underlying mechanisms of these systems.

One other notable spatial audio system popular among composers is Spat, which is designed to be used in MAX/MSP (MAX). MAX, unfortunately, is not FOSS; which makes it less accessible in the financial sense than these tools. Puredata (Pd), the FOSS version of MAX, does offer some great spatial audio tools. Unfortunately, we found many of these are difficult to operate for beginners, or don't have enough documentation - making them difficult for composers to use.

In contrast, all these DAW-based tools were seamless to install and offer substantial documentation which both composers and engineers can glean a lot of information from. The drawback of these systems is that interactive musical works are not as easily accomplished, due to the linear nature of DAWs. These tools are better suited for fixed-media works, such as movies, or open-loop XR experiences[5].

[4]Filters which simulate the effect of the human head and body on the sound source.

[5]Open and closed loop are XR terms, referring to non-interactive and interactive experiences.

ambiX

AmbiX is a cross-platform plug-in suite developed by Matthias Kronlachner [Kro14b], which can be used in conjunction with: Reaper, Ardour, MAX, or as standalone application. The suite uses the ambix convention [NZDS11], which features ACN[6] ordering and SN3D[7] normalization. Kronlachner also provides a mcfx suite (multi-channel effects suite) which includes the:

1. **mcfx_convolver**: multi-channel convolution which can be used to create ambisonic reverberation effects or encode microphone arrays to the ambisonic domain.
2. **mcfx_delay**: which delays each channel the same time.
3. **mcfx_filter**: which filters each channel with the same filter and displays a frequency analyzer of the sum of all channels.
4. **mcfx_gain_delay**: which sets different delay times for each channel and is used mainly for speaker calibration.
5. **mcfx_meter**: which is a multi-channel level meter.

The **mcfx** toolbox, while useful, does not offer much creative potential. In addition, there is nothing very interesting from an implementation perspective to discuss. Much like other ambisonic tool-kits, Kronlachner also provides: binaural decoding, multi-channel decoding, format converting[8], ambisonic encoding, and ambisonic rotation plug-ins[9].

[6] Ambisonic Channel Number.

[7] Schidmt Semi-Normalized Spherical Harmonics. Chapter 2 discusses ordering and normalization in greater detail.

[8] To switch between standards. For example converting from N3D normalization to SN3D, or from FuMa ordering to ACN.

[9] Details regarding these transforms are covered in Chapters 3 and 2.

Rather than discussing these common functions, we would like to discuss what is distinct about this plug-in suite. Kronlachner's paper [KZ14] describes some of the more unique spatial transforms implemented in **ambiX**. These include:

1. **ambix_directional_loudness**: which allows the user to "amplify, attenuate or filter out certain parts of the sound-field".
2. **ambix_maxre**: applies *MaxRe* weighting to an ambisonic signal, which suppresses side-lobes when panning. Typically used for decoding.
3. **ambix_mirror**: simple algorithm which allows mirroring about the x, y, or z axis. Exploits spherical harmonic symmetry.
4. **ambix_vmic**: generates a *virtual microphone* from a linear combination of spherical harmonics. Similar in principle to directional loudness but outputs a mono signal extracted from the sound-field.
5. **ambix_warp**: warps a sound-field towards the equator, poles, front, or back.
6. **ambix_widening**: frequency dependant rotation around the z-axis, use this for source widening or creating diffuse early reflections [ZFKC14].

MaxRe, mirroring, and virtual microphones - a technique also known as *beam-forming* - will be covered in more detail in Chapter 3. In this section, instead, we will cover *directional loudness*, *warping* and *widening* since these three effects are mostly unique to **ambiX**. All three libraries described in this chapter can be used together, since they use the same standards.

Directional Loudness The goal of direction loudness is to amplify a particular region of the sound-field. In contrast to beam-forming, the output of this effect is multi-channel (e.g. we maintain the same number of spherical harmonics).

In Kronlachner's paper [KZ14] a *t-design* approach is used for spherical sampling; t-designs are attributed to Hardin and Sloane [HS96].

In order to accurately sample an ambisonic sound-field of a given order N, the number of sampling directions must be equal to or greater than the number of harmonics given by $(N+1)^2$. Furthermore, the condition number of $\mathbf{Y}$, the matrix of spherical harmonics, used in this case to represent our sampling directions, needs to be sufficiently small[10].

The coordinates for these t-designs are ideal for sampling the spherical domain because they allow us to perform *Discrete Spherical Harmonic Transforms* (DSHT) without *pseudo-inversion*[11].

According to Hardin and Sloane:

> "A set of N points is called a spherical t-design if the integral of any polynomial of degree at most t over the sphere is equal to the average value of the polynomial over the set of N points. [Har]"

The spherical t-designs also have identical weighting coefficients at all nodes, in contrast to other sampling approaches (e.g. t-designs are uniform). Spherical t-designs vary in size - the maximum size Sloane provides is a 21-design with 240 points, which allows ambisonics up to order $N = 10$.

All of Kronlachner transforms are based on this spherical t-design. The spherical domain is sampled according to these coordinates, this generates a series of nodes to which rotational and magnitude changes can be applied[12] based on the desired effect.

Kronlachner describes the DSHT sans pseudo-inversion mathematically as:

[10]The condition number of a function measures how much the output value of the function can change for a small change in the input argument. A problem with low condition number is said to be well-conditioned. For an ambisonic problem, this means that small changes in coordinates should not substantially change the spherical domain sampling.

[11]They can be found here.

[12]Kronlachner notes that angle-distorting and directional-loudness require the t-design sampling to be of higher order than the ambisonic signal.

$$\boldsymbol{T} = \text{diag}\{\boldsymbol{b}\}\boldsymbol{Y}^{\text{T}}\left(\boldsymbol{\Theta}_t\right)\text{diag}\left\{\boldsymbol{g}\left(\boldsymbol{\Theta}_t\right)\right\}\boldsymbol{Y}\left(\mathcal{T}^{-1}\left\{\boldsymbol{\Theta}_t\right\}\right) \tag{1.1}$$

T corresponds to the square transformation matrix we seek to obtain, which will subsequently be multiplied by our matrix of *spherical harmonics* (SH) to achieve the desired transformation. Our SH signals can either derived by encoding a mono signal, or by encoding the signals of a spherical microphone array into the ambisonic domain.

$$\boldsymbol{b} = \left[\frac{2n_{ACN}+1}{\text{L}}\right]_{ACN} \tag{1.2}$$

$\boldsymbol{b}$ is a vector where n is the ambisonic order of the spherical harmonic, based on ACN format, and the values are ordered in ACN format[13]. L is the number of sampling directions in the t-design. Once calculated, the vector is diagonalized into a matrix[14]. $\boldsymbol{Y}^{\text{T}}\left(\boldsymbol{\Theta}_t\right)$ corresponds to the spherical harmonic sampling using our traditional real-valued ambisonic equation[15]:

$$Y_n^m(\varphi,\vartheta) = N_n^{|m|}P_n^{|m|}(\sin(\vartheta))\begin{cases}\sin|m|\varphi, & \text{for } m<0\\ \cos|m|\varphi, & \text{for } m\geq 0\end{cases}$$

ordered and normalized according to the ambix standard. Since it is transposed, each row of the matrix correspond to a single harmonic, and each column corresponds to a direction. $\boldsymbol{\Theta}_t$ is a L by 3 matrix corresponding to the coordinates of our t-design in Cartesian coordinates[16], which are converted to spherical coordinates, represented by ϕ and θ, to derive our SH matrix. Cartesian to spherical coordinate conversion is given by:

[13] m is the ambisonic degree, of the SH.

[14] In Kronlachner's thesis [Kro14b] a scalar value of $4\pi/L$ is diagonalized instead. See Equation 3.10 in the thesis.

[15] More information about this Equation can be found in Chapter 2, Section 2.1.1.

[16] In Cartesian coordinates these vectors are triplets with coordinates (x, y, z).

$$\varphi = \arctan \frac{\theta_y}{\theta_x}, \quad \vartheta = \arctan \frac{\theta_z}{\sqrt{\theta_x^2 + \theta_y^2}}$$

where ϕ corresponds to the azimuthal angle and θ to the elevation angle.

Using these coordinates, represented by the matrix $\boldsymbol{\Theta}$:

$$\boldsymbol{\Theta} = [\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{\mathrm{L}}]^T$$

we compute the coefficients corresponding to our spherical harmonic sampling.

The resulting L by $(N+1)^2$ matrix is transposed and multiplied with the *weighting function* $\boldsymbol{g}(\boldsymbol{\Theta}_t)$, which is a vector of amplitudes calculated to weight each node; there will be one weighting value for each sampling point in the t-design.

For direction loudness modifications, we use a neutral direction mapping (e.g. $\mathcal{T}\{\boldsymbol{\theta}\} = \boldsymbol{\theta}$), which means for this operation, it can simply be ignored. Therefore, the second $\boldsymbol{Y}$ term in our equation will be the same as the first, simply transposed. Due to the properties of this sampling method, their direct multiplication yields the identity matrix. The weighting function for directional loudness modification is given by:

$$g(\boldsymbol{\theta}) = u\left(\boldsymbol{\theta}_{\mathrm{c}}^{\mathrm{T}}\boldsymbol{\theta} - \cos\frac{\gamma_{\mathrm{c}}}{2}\right) \tag{1.3}$$

Where $u(\cdot)$ is the unit step function[17] and γ_{c} corresponds to the size of our region, which can be changed by the user[18]. For each direction in $\boldsymbol{\Theta}_t$ we take the inner product between the coordinate of the center of our directional loudness region and the coordinate of the sampling point in the t-design, to crop out part of the surround sound scene. If the term inside the unit step function is positive, we multiply the node by one, otherwise, we mute that node. Adding this cropped region to the original will result in a boost in the given direction. We can also multiply by negative one to subtract from the original sound-field.

[17] Which is equal to 1 for all inputs greater than or equal to 0.

[18] This is a *modulateable* parameter.

$$g(\boldsymbol{\theta}) = g_1 u\left(\boldsymbol{\theta}_{\mathrm{c}}^{\mathrm{T}}\boldsymbol{\theta} - \cos\frac{\gamma_{\mathrm{c}}}{2}\right) + g_2 u\left(\cos\frac{\gamma_{\mathrm{c}}}{2} - \boldsymbol{\theta}_{\mathrm{c}}^{\mathrm{T}}\boldsymbol{\theta}\right) \tag{1.4}$$

Equation 1.4 shows a modified function that allows one to determine the gains for both the areas within and outside our cropped region. g_1 corresponds to the gain for the inner region, and g_2 corresponds to the gain for the outer region. This equation has not yet been implemented, as far as we could tell. Another use case of this operation is applying different sound effects to an inner and outer region of the sound-field, whose dimensions are proportional and could be controlled by the user[19]. In HOAST [DMKH+20], one of the key projects discussed in Chapter 3, this directional loudness transform is used together with the warping function[20] to achieve a sonic zooming effect, which synchronizes with the video zoom of the 360° player.

Warping Warping combines a weighting function and angle changes into a single operation. Warping is used to stretch a certain region of the sound-field while squeezing other regions to prevent overlap [KZ14]. Kronlachner describes the warping operation for elevation angle ϑ to $\tilde{\vartheta}$. Warping in other directions can be done by pre- and post-rotations[21]. To preserve the total loudness of sounds within the stretched and squeezed parts of the sound-field, a gain weight is applied as part of warping. This can either be done as pre- or post-emphasis; regions where energy might overpower speaker responses will need to be de-emphasized.

Kronlachner proposes two types of warpings:

1. **Warping to elevate or lower the equator**: which moves all the energy towards the north pole, or the south pole, depending on the value of α.

[19] This would likely destroy the image, but for artistic purposes, that is fine.
[20] Type 2 warping, towards or away from equator.
[21] As discussed in Chapter 3, Section 3.5.2.

2. **Warping towards or away from equator**: which moves all the energy towards both poles simultaneously, or away from both poles simultaneously, depending on the value of β.

Elevating or Lowering the Equator Our Cartesian unit direction vector is composed of three variable for each axis in 3D space.

$$\boldsymbol{\theta} = \begin{pmatrix} \theta_x \\ \theta_y \\ \theta_z \end{pmatrix} = \begin{pmatrix} \cos\varphi\cos\vartheta \\ \sin\varphi\cos\vartheta \\ \sin\vartheta \end{pmatrix}$$

We use the substitution:

$$\mu = \sin(\vartheta), \quad \text{original},$$
$$\tilde{\mu} = \sin(\tilde{\vartheta}), \quad \text{warped},$$

to simplify our equations, and consider only *monotonically*[22] rising warping curves that map μ with interval $[-1, 1]$ to $\tilde{\mu}$ - with the same interval. In order to perform warping towards or away from the equator we use a *bilinear transform* of the form:

$$\tilde{\mu} = \frac{\mu + \alpha}{1 + \alpha\mu}$$

Our spherical angle ϑ is a function of all three coordinates. The simplest way to implement this algorithm is to first isolate θ_x, θ_y, and θ_z in equation 1.5, which we recall is the Cartesian to spherical coordinate conversion showed earlier. Note, α is user controlled parameter which should remain between -1 and 1, it is used to control the warping direction.

$$\vartheta = \arctan\frac{\theta_z}{\sqrt{\theta_x^2 + \theta_y^2}} \tag{1.5}$$

[22]Monotonic functions are functions that either entirely non-increasing or non-decreasing. Their first derivative, therefore, never changes sign.

so,

$$\theta_z = \tan(\vartheta)\left[\sqrt{\theta_x^2 + \theta_y^2}\,\right] \tag{1.6}$$

and

$$\theta_x^2 = \left(\frac{\theta_z}{\tan(\vartheta)}\right)^2 - \theta_y^2 \tag{1.7}$$

Solving for θ_x^2 is the same as θ_y^2 except the two terms are exchanged. With these three variables isolated we can replace ϑ for $\tilde{\vartheta}$, which equals $\arcsin\left(\frac{\mu+\alpha}{1+\alpha\mu}\right)$, according to our substitution. The three resulting values can then be plugged back into 1.5 to calculate our warped elevation angle. Note that θ_x^2 is intentionally left as a squared coordinate, since this is how it appears in Equation 1.5.

These new elevations angles are then used to calculate our spherical harmonic coefficients to determine the rotation of our nodes. The final step is to calculate the gain weighting which compensates for any added energy. The gain values for this type of warping are calculated using Equation 1.8. Note that these gain values need to be calculated after warping, so the new coordinates found should be used.

$$g(\mu) = \frac{\sqrt{1-\alpha^2}}{1+\alpha\mu} \tag{1.8}$$

which simplifies to,

$$g(\sin(\vartheta)) = g\left(\sin(\underbrace{\tan^{-1}\left(\frac{\theta_z}{\sqrt{\theta_x^2+\theta_y^2}}\right)}_{\vartheta})\right) \tag{1.9}$$

Warping Towards or Away from Equator This second type of warping is quite similar to the first in terms of its implementation, the only difference is that we are no longer using the bilinear transform. This type of warping pushes all the

energy towards the top of the sphere or the bottom of the sphere. Much like before, if we wanted to push energy in other directions, all we would need is a rotation before and after the warping. This warping also requires a gain compensation which has a different formula than Equation 1.8.

Equations 1.10 and 1.11 describe the angle and gain modifications respectively [KZ14].

$$\tilde{\mu} = \begin{cases} \frac{(|\beta|-1)+\sqrt{(|\beta|-1)^2+4|\beta|\mu^2}}{2|\beta|\mu}, & \text{for } \beta > 0 \\ \frac{(1-|\beta|)\mu}{1-|\beta|\mu^2}, & \text{for } \beta < 0 \end{cases} \tag{1.10}$$

being the angle transformation required, and,

$$g(\mu) = \left(\frac{1-|\beta|\mu^2}{\sqrt{(1-|\beta|)\,(1+|\beta|\mu^2)}} \right)^{\text{sgn}\{\beta\}} \tag{1.11}$$

corresponding to the gain compensation, also calculated after angle warping. The $\text{sgn}\{\beta\}$ is an operator that returns the sign of β, which is controlled by the user and should remain between -1 and 1.

Kronlachner's thesis also describes a hyperbolic paraboloid warping that does not seem to have been yet implemented. Section 3.5 of the thesis [Kro14b] explains that alternate mappings can be used for interesting musical results. The two implemented transforms were prioritized due to their usefulness. The hyperbolic paraboloid warping was inspired by the architectural design of the Philips Pavilion, by Le Corbusier, where Edgard Varèse performed his *Poème Électronique*, a seminal work of electro-acoustic spatial music, in 1958 at the World Expo[23].

Metering While not part of the same library of objects[24], we should also talk about the development of Kronlachner's metering plug-in [KZ14]. This plug-in was

[23] Iannis Xenakis, another major composer in this field, was also involved in the design of this building. Xenakis was an architect as well as a composer.

[24] Technically the metering plug-in is part of `mcfx`, not `ambiX`.

used to confirm that warping operations had been properly implemented, which can be difficult to determine using only our ears. The plug-in calculates *Root Mean Square* (RMS) and *peak values* for each sampled point in the t-design and provides adjustable *release time*. In order to determine the power at each node, the ambisonic signals are virtually decoded.

Logarithmic values of both measures are used, a common practice in audio systems design. The v by h[25] texture is used to display the directional RMS values. The color is selected based on the loudness of the signal at the point in the spherical grid. A Mollweide projection is also used to map the surround image to a 2D image. Figure 1.1 shows the 3D representation of the metering plug-in on the right and the Mollweide projection on the left.

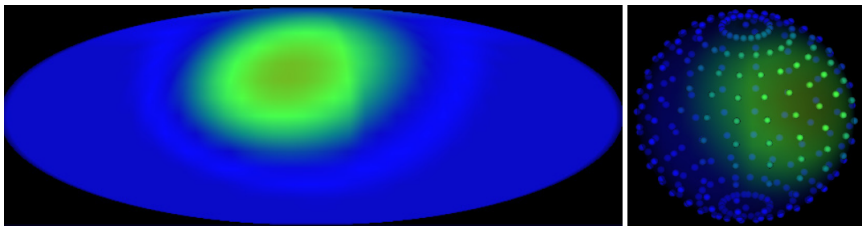


Figure 1.1: Ambix Meter: Mollweide Projection (left), 3D representation (right) [KZ14]

Unfortunately, it appears that this visualization was only prototyped in Pd, using Gem, but never compiled as a VST. Some of the other suites we will talk about in later sections do provide VSTs that use a similar mechanism. This section was included to help understand the design process of a simple visualization system for ambisonics.

Widening Kronlachner's widening algorithm is described in [ZFKC14] as a "frequency-dependent rotation". The system consists of 2-by-2 filter matrices that rotate complimentary harmonics of identical ambisonic order n and ambisonic degree $\pm m$. In

[25] Vertical v by horizontal h.

ACN ordering, for FOA, harmonics Y and X[26] are rotated as a group using these filter matrices, for example. This process is then extended to higher orders.

The filter matrices consist of FIR filters derived using a Fourier domain response represented by a phase modulated cosine[27]. E.g.

$$H(\alpha, \beta, \Omega) = \cos[\alpha \cos(\Omega) + \beta] \tag{1.12}$$

where:

α can be called the modulator amplitude,

β can be called the initial phase of the modulator, and,

Ω is the normalized frequency ($\Omega = \omega T$)[28].

The inverse Fourier transform of this response can be defined as the product of a Bessel function with a cosine in the time domain. Bessel functions are defined in the literature as cylindrical functions used when finding solutions to the wave equation in cylindrical or spherical coordinates. Since these also manifest themselves in certain application discussed in Chapters 2 and 3, the derivations are presented in Appendix A. One thing that stands out from this algorithm is that all harmonic of $m = 0$ are ignored, because their rotation about the z-axis has no effect.

The equation involved in deriving the four filters in our 2-by-2 matrix is given by:

$$\boldsymbol{R}^{m}_{\hat{\phi},\varphi_0,\mathrm{Q}}[\lambda\mathrm{Q}] = \begin{bmatrix} h\left(m\hat{\phi}, m\varphi_0, (\lambda-\Lambda)\mathrm{Q}\right] & h\left(m\hat{\phi}, m\varphi_0 + \frac{\pi}{2}, (\lambda-\Lambda)\mathrm{Q}\right] \\ h\left(m\hat{\phi}, m\varphi_0 - \frac{\pi}{2}, (\lambda-\Lambda)\mathrm{Q}\right] & h\left(m\hat{\phi}, m\varphi_0, (\lambda-\Lambda)\mathrm{Q}\right] \end{bmatrix} \tag{1.13}$$

[26] ACN index $q = 1$ and 3, and $m = \pm 1$.

[27] Note, this is not the FFT of an FM signal, but, rather, the frequency domain response itself is a phase modulated cosine.

[28] $\omega = 2\pi f$ and $T = 1/f$ where f is frequency.

where:

m is the ambisonic degree of the harmonic[29],

$\hat{\phi}$ is the dispersion constant (e.g. how wide or narrow the image will be),

φ_0 is *Direction of Arrival* (DoA) (e.g. the center of the widened area),

λ is an real integer value index, pertaining to the non-zero entries of the filters,

Q is the "time grid" and defines the number of zero entries between non-zero filter entries in the time domain (i.e. 110), and

Λ is the truncation offset (i.e. 9) that defines the number of non-zero entries in each filter ($0 <= \lambda <= \Lambda$).

Each filter in the 2-by-2 matrix is itself defined by the equation:

$$h(\alpha, \beta, \lambda \mathrm{Q}] = \cos\left(\frac{\pi}{2}|\lambda| + \beta\right) J_{|\lambda|}(\alpha) \tag{1.14}$$

Each parameter of each independent filter in Equation 1.13 is substituted into Equation 1.14 and used to compute time domain filters.

The dispersion constant times m becomes the α value used to compute the Bessel function, the DoA times m becomes the initial phase of the cosine function, and the λ value is used as index for these non-zero entries. λ goes from 0 to 2Λ and is multiplied by Q to find real indices q[30] where the filter takes non-zero entries. The total size of the filter should be $2\Lambda Q$.

Note that the diagonal entries of the matrix are identical, and the anti-diagonal entries have a $\pm\pi$ offset. Based on the figures on the publication [ZFKC14], it appears the filter is then centered with the impulse at 0. However, this causes the filter to

[29] Recall this is both for $\pm m$. Use the absolute value of m.
[30] In the time domain.

no longer be causal[31]. Since the filter shape is perfectly symmetric, we believe the anti-causal section is discarded. The author omits these details, but this would for example explain why the truncation factor is multiplied by 2.

Since this equation is only valid for frequency-dependent rotation about the z-axis, additional frequency-independent rotations are includes allowing the sound to be widened along all axes. The SHs are temporarily re-oriented before widening, and then rotated back to their original position, the same process is undertaken for the frequency-independent rotation algorithm in this suite. By adjusting the parameters of this widening algorithm (e.g. the time grid and dispersion constant) it is also possible to create a diffuse reverberation effect.

Conclusion The **ambiX**, and accompanying **mcfx**, plug-in suite is one of the most comprehensive and easy-to-use tools to create spatial music we have found. The graphics are quite rudimentary compared to other tool-kits, but is is advantageous when computer processing speeds are limited. The **mcfx_gain_delay** effect is particularly useful for calibrating loudspeaker set-ups.

The loudness modification, warping and widening effects are particularly interesting and unique. One possible contribution, beyond implementing the hyperbolic paraboloid warping, would be to create a modular effect where the user could select from a set mathematical functions. Additionally, we could extend the idea of frequency-dependent rotation to generate frequency-dependent warping. The transforms are often ignored due to their impracticality for real-world scenarios, however, they correspond to interesting musical gestures we believe might be worth exploring compositionally.

One of the small criticisms of this library is that one needs to download different VST packages depending on the ambisonic order they wish to work in. Instead, a more elegant approach was taken in the IEM tool-kit, wherein the user can select the

[31] It has non-zero entries before index 0.

number of inputs and outputs. Another issue we encountered is that **ambiX** is not signed properly according to the OSx development guide. With Catalina or higher versions, this prevented us from opening the effects in Reaper, the DAW we chose to use for this analysis. The following command was used to disable this security preference:

```
$ sudo spctl --master-disable
```

This allowed us to use apps from anywhere online (e.g. unidentified developers); without this command the security preferences do not show this option. Another small issue with this suite is that most effects don't have direct *Open Sound Control* (OSC) integration, which means if the DAW does not support OSC, the head-tracker the user is using will not be able to communicate with the VST.

In terms of sound quality the binaural decoder sounds very transparent[32] and the directional loudness plug-in was found to be very helpful. It has an intuitive interface, however, we believe it could be improved by providing the user with visual feedback. The widening algorithm sounded very pleasant, but there are noticeable distortions when modulating parameters. The warping effect was far smoother under modulation, but hard to visualize without additional effects. We decided to use the *IEM EnergyVisualizer* to confirm the output. Using multiple encoded sources greatly improves the warping effect. A DIY head-tracker was used to test all these effects[33].

1.2.1 IEM Plug-in Suite

IEM's plug-in suite is one of the most comprehensive ambisonic tool-kits available today. Much like Kronlachner's suite, these VSTs were developed with JUCE and are completely open source, which allows any developer to copy the files and create their own versions of these effects. In total this toolbox has 20 independent blocks

[32] We used the `iem_cube_h2_mb_dpa_allrad-o5-with-tail` preset.
[33] By Tomas Rudzki. See link.

of software that work together to facilitate ambisonic production.

The first element in the toolkit is the **AllRADecoding** (All-Round Ambisonic Decoding) based on Zotter and Frank's 2012 paper [ZF12]. In AllRAD, we combine VBAP and Ambisonics, to exploit the benefits each have over the other. Ambisonic decoding has been shown to suffer in performance when being rendered using irregularly-spaced speaker arrays. In AllRAD, the ambisonic sound-field is virtually decoded using a t-design, and, subsequently, the signals are re-decoded using the VBAP algorithm. This "maintains the overall variation of the energy while keeping the energy spread small"[34].

The second plug-in is the **BinauralDecoder**, which uses the *MagLS* approach proposed in [SZH18]. Rather than convolving each virtual speaker feed with a corresponding IR, this algorithm encodes HRTF sets into the ambisonic domain using a *Least Squares* (LS) algorithm at low frequency. At higher frequencies, only the magnitude of the HRTFs is considered in the LS problem - as opposed to using complex frequency values[35].

The LS method provides significant computational improvements, considering some data-sets with thousands of HRTFs can be suitably represented with a far smaller number of spherical harmonics. In addition to the timbre correction via the *MagLS* optimization, the plug-in also allows for *Headphone Related Transfer Function* (HpRTF) equalization[36]. These compensate for the irregular frequency responses of various headphones and are included in the plug-in.

Many of the other plug-ins provided by IEM unfortunately do not have extensive publications which accompany them, however, for many of them, we can deduce how they might be implemented using previously discussed concepts. For example the **DirectionalCompressor** is a dynamics compressor that lets you control the dynamics of a spatial region. Using Kronlachner's [KZ14] method for cropping the

[34]More detail of this decoder are presented in Chapter 3.
[35]More information about this method can be found in [ZF19] Section 4.11.2.
[36]These HpRTF data sets can be found here.

sound-field, as discussed in Section 1.2, would be one way to accomplish this. Daniel Rudrich, who authored this plug-in, also included a side-chain feature that allows one to compress the "rest" of the sound-field based on the dynamics of our cropped region. This can be useful for example if a singer is buried in the sound-field, and we want to accentuate this particular signal.

The **DirectivityShaper** is a plug-in from Rudrich, which alludes to the concept of *timbre spatialization* [LS17]. Timbre spatialization is described by some composers as the decomposition of a sound into multiple bands, each of which is independently spatialized. This design goes a bit further, however, by providing control over the directivity of each sub-band. In Rudrich's design, the input mono signal is filtered into "four independent bands, to which different directivity patterns can be applied" [IEMb]. After the signal is split, the user can control the directivity pattern and orientation by virtual beam-forming[37]. These signals are subsequently re-encoded into the ambisonic domain for mixing with other signals, or to apply other effects. Rudrich calls the output of the shaper, "directivity signals" [Rud].

DualDelay is another effect by Rudrich that generates a stereo delay; these delay lines are subsequently panned in ambisonics using LFOs[38]. Rudrich et al. [RZF] describe the prototyping of this effect, which was accomplished using the **ambiX** suite in Reaper. In their publication, the ambisonics workstation developed by Rudrich in Reaper was controlled using a pedal-board on stage. This demonstrates the viability of these effects for live performance[39]. After the performance, the prototype effect was turned into a plug-in for more streamlined usage. "The most exciting feature is the rotation of the whole sound-field every single time it runs through the delay-line. [IEMb]."

Sebastin Grill developed the **FdnReverb** for ambisonic diffuseness control, based on the *Feedback Delay Network* (FDN) designs by Gerzon [Ger76], and Stautner and

[37] The mathematics of this process are also covered in Chapter 3.
[38] Low-frequency oscillators, used to modulate parameters.
[39] Certain effects have higher latency, which make them unsuitable for live performance.

Puckette [SP82]. What is exemplary about these networks is that, much like with ambisonics, linear combinations of audio signals (in this case delay taps) are combined to produce various speaker feeds.

The fact that by their design multiple outputs are produced makes them ideal for multi-channel effect processing. Alary et al. [APSV19] discuss the development of an ambisonic reverberator which preserves directional properties of the sound-field. The traditional **A** matrix in the FDN design is expanded to included sub-matrices created to fit the number of harmonics required for the ambisonic order.

Alary et al. [APSV19] note the computational cost of this task. With 4 delay lines and 3OA[40], 64 total delay lines are required. However, FDNs are normally designed with 32 or 64 independent delay lines to "ensure a sufficiently high modal and echo density". This means the user will need to find a compromise between reverb density and ambisonic order[41].

The **MultiBandCompressor** is another dynamics effect, however, in contrast to other compressors in this suite, this effect splits the ambisonic signal into multiple frequency bands. Two relevant factor stand out from the development of this VST. Firstly, the filters employed to split the spectrum are based on a *Linkwitz-Rilely* crossover design which helps retain the magnitude response of the mix, without introducing phase issues which might affect localization. Secondly, to further ensure localization is preserved, the plug-in analyzes the W (omni) channel, in all four bands, and applies the same gain reduction to all harmonics - this ensures that higher order harmonics still accurately represent the sound-field. The **OmniCompressor**, by Daniel Ridruch, simplifies this a bit - it also uses the W channel to attenuate all harmonics, but there is no multi-band filtering. As a result, the **OmniCompressor** is cheaper computationally, thus multiple instances can be loaded without significant overhead. The **MultiBandCompressor**, on the other hand, might be better

[40] 16 channels.

[41] **FdnReverb** is not based on this paper per se, but [ZF19] seems to suggest a similar approach is used.

reserved for a master bus.

There are some other interesting effects, most notably the **RoomEncoder**, which is based on a real-time *Image Source Method* (ISM) implementation[42]. The ISM is used to simulate room acoustics by modeling reflections from walls using delay lines. These effects are generally very expensive computationally, but offer the ability to change the listener position relative to the source, which ambisonics can't offer unless extended via multiple proximal sound-fields [TC19]. The input of this effect is a mono signal, and the output is an ambisonic signal.

Unfortunately, due to the lack of publications associated with these effects, one must speculate about how they were designed. In contrast to other plug-in suites, however, the IEM suite does have extensive online documentation catered towards musicians. This is likely the biggest benefit of this suite. In contrast to other tool-kits, we found IEM's tool-kit was easy to install and straightforward to use. The sound quality was great and we found everything worked as intended.

The online documentation (e.g. [IEMa]) even describes some wrappers one can use to load VSTs in Pd or SuperCollider, which means if the latency settings are accounted for, these can be used in real-time computer music pieces[43]. Since there are so many effects in this suite, we were not able to discuss all of them in detail here. We chose to focus on those we believed were most interesting and provide the most creative potential. We advise the reader to download the suite themselves and explore the other plug-ins it offers. This is the suite we chose to use for our latest musical project, described in Section 1.3.

For someone interested in a professional grade solution to ambisonic production, we believe this tool-box is the best place to start. The final tool-box we will discuss, has more sophisticated effects, but for most people, IEM's is good enough. Some of our favorite features of this tool-box include:

[42]The documentation is lacking, but the code seems to suggest this.
[43]See the FAQ.

1. The ability to save and load presets,

2. The automatic re-configuration of the effect, with regards to channel count, based on the ambisonic order, and,

3. The ability to send OSC messages directly to the binaural decoder[44].

1.2.2 Sparta Plug-in Suite

The Sparta Plug-in Suite from Aalto University is another important toolbox for production of ambisonic music. Thankfully, there are great number of academic papers related to its development so we don't need to speculate how these were created. Much like before we will try to focus our attention on plug-ins that are specific to this toolbox, and ignore those that are ubiquitous to all suites. The SPARTA suite is split into four main sections:

1. **SPARTA suite**: this is the main part of the total suite. It includes all the basic ambisonic plug-ins, however, the decoders provide additional features compared to IEM, and there are a few unique effects as well.

2. **COMPASS suite**: based on the Coding and Multi-directional Parameterisation of Ambisonic Sound Scenes (COMPASS) [PTP18].

3. **HO-DirAC suite**: Higher Order Directional Audio Coding suite is based on the DirAC method [Pul07].

4. **Other**: this category includes the **cropac_decoder** (Cross-Pattern Coherence Decoder) and the **HOSSIR** (Higher-order Spatial Impulse Response Rendering) effect.

SPARTA Suite

[44]In fact, all effects accept OSC messages. See link for more info.

ambiBIN The first plug-in in the SPARTA suite [MP19][45] is called **AmbiBIN** which, as the name suggests, is a binaural decoder. The decoder has a built-in rotation and accepts OSC data from a head-tracker. According to the online documentation [McCb] "the rotation angles are updated after the time-frequency transform, which allows for reduced latency compared to **AmbiDEC** + **Rotator**[46]".

In other words, rotation matrices in this effect are calculated using complex spherical harmonics, rather than real-valued ones. The plug-in also allows SOFA [MIC+13] HRTFs to be imported. It features four different types of binaural decoding: *Least-Squares* (LS), *Spatial Re-Sampling* (SPR), *Time-Alignment* (TA), and *Magnitude Least-Squares* (MagLS). Since the LS and MagLS methods have already been discussed, we will focus on TA and SPR here.

Bernschütz et al. [BGPA14] describes the process of Spatial Re-Sampling. The methods objective is to reduce coloration usually associated with the LS method of HRTF encoding. In this method, rather than using the largest possible number of HRTFs that adhere to some spatial sampling scheme, the encoding is performed with a subset of IRs. This subset of IRs are selected to match to same degree the spatial resolution of the ambisonic signals. In addition, the binaural decoding matrix can be re-sampled at various orders to examine the impact in quality.

> "Down-sampling[47] reduces the spatial sensitivity of the HRTFs, and enables their adaptation to a system with lower modal resolution[48]."

The author also found that the type of grid used for sampling the HRTFs had an effect on the reproduction quality. A *MUltiple Stimuli with Hidden Reference and Anchor* (MUSHRA) test revealed improvements towards the unwanted high-frequency roll-off associated with previous methods (e.g. LS method). It also quantified how

[45] This reference provides an overview of SPARTA and COMPASS.
[46] Two other Sparta plug-ins. We will cover **AmbiDEC** later. The Sparta **Rotator** is a time-domain HOA rotation plug-in.
[47] The binaural decoder, in the spatial domain.
[48] For example FOA.

many orders can be removed for satisfactory reproduction. This is useful for determining the smallest number of convolutions required for suitable playback of low order ambisonic signals.

The second method is called the Time-Alignment [ZSH18] method which is paired with a diffuse-coherence constraint/correction in the associated paper[49]. As the authors note, the proposed binaural renderer is based on two stages: pre-processing and and optimization. The pre-processing stage refers to the time-alignment, which involves a frequency-dependent ITD equalization which retains ITDs at low frequencies[50] and removes them for high frequencies[51]. According to the authors: "Due to the time alignment, lower SH orders are sufficient to represent the directivity patterns of the ears at higher frequencies." This means a smaller number of SH are needed to retain directional information. This process is accomplished by using an all-pass filter whose phase response is dictated by a value τ corresponding to the time difference between ears relative to the center of the head according to a simple geometric model:

$$\tau_p^r = \cos(\theta_p)\sin(\phi_p)\, r_H c^{-1}, \quad \tau_p^l = -\tau_p^r \tag{1.15}$$

where

τ_p^r is the time offset for the left ear for grid node p,

θ_p is the elevation angle of the grid node[52],

ϕ_p is the azimuth angle of the grid node,

[49] Although the plug-in allows one to apply this correction to any of the other decoders methods as well.

[50] Below 1.5kHz in this paper ("empirically chosen").

[51] ITDs are a more salient binaural cue at low-frequencies. ILDs are more salient at high frequencies. This is called Duplex theory, proposed by Rayleigh in 1907.

[52] Traditional ambisonic coordinate system. Azimuth increases counterclockwise, elevation spans $\pm 90°$.

r_H is the head radius, 8.5cm in this paper, and

c is the speed of sound (343 m/s).

The all-pass filters for each corresponding ear and individual grid node position are calculated according to:

$$A_p^{l,r}(\omega) = \begin{cases} 1 & \text{for } \omega < \omega_c \\ e^{-i(\omega-\omega_c)\tau_p^{l,r}} & \text{for } \omega \geq \omega_c \end{cases} \tag{1.16}$$

Where $\omega_c = 2\pi f_c$ with f_c being the cut-off frequency of 1.5kHz, and, $i = \sqrt{-1}$ being the imaginary number. $\tau_p^{l,r}$ comes from Equation 1.15. Each HRTF for each position and ear is then multiplied in the frequency domain with its corresponding all-pass filter:

$$\hat{h}_{l,r}\left(\Omega_p, \omega\right) = h^{l,r}\left(\Omega_p, \omega\right) A_p^{l,r}(\omega) \tag{1.17}$$

where Ω_p is the matrix containing all grid node positions.

The optimization stage refers to the aforementioned diffuse-coherence constraint and correction. The purpose of this is to achieve the diffuse-field response and *Interaural Coherence* (IC) behavior of the original HRTF set. An SVD[53] approach is employed using these additional constraints, which force the solution to remain as similar as possible, with regards to these criteria, as the original HRTF set.

The diffuse-field response criteria is implemented using a covariance matrix. In [ZSH18] it is given by:

$$\boldsymbol{R}_H \cong \boldsymbol{H}\boldsymbol{W}\boldsymbol{H}^{\mathrm{H}} = \begin{bmatrix} r^{ll}(\omega) & r^{lr}(\omega) \\ r^{rl}(\omega) & r^{rr}(\omega) \end{bmatrix} \tag{1.18}$$

The superscript H corresponds to the conjugate transpose - this means that all the bins in the matrix are complex conjugated and then the matrix is transposed.

[53] Singular Value Decomposition, explained in more detail in Section 1.2.2.

The $\boldsymbol{H}$ corresponds to our matrix of HRTFs, and $\boldsymbol{W}$ to the weights according to the sampling scheme. The main diagonal entries correspond to the diffuse-field energy of the left and right ear. The other entries provide the covariance between ears.

After this covariance matrix has been used to yield the decoding matrix, these values are re-introduced in the evaluation phase, which measures IC - among other qualities. The IC parameter is meant to determine if the decoding matrix indeed has a similar diffuse-field behavior to the reference decoder. IC is low in diffuse-field, and high in free-field. Equation 1.19 shows the calculation for the *Interaural Coherence* measure.

$$IC(\omega) = \frac{\left|r^{lr}(\omega)\right|}{\sqrt{r^{ll}(\omega)r^{rr}(\omega)}} \tag{1.19}$$

ambiDEC The next plug-in in this comprehensive list is the **sparta_ambiDEC** which is designed for loudspeaker-based reproduction. Much like the binaural decoder, it offers various ways of decoding the sound-field. The list of techniques includes: All-Round (AllRAD), Energy-Preserving (EPAD), Sampling (SAD)[54], and Mode-Matching (MMD). AllRAD, SAD and MMD are covered in Chapter 3, therefore, in this section, we will focus on the EPAD decoder.

Zotter [ZFP13] offers a good overview of these different decoding systems, along with a comparative analysis. In order to describe EPAD we first need to define *Singular Value Decomposition* (SVD), as the method relies on it.

SVD is a method used to compute the pseudo-inverse of a matrix. The SVD factors any matrix $\mathbf{A}$ (real or complex) into three other matrices:

$$\mathbf{A} = \mathbf{U} \times \mathbf{\Sigma} \times \mathbf{V}^{\top} \tag{1.20}$$

$\mathbf{U}$ and $\mathbf{V}$ are orthonormal[55] and $\mathbf{\Sigma}$ is diagonal. The pseudo-inverse is then given

[54]The online documentation has an error. They use the word spatial, but they mean sampling ambisonic decoder.

[55]The vectors are perpendicular to one another and of unit length.

as:

$$\mathbf{A}^{\dagger} = \mathbf{V} \times \Sigma^{\dagger} \times \mathbf{U}^{\top} \tag{1.21}$$

$\Sigma^{\dagger}$ is easy to compute because it is diagonal; we simply substitute each non-zero entry with its reciprocal [HLB08][56]. The diagonal entries of $\mathbf{\Sigma}$ are called the *singular values* and $\mathbf{U}$ and $\mathbf{V}$ are both *unitary*[57] matrices with dimensions m by m, and n by n respectively. Both of these matrices correspond to rotations of the space, while the singular values represent scaling operations. The columns of $\mathbf{U}$ and the columns of $\mathbf{V}$ are also called the *left-singular vectors* and *right-singular vectors* of $\mathbf{A}$. The matrices also represent bases of the original matrix. The singular values represent the eigenvalues, which correspond to the magnitude of these bases.

In EPAD, we perform SVD twice, once on the original matrix of spherical harmonics representing the speaker positions, and a second time on this same matrix multiplied by the truncated unitary matrix $\mathbf{U}$, from the first SVD. The basis of EPAD is the removal of the singular values, which corresponds to an energy-preserving decoder[58]:

$$\boldsymbol{D}_{\mathrm{EPAD}} = \hat{\boldsymbol{V}}\hat{\boldsymbol{U}}^{\mathrm{T}} \tag{1.22}$$

However, while this decoder design does preserve energy reasonably well, it does not perform ideally under the assumption of hemispherical speaker arrangements. This led the derivation of a new set of basis functions: "designed in the continuous domain such that they allow for energy-preservation on the desired fraction of the sphere [ZPN12]." These new basis functions come from the shortened $\mathbf{U}$ matrix from the original SVD. Namely, the $(2N+1)$-sized matrix is reduced to $N+1$ columns. The vectors representing the most energy are kept, and those representing the least

[56] See appendix [HLB08].

[57] A square matrix is unitary if its conjugate transpose is also its inverse: $U^{*}U = UU^{*} = I$.

[58] In contrast to the SAD: $\boldsymbol{D}_{\mathrm{SAD}} = (4\pi/\mathrm{L})\hat{\boldsymbol{V}}\hat{\boldsymbol{S}}\hat{\boldsymbol{U}}^{\mathrm{T}}$.

energy are discarded, this is done with simple indexing since the SVD algorithm sorts the eigenvectors by design.

$$\check{\boldsymbol{U}}\check{\boldsymbol{S}}\check{\boldsymbol{V}}^{\mathrm{T}} = \mathrm{svd}\left\{\boldsymbol{U}^{\mathrm{T}}_{:,1:\mathrm{N}+1}\boldsymbol{Y}_{\mathrm{N}}\right\} \tag{1.23}$$

Equation 1.23 shows the re-expansion of the loudspeaker positions in SH domain, represented by $\boldsymbol{Y}_{\mathrm{N}}$, via the truncated $\boldsymbol{U}^{\mathrm{T}}_{:,1:\mathrm{N}+1}$ matrix. $\boldsymbol{U}$, as we noted, is given by the first SVD of the SH matrix representing the speaker array. The notation defines the retention of all rows, and $(N+1)$ columns, the matrix is then transposed and multiplied with the original SH function. Finally, equation 1.23 shows a second SVD performed upon the product [ZFP13].

$$\boldsymbol{D}_{\mathrm{EPAD}} = \check{\boldsymbol{V}}_{:,1:\mathrm{N}+1}\check{\boldsymbol{U}}^{\mathrm{T}}\boldsymbol{U}^{\mathrm{T}}_{:,1:\mathrm{N}+1} \tag{1.24}$$

Equation 1.24 shows the final EPAD decoding, which combines the unitary matrices from the initial and second SVD. The final $\boldsymbol{D}_{\mathrm{EPAD}}$ matrix is multiplied with the B-format signals yielding our speaker signals.

In addition to the various decoding methods provided, the plug-in also allows for binaural reproduction using SOFA formatted HRIRs. The virtual loudspeaker approach is used in this plug-in in contrast to **AmbiBin**. Finally, the plug-in allows for: double decoding using a specified cross-over frequency, importing JSON files corresponding to speaker layouts, and energy or amplitude normalization, when using multiple decoder orders.

DRC + ISM Model The next plug-in is the **AmbiDRC** (dynamic range compressor) which unlike traditional compressors is frequency dependent, and, like IEM's compressor, is also based on the analysis of the W harmonic - to retain localization accuracy. The internal FFT-based DRC is explained in [MV+17]. It is described as a high-resolution multi-band compressor, however, current plug-in does not reveal these parameters, so it acts more like the **OmniCompressor** from

IEM. **AmbiRoomSim** is another effect in the suite that work similarly to the IEM **RoomEncoder**: they are both ISM ambisonic panners that simulate reflections using a "shoebox" room model.

Array2SH The **Array2SH** VST is used to encode ambisonic arrays into the ambisonic domain. The plug-in allows one to import microphone coordinates, for custom arrays, and select various other properties of the array design, such as: rigid/open enclosure, spherical/cylindrical geometry, radius of baffle, radius of array (in cases where sensors protrude from baffle) and sensor directivity. These are all used to generate the appropriate encoding matrices and equalization filters used to convert signals into the SH domain. This plug-in does not use any measurements to encode ambisonic arrays. Unfortunately, since the behavior of capsules is not linear, using a model to encode ambisonic arrays generally leads to imperfect results. In Chapter 2 we discuss more about the process of encoding microphone array signals into the ambisonic domain.

Binauraliser This effect is an interpolation-based HRTF binaural plug-in which can be used for headphone virtualization of surround sound systems, for example. In this VST, ITDs are extracted using the "cross-correlation between the low-passed filtered left and right time-domain responses [MP19]." The interpolation values are calculated at run-time and applied to the magnitude responses and ITDs separately, which are then recombined per frequency band.

Cross-correlation is given by:

$$(f \star g)[n] \triangleq \sum_{m=-\infty}^{\infty} \overline{f[m]} g[m+n] \tag{1.25}$$

and is calculated for every value n. Since both signals are real-valued $\overline{f[m]}$, the complex conjugate operator, yields $f[m]$. A complex number is equal to its complex conjugate if its imaginary part is zero, or equivalently, if the number is

real. We should see a spike when signals align, which we can use to calculate the ITD. This allows us to process signals in the frequency domain saving us significant computations required from convolution.

Unfortunately, there is not an associated publication that provides in depth information. While HRTF interpolation has been researched for many years now, we would have liked to see references in the online documentation for those not as familiar with the concept, even if these were written by authors at other Universities. This plug-in also supports OSC-based head-tracking and SOFA HRIR sets. One possible experiment might involve comparing the latency and quality difference between this effect and the ambisonic-based binaural synthesis system **ambiBin**.

Decorrelator In the online documentation [McCb] there is also an interesting **Decorrelator** plug-in "based on randomised time-frequency delays and cascaded all-pass filters." However, this plug-in also has no references yet and has limited features. The GUI suggests two important features: the ability to bypass transients, and compensate level. This effect is not intended to be used with ambisonic signals, as it would likely destroy the spatial image[59]. Using a system akin to ATK [LA14], it should be possible to virtually decode each ambisonic channel, apply a decorrelation, and re-encode to preserve the spatial image.

Visualization FX The SPARTA suite also has three different visualization effects: **sparta_dirass**, **sparta_powermap**, and **sparta_sldoa**. For sake of brevity, a thorough discussion about these effects will be neglected. The internals of all these three different effects are quite sophisticated mathematically. Here we will only give a textual description since none of these actually affect the sound signals.

The **DirASS** (Directional re-assignment) [MPP19] relies on initial *Direction of Arrival* (DoA) estimates, which it uses to assign a beam-former to regions of high

[59]Except if one chose to do this intentionally for musical effect.

intensity. The plug-in allows one to place a real-time video behind the activity map, which allows one to make sure ambisonic recordings are properly lined up with 360° videos. Alternatively, this can be useful for adding foley or sound effects to immersive media.

The **PowerMap** plug-in [MDMP17] uses a statistical approach to reveal the likely direction of sound arrival based on various different analysis methods, which the user can select. For the end user who knows nothing about the underlying algorithms, **DirASS** and **PowerMap** appear to do the same thing. The latter also allows footage to be overlaid on the activity map to create a "make-shift acoustic camera. [McCb]".

Finally, the **SLDoA** (Spatially Localized DoA) estimator uses VBAP beampatterns for directions that are uniformly distributed on the surface of a sphere [McCb], [MDMP+19]. It then analyses 0th and 1st order ambisonic components at these directions and displays different color circles for sub-bands of the signal. In contrast to the **PowerMap** or **DirASS** visualization, this effect is far inferior in quality, but much faster due to the simplicity of the design. Much like it's counterparts, one can also place a video on this GUI's display.

Convolution FX The SPARTA suite also features two different convolution plug-ins with slightly different use cases. The **MatrixConv** plug-in does convolution and matrixing in a single application. The documentation provides three possible use cases for this effect:

1. **Spatial reverberation**: involves taking the IR of a room with an ambisonic array and then convolving a mono signal with these responses to produce an ambisonic auralization of the space.

2. **Mic Array to Ambisonics Encoding**: using a matrix of filters it is possible to encode HOA arrays to ambisonics. In this situation, the matrixing is required

to combine all filtered capsule signals to produce the resulting harmonic.

3. **Advanced Spatial Reverberation**: using again an ambisonic IR, import it into **HO-SIRR**, another plug-in from Aalto, to generate virtual speaker responses simulating the original space. Then import these into **MatrixConv** for playback over speakers or headphones. The authors concluded that this parametric approach is more accurate and flexible than the direct convolution described in use case 1.

The **MultiConv** effect is similar to **MatrixConv** but does no matrixing of signals. So the number of signals in, will be the same number of signals out. One use case described by the author is headphone equalization[60]. Another use case could be ambisonic recordings, in lieu of mono recordings, convolved with ambisonic IRs.

Panner, Rotator, Spreader The SPARTA **Panner**, described in [LVJ+14], is an advanced VBAP [Pul97] implementation that takes into account frequency dependent effects of speaker placement and room acoustics, and uses these additional criteria to normalize signal gains in the frequency domain. The authors used simulations in MATLAB using various *Direct-to-Total* (DTT) energy ratios to derive a frequency-dependent gain function, which was then used in a subjective experiment confirming the reliability of the model.

The **Rotator** is similar to other rotators we discuss in Chapter 3. For HOA rotation matrices there are three common implementations:

1. **Zotter's Method**: derived in his thesis, involves computing a single 90° R_y rotation matrix to position the harmonic for continuous rotation using a more easily derived R_z matrix.

[60]Headphone equalization filters can be found here.

2. **Complex SH Rotation**: rotations in the frequency domain are simpler to calculate for SHs, however, all signals need to be transformed using an FFT, which might prove computationally expensive.

3. **Recurrence Relation**: using recurrence relations it is possible to calculate rotation matrices for all real-valued SH [IR96].

The SPARTA **Rotator** uses the third technique, also used in [GAK+19] and [PPQ16]. Finally, the **Spreader** is described as:

> "An arbitrary array (e.g., HRIRs or microphone array IRs) panner with coherent and incoherent spreading options.[McCb]"

Unfortunately, no other information was found regarding this plug-in. What we surmise is this VST was designed to be used with sets of HOA microphone array IRs, which can be linearly combined, after convolution, for panning of virtual sources. Luckily, the developer, Leo McCormack, gave us additional information over email correspondence. We quote:

> "The 'Basic' spreading mode is just making identical/coherent copies of the input signal and convolving them with all the HRIRs (or, e.g. microphone array IRs) that are within the spreading area, followed by taking the mean. It's the simplest way to make spread sound sources (this also has parity with e.g. MDAP used for loudspeaker panning spreading). However, it doesn't really sound that great for large degrees of spread, so more elaborate methods tend to instead use decorrelated/incoherent copies of the input signal and place them around the spreading area.
>
> The 'OM' (optimal mixing) and 'EVD' methods are our way of doing incoherent source spreading by directly imposing the diffuse-field characteristics for a spread source onto the input signals (i.e. no ad-hoc placement of many decorrelated copies, it's instead a direct solution). The 'EVD' (eigenvalue decomposition) does this without any constraints, so the signal quality is quite poor (it's only there for demonstration purposes), whereas 'OM' adds some constraints to fulfil the target diffuse source characteristics while also retaining the signal quality."

COMPASS Coding and Multidirectional Parameterisation of Ambisonic Sound Scenes (COMPASS) [PTP18] is based on parametric methods generally used for audio data compression. In this case, however, the statistics are used to provide individual control over directional and incoherently distributed sources (e.g. diffuse sound). The method relies on the decomposition of a complex sound-field, represented by spherical harmonic signals, into source signals (e.g. directional elements) and diffuse sound. Mathematically this is expressed as

$$\mathbf{a}(t,f) = \mathbf{a}_{\mathbf{s}}(t,f) + \mathbf{a}_{\mathrm{d}}(t,f) = \mathbf{Y}_{\mathbf{s}}(t,f)\mathbf{s}(t,f) + \mathbf{a}_{\mathrm{d}}(t,f) \tag{1.26}$$

where $\mathbf{a}(t,f)$ is the original ambisonic signal, $\mathbf{a}_{\mathbf{s}}(t,f)$ represents the source signals, and $\mathbf{a}_{\mathrm{d}}(t,f)$ represents the diffuse part of the sound-field. Furthermore, we see that the directional part of the sound-field can be represented by $\mathbf{Y}_{\mathbf{s}}(t,f)\mathbf{s}(t,f)$, where $\mathbf{s}(t,f)$ are unknown non-directional source signals, and $\mathbf{Y}_{\mathbf{s}}(t,f)$ are the encoding harmonics corresponding to the DoA of those signals.

COMPASS relies on an *eigenvalue decomposition* (EVD) of the *Power Spectral Density* of the original ambisonic signal. The EVD is a factorization of a matrix into linearly independent vectors, called eigenvectors, and their corresponding eigenvalues, that determine the magnitude of these vectors. These operations are performed using linear algebra tool-boxes under the hood[61].

The PSD matrix can be defined as a temporal and frequency averaging of the power (e.g. magnitude squared) of the auto-correlation function (e.g. the convolution of the original signal with itself). In Politis et al. [PTP18], it is defined as

$$\mathbf{C}_{\mathbf{a}}(t,j) = \alpha\mathbf{C}_{\mathbf{a}}(t-1,j) + \frac{(1-\alpha)}{\Delta f_j}\sum_{f_{j-1}+1}^{f_j}\mathbf{a}(t,f)\mathbf{a}^{\mathrm{H}}(t,f) \tag{1.27}$$

where:

α is a temporal averaging coefficient whose value should be between 0 and 1,

[61] See OpenBLAS.

j is the frequency averaging coefficient (e.g. the averaged band index),

f_j is the upper frequency index,

$\Delta f_j = f_j - f_{j-1}$ (e.g. the bandwidth of the filter), and

$f_0 = 0$.

$\mathbf{a}(t, f)$ and $\mathbf{a}^{\mathrm{H}}(t, f)$ are vectors containing a single bin. These bins contain the magnitude squared of all harmonics at that frequency. In FOA, for example, the resulting matrix is a four-by-four matrix. We must calculate multiple all matrices, in the defined frequency range, and average these according to the bandwidth of the band[62]

The α coefficient is linked to the decay time constant τ via the following equation

$$\alpha = e^{-R/(\tau f_s)} \tag{1.28}$$

where:

R is the hop size of the STFT[63], and

f_s is the sampling rate.

In COMPASS thee *Equivalent Rectangular Bandwidth* (ERB) values are used in the frequency averaging operation, which attempts to describe the frequency-dependent behavior of the basilar membrane (e.g. our inner ear). These frequency values define the starting, center, and stopping frequencies of filters modeling our auditory system's behavior in response to external stimuli[64].

[62] It is impossible to know the $t-1$ PSD, therefore the matrix at time 0 will be only the right-hand side term.

[63] The number of samples we jump from one FFT analysis to another. Typically the hop size is half or a quarter of the block size, depending on desired resolution.

[64] The author does not provide equations to compute these.

The number of sources K is "estimated from the distribution of eigenvalues $\mathbf{U}$". K is assumed to be smaller than the number of spherical harmonics, which is a limitation of this algorithm. The SORTE method [HN13], is used to determine the number of sources based on statistical measures of the eigenvalues from the EVD of the PSD. The variance of eigenvalues is also used to determine the measure of diffuseness according to the method in [EJ16].

Finally, a *Multiple Signal Classification* (MUSIC) algorithm is employed using the calculated value of K[65]. With the MUSIC algorithm, the DoA of K sources can be determined. With the DoAs found, the authors extract the source and ambient signals, and their powers, using a beam-forming matrix. The diffuse signal is found by subtracting the directional sources from the original signal.

In a data compression system, this would allow one to transmit only K sources, rather than all spherical harmonics, along with one additional channel corresponding to the diffuse field. This, along with meta-data defining the direction of sources, would be used to reduce the transmission load. In COMPASS, however, this method is used to provide additional control over the sound-field, allowing some interesting effects to be possible. In the next section, we will describe some of these effects.

Decoder The **COMPASS decoder** [PTP18] allows for FOA, 2OA and 3OA decoding with up to 64 outputs. It contains a binaural engine with SOFA HRIR sets, however, no rotation system is integrated into the plug-in[66].

Due to its parametric approach to decoding, the effect features two key features not possible using traditional decoder:

1. Balance control between direct sound and ambient components.

[65] K corresponds to the number of sources, calculated using the SORTE algorithm.

[66] Therefore, for binaural auditioning, one might need to send the output of this effect to SPARTA's binauraliser, for example.

2. Mixing control between fully parametric decoding and linear ambisonic decoding. [McCa]

The diffuse/direct control creates a similar effect to that of de-reverberation. In contrast to other effects in which this is accomplished simply by attenuating the omnidirectional 0^{th} harmonic, the parametric approach from COMPASS applies this de-reverberation to all channels of the B-format signal.

Because the parametric synthesis can yield artifacts, especially when decoding FOA, the linear ambisonic decoding balance parameter is also provided. This will determine the blend between the parametric and linear decoding signals presented to the listener.

Since the effect relies on complex time-frequency analysis/resynthesis, the authors note that large buffer sizes are required [MP19] - making this effect unsuitable for interactive works which might require low latency[67]. It is well known that the size of the FFT determines the time-frequency resolution of the analysis/resythesis; larger FFT sizes provide better frequency resolution at the expense of time resolution, and vice versa.

Binaural The **COMPASS binaural** effect is an optimized version of the **COMPASS decoder** allowing which is used for binaural decoding using the COMPASS system. Much like the **COMPASS decoder**, it gives one control of the direct/diffuse sound balance and parametric/linear decoding balance. The VST allows one to use their own SOFA HRIRs and supports head-tracking hardware via OSC. It is meant for HMD[68] ambisonic content and allows for a reduces buffer size of 512 samples, which reduces the latency to ∼10 msec at 48kHz [McCa].

[67] The minimum buffer size for this effect is 1024 samples.
[68] Head Mounted Display.

BinauralVR This effect is similar to the **COMPASS binaural** VST, with the exception that is allows for 6DoF binaural decoding of the ambisonic sound-field. The effect assumes that all signals are initially positioned 2 meters away from the center of the listener, although this setting can be modified to increase the radius of the sphere. The same parameters as the other two decoders mentioned in COMPASS are available[69]. The VST then receives orientation and translation data via OSC and has an ID so multiple instances can be loaded in a single DAW project. Unfortunately, the documentation does not provide information regarding how this is accomplished, leaving the reader to speculate about its design. We believe most likely the system uses the ISM and subsequently decodes the SH output using a binaural decoder.

Gravitator The **COMPASS Gravitator** is not comprehensively documented but remains a one-of-a-kind effect that warps the sound-field using parametric data. The effect analyzes the SH signals for directional components and subsequently allows the user to pull these multiple sources towards one or more specified positions according to some gravitational parameter. The gravitational parameter dictates how stronger the pull towards the desired direction(s) will be. While other warping effects have formerly been described, such as those in Section 1.2 [Kro14a], none use a parametric approach, which ultimately provides greater control and flexibility over encoded signals. In contrast to other effects, this plug-in extract source signals from a complex ambisonic signal and allows one to change their position according to this warping.

Sidechain The **COMPASS SideChain** is not a compressor, in the traditional sense. The name might be confusing, given that sidechain dynamic compression is a well-known technique in audio production. What this effect does, instead, is it imposes the diffuse and direct field analysis of one signal to another.

[69] Diffuse/direct and parametric/linear control.

It is an experimental design, which aims at creatively distorting the spatial image using a parametric approach. Given two input sound-fields[70] the DoA and diffuse measures of one can be imposed on the other and viceversa. This effect is the most conceptually interesting effect we found in the toolkit. One possible improvement could be a slider that allows the musician to control the **COMPASS SideChain** mode in a more gradual manner[71].

SpatEdit The **COMPASS SpatEdit** software requires two instances of the effect to be used simultaneously. There are two possible ways to use this effect. In the first scenario the first instance tracks the directional source signals and outputs independent beam-former signals, corresponding to these sources. The user can then modify these signals using any traditional mono effect. The modified signal is then passed to the second instance, which also receives the parametric information from the first instance, and subsequently re-synthesizes the SH signals.

In the second scenario, rather than the first instance of **COMPASS SpatEdit** outputting the directional signals, it outputs the diffuse part of the sound-field. The directional signals are passes internally to the second instance. This allows the user to apply parametrically, any effect they desire on the incoherent part of the original SH signal. This is reminiscent of the ATK [LA14] virtual decoding method. However, in contrast to the ATK method [LA14], which gives users control of various regions of the sound-field, the COMPASS method gives users control of individual sources, encoded in the SH signal, and the overall diffuse signal.

Tracker The **COMPASS Tracker** is likely the most sophisticated and complex ambisonic effects in this toolkit. The primary function of the effect is to track the position of sources as they move around the sound-field. Naturally, for complex

[70] Scene A [channels 1- 16] and scene B [channels 17-32].
[71] Tthe range could also include negative values, to dynamically switch the imposition.

sound scene with multiple sources moving simultaneously, often overlapping each other, this can be extremely difficult.

The most interesting feature of this effect perhaps is the ability to decompose a sound-field into "stems", even if there are moving sources in the sound scene. According to the author [McCa], due to the statistical uncertainty of this task, the effect often requires tuning, in order for accurate results to be displayed.

It also should be noted, that because of the probabilistic nature of the system, the same sound-field might yield different results, however, after tuning the effect, the results should be consistent. The only issue we have with this effect is the visualization, which is harder to interpret than some of the other visualizers we talked about.

Upmixer **COMPASS Upmixer** is designed to take lower order ambisonic signals and attempt to generate a HOA components using the statistical framework of the COMPASS algorithm. This is useful for example when a musician has recordings from FOA microphone which they wish to combine with synthesized HOA material. In order for both signals to have equivalent spatial resolution the FOA needs to be upmixed to HOA. The plug-in has a diffuse/direct control allowing one to determine the directional balance of the transformation, as well as drop-down boxes for input/output order and format (e.g. ACN and SN3D).

HO-DirAC *Higher-order Directional Audio Coding* (HO-DirAC), in contrast to COMPASS, performs parametric analysis inside pre-defined angular sectors that are independent of the ambisonic signal. In other words, in contrast to COMPASS, where DoAs are estimated before beam-forming, in DirAC, beam-forming is applied using a regular distribution of directions from which diffuseness and directional predictions are found. In Politis et al. [PVP15], the authors describe using a number of t-designs for segmenting the sound-field during the analysis stage of this method.

Much like COMPASS, signal statistics such as the PSD and *Cross Spectral Density* (CSD) are calculated to determine the character of these signals. This requires again the use of a STFT which is time-averaged and a filter-bank, in this case, a *Quadrature Mirror Filter-bank* (QMF), which is not perceptually motivated but designed to allow perfect reconstruction of the signal in the mathematical sense[72].

In this chapter, we will only describe mathematically the analysis stage of the HO-DirAC method. For different applications (i.e. microphone array processing, binaural decoding, or loudspeaker array decoding) the synthesis considerations vary. A limited discussion on the specifics for these reproduction methods will follow.

HO-DirAC Analysis

Given $p(k)$, corresponding the the FFT of the omnidirectional ambisonic pressure signal (e.g. W harmonic) and $v(k)$ an $(N+1)^2-1$-sized vector containing the FFT of all other harmonics at bin k, the *instantaneous power spectrum* of the pressure signal is denoted:

$$\hat{S}_{\mathrm{pp}}(k) = |p(k)|^2. \tag{1.29}$$

which as we see is simply the magnitude squared.

The *combined power spectrum* of the velocity signals is denoted:

$$\hat{S}_{\mathrm{vv}}(k) = \mathbf{v}^H(k)\mathbf{v}(k) \tag{1.30}$$

where the H superscripts denotes the Hermitian transpose of vector[73]. Note that both the combined power spectrum and instantaneous power spectrum (of the pressure signal) return scalar quantities. Meanwhile, the cross-spectrum between pressure and velocity is denoted as the vector[74]:

[72] By using symmetric filters the phase and magnitude of the signals are preserved.
[73] That is, the complex conjugate plus transposed version of the original vector.
[74] Whose size is the same as the original $v(k)$ vector.

$$\hat{\mathbf{s}}_{\mathrm{pv}}(k) = p^*(k)\mathbf{v}(k) \tag{1.31}$$

where the start superscript denotes the complex conjugate of the complex value at $p(k)$.

These three quantities are used to calculate the three key parameters used by the DirAC algorithm for signal analysis. These three quantities are called the *active intensity vector* ($\mathbf{i_a}(k)$), *energy density* ($E(k)$), and *diffuseness* ($\psi(k)$).

$$\begin{aligned} \mathbf{i_a}(k) &= -\Re\left\{\hat{\mathbf{s}}_{\mathbf{pv}}(k)\right\} \\ E(k) &= \frac{1}{2}\left[\hat{S}_{\mathrm{vv}}(k) + \hat{S}_{\mathrm{pp}}(k)\right] \\ \psi(k) &= 1 - \frac{2\left\|\Re\left\{\hat{\mathbf{s}}_{\mathrm{pv}}(k)\right\}\right\|}{\hat{S}_{\mathrm{vv}}(k) + \hat{S}_{\mathrm{pp}}(k)} \end{aligned} \tag{1.32}$$

where $\Re$ corresponds to taking the real part of the complex number and the norm operator is defined as $\|\boldsymbol{x}\|_2 := \sqrt{x_1^2 + \cdots + x_n^2}$. Note once more that energy and diffuseness correspond to scalars, while $\mathbf{i_a}(k)$ corresponds to a vector. The DoA can then be extracted via

$$\mathbf{n}(\Phi(k)) = \frac{\Re\left\{\hat{\mathbf{s}}_{\mathrm{pv}}(k)\right\}}{\left\|\Re\left\{\hat{\mathbf{s}}_{\mathrm{pv}}(k)\right\}\right\|}. \tag{1.33}$$

These same analytic methods are then applied to sectors of the sound-field generated via regularly distributed beam-formers. This allows for local and global analysis of the sound-field, where the latter represents a spatially sampled analysis. The sector profiles are used in a synthesis stage which uses decorrelation functions for diffuse-field generation, and a VBAP design to pan extracted sources in the final representation. The synthesis uses linear combinations of the beam-formers to attempt to reproduce the original sound-field's statistical properties as best as it can.

In the use of creative applications, however, this provides not only individual control of the diffuse and directional parts of the signal, but one may chose to inten-

tionally distort angular segments of the original ambisonic signal. In the next section we will discuss some existing effects based on the HO-DirAC algorithm.

Decoder Politis et al. [PVP15] describe the synthesis method involved this decoder. As noted on the online documentation[75] a "covariance-domain framework for spatial audio processing" is employed to generate the decoded signals. The decoder first generates "prototype" decoding signals using the EPAD method discussed earlier. These "prototype" signals are analyzed using the HO-DirAC method to produce target covariance matrices, which are used to approximate the EPAD decoder as best as possible.

The decoder, much like other parametric decoders from Aalto, provides diffuse-to-direct and linear-to-parametric control. We believe there are additional creative possibilities using this method that have not yet been explored. For example, one could design an effect that would let them swap the positions of multiple regions of the sound-field, or pull all regions towards a desired direction. Alternatively, one could select multiple regions and only reduce the diffuseness parameter at those angles. This method however, has only been applied so far to the problem of reproducing ambisonic microphone recordings (of low order) at higher orders - to good effect.

Binaural The binaural decoder follows a similar process as the regular HO-DirAC decoder, with the exception that in the binaural decoder, the "prototype" signals come from the linear binaural decoding of the original sound-field. Both the regular decoder and the binaural decoder are exceptional for decoding low order SH content at higher orders. This is perhaps one of the key benefits of parametric methods for ambisonic reproduction. Both decoders also count with an "Analysis order" described online:

> "When using Ambisonic signals derived from microphone arrays, the frequency ranges at which higher-order components can be obtained varies

[75] Found here.

> depending on the order; in these cases, the "Analysis-order" may be specified per frequency band to account for these imperfect input Ambisonic signals."

Upmixer The upmixer is described as nothing more than a wrapper for the HO-DirAC decoder. It first decodes the input to a virtual t-design, and then re-encodes the virtual loudspeaker signals in the target order. It uses the same analysis and re-synthesis approach as all other HO-DirAC effects, which means it extracts diffuseness and directional properties of lower order materials and applies these to higher order sound-fields using a regular distribution of beam-forming signals.

Other

CroPaC Binaural Decoder Cross-Pattern Coherence (CroPaC) [MDM19] decoder is based on the cross spectrum of spherical harmonic pairs of neighboring order n and $n+1$ (e.g. W and X). The cross spectrum between these two harmonics is calculated over a grid of look directions to generate a power map.

This power map makes up a pseudo-spectrum which can be used to visualize the DoA or extract the relevant signals using a beam-forming approach. Due to the nature of the algorithm, the power-map undergoes a half-wave rectification process which ensures the anti-symmetric look direction is ignored.

It is furthermore side-lobe suppressed in order to ensure that side-lobe peaks do not affect the DoA analysis. The directional signals, after extraction, are subtracted from the complete sound-field to extract the diffuse (e.g. residual) signal. This particular parametric approach was shown in subjective evaluations to exceed the performance of the spatial re-sampling decoder discussed earlier.

In many ways, this method is similar to the COMPASS approach, with the exception of the selected method for DoA analysis. As the author notes:

> "The proposed approach is inspired by the COMPASS method. However,

> along with the CroPaC post-filter, it also employs instantaneous source direction estimation and synthesises the output in a linear manner as much as possible; in order to improve the fidelity of the output signals."

HOSSIR The final element in this toolbox is the *Higher-order Spatial Impulse Response Rendering* (HO-SIRR) effect. The purpose of this plug-in is to use B-format impulse response to derive arbitrary loudspeaker array responses. In essence, one records an impulse response using a SMA and then HOSSIR creates a decoding matrix for the desired loudspeaker set-up, therefore deriving multiple impulse responses which can be loaded into other plug-ins for a virtual surround sound auralization based on specific room acoustics. However, the HOSSIR system extends this further by applying parametric analysis and post-processing to improve the fidelity of the auralization. Since this effect is still in beta (e.g. under development) we will not further discuss it in this chapter.

Conclusion The collective Sparta Plug-in Suite is certainly one of the most comprehensive and professional grade ambisonic tool-kits available. Its parametric approach to sound-scene analysis, transformation, and re-synthesis make it one of a kind. The tool-kit indubitably comprises one of the most sophisticated free and open source ambisonic solutions available today.

As we have seen, however, some of these effects seem to be achieving the same purpose as others in the suite. As a result, there is a surplus of effects in this suite that perhaps might not be necessary for most people. Additionally, the online documentation is slightly too complex for your average user.

One of the greatest strengths this tool-kit has, is the sheer number of associated publications. Most of these effects have accompanying papers, which make it relatively easy for developers to understand what the open source code is doing. However, from the standpoint of musicians and sound designers, these publications are too complicated. We believe the online documentation could do a better job at

conveying the differences of these methods, using language that is easy for musicians to understand.

We tested this tool-box using the same head-tracker as with the other toolboxes with 3OA[76] encoded stereo files. The sound quality was good but not noticeably better than simpler methods, in our opinion. It's possible that where the parametric methods really shine though, is in up-mixing FOA material recorded with ambisonic arrays, which we have not yet tested. The other slight difficulty is working with these effects is that the parametric analysis and re-synthesis methods tend to require considerable CPU power. We have not done a formal analysis, but we suspect the COMPASS, HO-DirAC and CroPaC methods significantly increase the CPU load.

1.3 Featured Project

Recently[77] the author submitted a Web-VR project, which was created with the help of two undergraduate students at UCSD[78], to the Audio Mostly conference[79]. The project was titled after the name of the song, written and produced by Tim Gmeiner, a current undergraduate student at UCSD's music program[80]. The present author has formerly been involved in organizing concerts featuring collaborations with graduate and undergraduate students[81]. Unfortunately, due to the pandemic, the possibility of using our facilities to create and perform spatial music was limited. In order to present this ambisonic music project, therefore, the author adopted a WebXR framework and a *JavaScript* (JS) solution for ambisonic binaural decoding.

The project is called "Pigments of Imagination", and it was mixed in Reaper using the aforementioned IEM plug-in suite. The WebVR framework *A-Frame* was

[76] Third order ambisonics.
[77] May 2021.
[78] Eito Murakami and Tim Gmeiner.
[79] Conference site can be found here. The piece was submitted as a musical contribution.
[80] Interdisciplinary Computing in the Arts Major(ICAM).
[81] Link.

used to create animations and provided the general layout of the graphic scene. One of benefits of WebXR solutions is that they can be experienced using personal computers, mobile devices and HMDs. This particular project was designed to be experienced using a Google Cardboard HMD, but can also be viewed on Chrome or Firefox on desktop computers. The HOA audio adapts to the user head rotation, which is controlled using a click and drag behavior in A-Frame.

In order to accommodate the network limitations of various users the audio was compressed using Audacity. Furthermore, since the HOA files - even after data compression - were relatively large, a FOA version and a binaural version were also created. These projects can be accessed here. The website also provides a link to a PDF with additional information regarding the project.

Here we include the program notes:

> "Pigments of Imagination is a popular music track touching on themes of imagination and space travel. The song is composed and produced by Timothy "Ill Poetic" Gmeiner and features vocals by both Ill Poetic and Nick Tolford, as well as flugelhorn by renowned jazz artist Stephanie Richards, and additional synthesizer parts by DJ/Producer King Britt. The stems from the song were used to create a special HOA mix, which you can hear online via the A-Frame project we created. In addition to the HOA, the A-frame project features custom animations that provide dynamism to the experience. During our experiments, we found that our HOA files did not load on most mobile devices, thus, alternate mixes in FOA and static binaural are also made available."

1.4 Conclusion

This chapter has dealt with spatial music tool-kits for the generation of spatial music, a genre in which space forms a primary aesthetic element of the composition. We discussed the requirements and consideration required in the design of these tool-kits, and outlined the internal design of various effects from three kits.

During the course of our investigation, we also proposed a number of possible improvements that could be applied to these designs. For example, w.r.t. **Ambix** we described to hyperbolic paraboloid effect, which has not yet been implemented, as well as extensions of this design, using arbitrary mathematical functions, and frequency-dependent warping. W.r.t the **IEM Plug-in Suite** we highlighted the lack of academic publications, which make it difficult for developers to contribute to these designs. Finally, with respects to **Sparta**, we noted the difficulty in adopting these effects due to their complexity, which could be alleviated by simpler musical language in the accompanying online documentation.

While these tool-kits are exceptionally well suited for fixed-media works, it remains to be seen the degree to which these can be adopted for live-performances and interactive pieces. Another issues is that a relatively good machine is required to work at high ambisonic orders, which might be necessary for large audiences. The graphical requirements of these effects should also be taken into account, since this too can become a potential bottleneck. Finally, popular free and open source computer music environments do not support VSTs yet without additional components, which means composers using these tools will need to do some additional work to include these in their pieces.

Chapter 2

Low-cost Microphone Array Design and Calibration

Question: *Toward the aim of making spatial audio and ambisonics more accessible, i.e. more affordable, comprehensible, usable and available to the average user, discuss the design and development considerations, along with possible trade-offs, of low-cost sound-field microphone designs and accompanying (open-source) processing software for signal encoding/decoding/formatting. How have your own experiments and implementation of an existing design informed your future work and goals? What are your impressions of the viability of this tool for musicians wanting to incorporate spatialized audio into their work?*

2.1 Introduction

During the 21st century we have seen a renewed interest in the field of virtual spatial acoustics, as investments in *Extended Reality* (XR) systems have exploded. With the proliferation of *Head Mounted Displays* (HMDs) as entertainment systems for personal use, a number of composers and sound engineers have expectedly began

researching the use of *Spherical Microphone Arrays* (SMA) towards musical ends.

Generally, although not exclusively, SMAs are associated with a technology known as ambisonics, developed in the 1970's by Michael Gerzon et al. The initial formulations of the system, denominated *First Order Ambisonics* (FOA), consisted of deriving three orthogonal figure-8 microphones to encode sound particle velocity in three-dimensional space. This spatial audio system is based on the theory of *spherical harmonics* (SHs), also used in various other fields including computer graphics and astronomy.

Spherical harmonics can be considered basis functions of 3D spherical phenomena, much the same way that simple trigonometric function can be considered bases for 2D phenomena. The well known *Discrete Fourier Transform* (DFT) is used to decompose complex sound events into a sum of sin and cos waves of varying amplitude, phase, and frequency. Much the same way, spherical harmonic theory aims at decomposing spherical sound phenomena using a combination spherical harmonics of increasing order.

A lot of work has been done in the last 20 years to extend FOA into *Higher Order Ambisonics* (HOA), providing a larger sweet spot[1], and improved directionality. Various microphone designs have been documented and evaluated in academic publications, however, very few have been documented to the extent that these can be recreated by people without extensive technical backgrounds. In the commercial sector, a few FOA and HOA solutions exist, however, most of these remain economically unviable for most artists.

The aim of this work is to present existing HOA microphone designs, considering principally those which favor low-cost and high-quality in the design process. The present author has already been involved in the process of constructing a low-cost FOA microphone using *Micro-Electronic Mechanical Sensors* (MEMS). Unlike traditional *Electret Condenser Microphones* (ECMs), MEMS capsules may come equipped

[1] Area within which audience members can sit and get a coherent sound image.

with *Analog-to-Digital-Converters* (ADCs), allowing direct integration with *Micro-Controller Units* (MCUs). The reduced cost of these MCUs over studio ADCs make these capsules ideal for anybody building a HOA array on a budget.

2.1.1 Sound-field Microphones

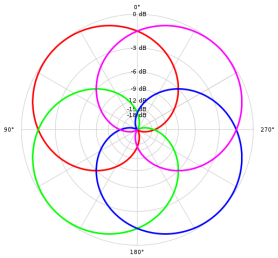
First Order Ambisonic (FOA) microphones were developed initially in the 70's by Michael Gerzon, from Oxford's department of Mathematics, and Peter Fellgett, from the University of Reading [Ele91], under the supervision of the National Research Development Corporation (NRDC) in England. Gerzon and Fellgett were indubitably inspired by Alan Blumlein's work in the field of stereophony.

Blumlein, another Englishman, was the pioneer of various audio technologies in the 70's. Chief among these was a sum-and-difference matrix which was capable of creating high quality stereo recordings. This technique, denominated *Mid-Side Stereo*, derives two cardioid patterns from the combination of one omni-directional microphone, closely positioned (e.g. coincidently positioned) with a figure-8 microphone. Gerzon and Felgett envisioned a 3D sound system which could be created by adding two additional figure-8 microphones, each representing a different axis in 3D space.

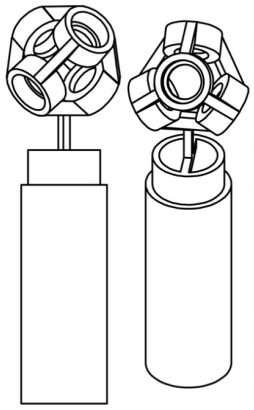
Instead of using three velocity microphones (e.g. figure-8 microphones), however, Gerzon and Fellgett designed a way to derive these three responses from four cardioid sensors mounted on the faces of a tetrahedron - using a sum-and-difference matrix akin to Blumlein's. Consider Figure 2.1; on the left, we can see the polar pattern of the FOA mic from a birds-eye view, on the right, we see a *Computer Assisted Design* (CAD) drawing of an actual FOA microphone[2].

As we can see from this top view, three orthogonal figure-8 microphones can be derived from the sum and difference of these different cardioid capsules. For example,

[2]Created by Yigal Kamel from "The Cooper Union for the Advancement of Science and Art" in SolidWorks.



(a) FOA Patterns (Top View) [Net]



(b) SolidWorks Drawing - FOA Microphone

Figure 2.1: FOA Polar Patterns (Top View) and CAD Drawing of FOA Mic

to derive the front-back microphone, we sum the front capsules and subtract the back capsules - the sides will cancel out and we will be left with a figure-8 response.

In addition to the three orthogonal[3] figure-8 microphone signals, an additional omni-direction channel, often denoted with the letter W, can be encoded by summing the four cardioid capsules - we call this a pressure signal. Once these four signals are recorded, they can be *decoded*[4] into any number of different reproduction systems. For example, the combination of signals can be used to yield a stereo representation with good mid/side separation[5].

Unfortunately, one of the main problems with ambisonics, or any surround sound reproduction system for that matter, is correct speaker placement. As Elen [Ele91] points out:

> "Most readers, I am sure, have visited numerous friends who keep their stereo loudspeakers in some very odd places – one channel behind the

[3]E.g. perpendicular.

[4]Decoding refers to using linear combinations of SHs to create speaker or headphone signals.

[5]The UHJ system, also known as "C-format", was an ambisonic system designed to be compatible with mono and stereo media.

> sofa and the other on top of the bookcase, for example. It's hard enough to get people to put two speakers in sensible places for stereo – what about four for surround sound?"

Luckily, many modern advancements in technology allow us today to record and reproduce ambisonic music seamlessly. Large speaker arrays can be substituted with binaural synthesis processes, reducing both the cost and complexity of ambisonic reproduction. Ambisonic microphones are a particularly attractive recording technology due to their simplicity compared to the encoding[6] individual sources. A good ambisonic microphone should only need to be gain-adjusted, for clear and realistic reproduction of spatial audio.

Ordering and Normalization

A common problem with sound-field microphone design can be the confusing naming and labeling of capsules, which is crucial to proper encoding. In simple FOA microphones, various authors have adopted the naming: FLU, FRD, BLD, and BRU (where F/B correspond to front/back, L/R correspond to left/right, and U/D correspond to up/down). For HOA, alternate naming systems must be developed and maintained throughout calibration and encoding. This can be especially complicated at higher orders, where designs might feature 64 capsules or more.

Harmony must also be maintained between the ambisonic signal *ordering* in the encoder and decoder. SH ordering refers to the indexing of SHs in the final B-format signal[7]. Meta-data can be used to determine if the sound-file is ordered according a particular standard, and if other relevant processing has been introduced to the signals which the decoder might need to account for[8].

[6] Encoding corresponds to transforming raw audio signals into spherical harmonics (SHs). The term encoding also refers to transforming raw HOA array signal into SHs.

[7] B-format is the name given to audio that has been encoded.

[8] For example, if *Near Field Compensation* (NFC) filters have been applied.

In addition to channel ordering, we must also account for the *normalization* system used in the encoding of the signals. The N3D normalization format, results in a set of orthonormal basis vectors, which have useful mathematical properties. However, the preferred normalization today is SN3D, which forces SH above W to not exceed the level of the 0th ambisonic order harmonic [Kro14b].

There are various standards for *ordering* of ambisonic channels and their *normalization*. The most common are the *Furse-Malhalm* (FuMa) [CRPGT+19] ordering scheme, and the *ACN*[9] ordering scheme. Table 2.1.1 shows these two ordering schemes up to 3OA[10].

Ambi Ordering (3OA)		
	FuMa	ACN
W	0	0
Y	2	1
Z	3	2
X	1	3
V	8	4
T	6	5
R	4	6
S	5	7
U	7	8
Q	15	9
O	13	10
M	11	11
K	9	12
L	10	13
N	12	14
P	14	15

At this point, FuMa, and the *maxN* normalization scheme it used are mostly legacy standards[11]. Developers are instead focused on the Ambix format [NZDS11],

[9] Which stands for Ambisonic Channel Number.

[10] A third, more obscure ordering called SID by Daniel [DM04], was left out. 3OA stands for third order ambisonic.

[11] E.g. they are no longer used.

which consists of the ACN ordering scheme and the *Schmidt semi-normalized* harmonics format (SN3D) for normalizing. Equation 2.1 shows the SN3D normalization, according to Gorzel's 2019 paper on the implementation of Resonance[12][GAK+19].

$$N_{n_{SN3D}}^{|m|} = \sqrt{(2 - \delta_m)\,\frac{(n - |m|)!}{(n + |m|)!}} \tag{2.1}$$

where $\delta_m = 1$ for $m = 0$, and 0 otherwise. In the Ambix specification paper [NZDS11], one should note there is a $1/4\pi$ term that is ignored in our formula. Ultimately, we can treat this term as a scalar, since it is independent of *ambisonic order* (n), or *ambisonic degree* (m) - thus, it can justifiably be omitted.

Another normalization standard is the N3D normalization scheme. The aforementioned N3D normalization standard is defined by the following equation from [PPQ16]:

$$N_{n_{N3D}}^{|m|} = \sqrt{(2n + 1)\frac{(n - |m|)!}{(n + |m|)!}}. \tag{2.2}$$

This equation is included here simply for completeness. Most authors and companies use SN3D as the default normalization today in ambisonic development. Conversion between the two is also discussed in [PPQ16].

Ambisonic Degree and Order

In mathematics and physics, the order (m) and degree (l) of spherical harmonics is the opposite of the ambisonic order (n) and ambisonic degree (m). In other words, the ambisonic order is the mathematical degree of the spherical harmonic, and the ambisonic degree is the mathematical order of the spherical harmonic[13].

Figure 2.2 shows a classic representation of spherical harmonics of increasing *degree* (l), where the top harmonic has a degree of 0 and the bottom harmonic has

[12]Resonance is an open-source Google product for ambisonic manipulation.
[13]We are not really sure how this came to be, it simply is.

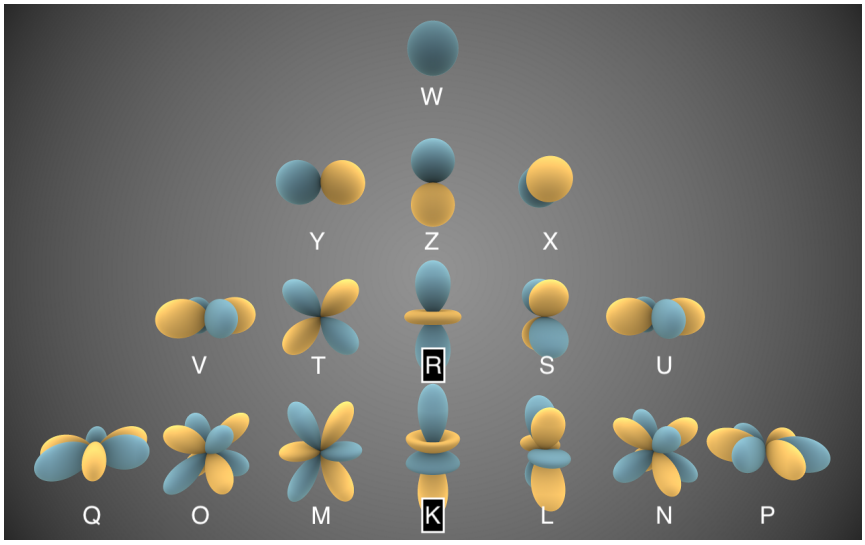


Figure 2.2: Spherical Harmonics [Ini14]

a degree of L. The *order* (m), in the mathematical sense, spans from negative to positive values and includes 0. The image was modified to show the letters associated with each harmonic, according to ACN ordering.

To avoid confusion we will use the term *ambisonic order* rather than order, along with the corresponding notation[14]. For example, in Figure 2.2, the ambisonic order (n) increases from 0 to N, where the top harmonic has an ambisonic order of 0, and the bottom harmonic has an ambisonic order of N. The ambisonic degree (m) spans negative and positive values, and includes 0 at the center column.

Coordinate System

An important consideration for ambisonic research is the 3D coordinate system. In spatial audio research the coordinate system used is unfortunately not always consistent. This is especially true when we switch over to discussions of different

[14]Unless mathematical order is implied.

spatial audio methods such as VBAP[15] or OBA[16]. For example, in SpatDIF [PLS12], the azimuth angle increases clockwise, and the X, and Y, axes are switched relative to the Ambix convention.

In this text, when talking about ambisonics, we will use ϕ to denote the azimuth angle, increasing in the anti-clockwise direction, and θ to denote the elevation angle, increasing upwards to 90°. This has become the standard way of writing ambisonic equations. The X, Y, and Z axes point towards the front, left and up respectively. The conversion between spherical and Cartesian coordinates is defined as [GAK+19]:

$$\begin{aligned} x &= \|\vec{r}\| \cos(\phi) \cos(\theta) \\ y &= \|\vec{r}\| \sin(\phi) \cos(\theta) \\ z &= \|\vec{r}\| \sin(\theta) \end{aligned} \tag{2.3}$$

where $\|\vec{r}\|$ is the norm of the vector r, which is the distance of our source from the origin of our sphere[17]. In Figure 2.3, point P can be defined by the Cartesian vector with variables x, y, z, or spherical coordinate vector with elements ϕ and θ.

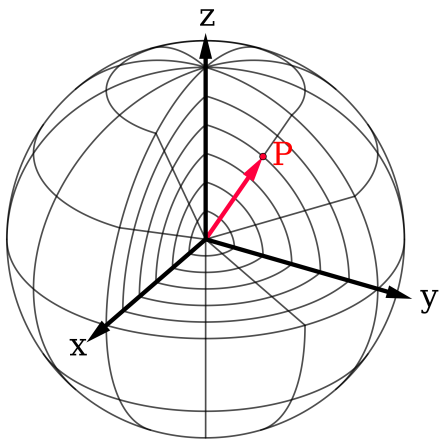


Figure 2.3: Spherical Coordinate System [Sha]

[15] Vector Based Amplitude Panning.
[16] Object Based Audio
[17] The norm is the length of the vector, which is the square root of the sum of the squares.

Real Valued SH

As an introduction to the topic of ambisonics, we also present equation 2.4, which is used to calculate real-valued *spherical harmonics*.

$$Y_n^m(\phi,\theta) = N_n^{|m|} P_n^{|m|}(\sin(\theta)) \begin{cases} \cos(|m|\phi) & \text{if } m \geq 0 \\ \sin(|m|\phi) & \text{if } m < 0 \end{cases} \tag{2.4}$$

$N_n^{|m|}$ corresponds to the normalization term discussed earlier. $P_n^{|m|}$ are the *associated Legendre functions*, in this case evaluated for $\sin(\theta)$. Depending on the ambisonic degree, a suitable right side argument is chosen to determine our solution. These functions are used to encode mono signals into SH representations, and also necessary to encode/decode signals recorded by SMAs.

Associated Legendre Functions

The associated Legendre functions are solutions to differential equations used to describe wave functions in the spherical domain. Most authors use numerical libraries, such as Julia or MATLAB, to calculate these. However, if we wanted to calculate these ourselves, we could also use the recurrence relation and a first few given terms to manually calculate these for any l and m we desire[18].

Equation 2.5 shows the recurrence equation including the *Condon-Shortley* phase term [Wei].

$$(l-m)P_l^m(x) = x(2l-1)P_{l-1}^m(x) - (l+m-1)P_{l-2}^m(x) \tag{2.5}$$

In our case, we can calculate $P_n^{|m|}(\sin(\theta))$ by using the initial conditions:

[18]Note here the use of l, m rather than ambisonic order and ambisonic degree.

$$\begin{aligned} P_0^0(\sin\theta) &= 1 \\ P_1^0(\sin\theta) &= \sin\theta \\ P_1^1(\sin\theta) &= -\cos\theta \end{aligned} \tag{2.6}$$

[Wei] shows these three initial conditions for $x = cos(\theta)$. We can use the general equations for $P_n^{|m|}(x)$ to find the first three solutions to $P_n^{|m|}(\sin(\theta))$. The Pythagorean identity $\sin^2(\theta) + \cos^2(\theta) = 1$ shows that $P_1^1(\sin\theta) = -\cos\theta$. Equation 2.7 from [Wei] shows the general formula for associated Legendre polynomials for the first three terms, allowing us to set-up our recurrence relation. E.g.

$$\begin{aligned} P_0^0(x) &= 1 \\ P_1^0(x) &= x \\ P_1^1(x) &= -\left(1 - x^2\right)^{1/2} \end{aligned} \tag{2.7}$$

Condon-Shortley Phase One final consideration pertains to the polarity of the spherical harmonic, which is determined by the *Condon-Shortley* phase term - defined as $(-1)^m$, where m is the mathematical order[19]. Equations 2.5 and 2.7 both include this Condon-Shortley phase term. Nachbar [NZDS11], who authored the Ambix specification, suggests removing the Condon-Shortley term in ambisonic software development - as it only really simplifies matters in the field of mechanics.

Our third initial condition from Equation 2.6 still satisfies the Pythagorean theorem if the Condon-Shortley phase gets removed[20]. To prove this set $P_1^1(x) = \pm\left(1 - x^2\right)^{1/2}$ equal to $\pm cos(\theta)$, and set $x = sin(\theta)$. The sign of $P_1^1(x) = \pm\left(1 - x^2\right)^{1/2}$ is irrelevant, since squaring both sides will force both arguments to be positive.

In our numerical computing library, whichever we chose, we will have to make sure to compensate for this $(-1)^m$ term. Alternatively, we can choose to write our own function that excludes the term. In order to compensate for the extra $(-1)^m$,

[19] Note that m always corresponds to the orientation of the harmonic.

[20] The phase change only occurs for odd harmonics, excluding the first harmonic, which is a special case - since $(-1)^0 = 1$.

we simply add another $(-1)^m$ to our recurrence equation[21]. This will make all our harmonics of the same polarity, regardless of the value of m.

2.2 Coincident Microphone Arrays

Of primary importance to our work is the development and optimization of coincident microphone arrays. Coincident microphone arrays come in various shape and sizes. Some are integrated into camera systems, while others are solely audio devices. The predominant characteristic of these instruments is that proximity between sensors is minimized in order to improve localization estimations and operating frequency of *virtual microphones* created by combinations of signals. There are three main types of coincident microphone arrays we were able to discern from the literature: *spherical microphone arrays* (SMA), studio microphone arrays, and planar microphone arrays.

While most planar microphone arrays are not meant for high-quality studio recordings, we will nonetheless cover some of these designs here, since many use similar components as the proposed HOA system being developed by the author. Studio microphone arrays are indubitably interesting, but outside the scope of this chapter. The main drawback of most studio microphone arrays is the high cost associated with the designs. Studio mic array designs include the double-MSZ[22], or the native B-format array, which lead to the modern tetrahedral design.

2.2.1 Planar Microphone Arrays

Planar microphone arrays are a subset of coincident microphone arrays which attempt to sample a sound-field using either a linear (e.g. one-dimensional), or circular (e.g. two-dimensional) array of sensors. Most planar microphone arrays,

[21] Or use an `abs()` function, whichever is faster.
[22] By Paul Geluso.

although not all, fall under the category of arrays for *noise suppression*; in they are not designed for multi-channel recording. Instead these devices output a single audio channel that is the sum of various sensors delayed and amplified based on the desired direction(s) of sound capture. This technique, known as *beam-forming*, is also applicable to spherical arrays.

Backman's Designs Backman [Bac06b] provides some insight into planar microphone arrays and their relationship to sound-field microphones. The author in that publication relies on *Micro-Electrical Mechanical Systems* (MEMS) capsules for his designs, much like in the present author's own work. Backman suggests using multiple transducers per sampling node to improve *signal-to-noise ratio* (SNR) of the system and provide improved polar pattern control.

In a second paper [Bac06a], Backman expanded upon his theoretical framework and proposed a 5.1 planar array based on MEMS microphones. Unfortunately, little information was presented regarding a 3D audio capture system; the designs described were predominantly for 2D audio[23]. While the author discusses 3D design, we were not able to find schematics or models to construct these arrays.

Chen et al. Chen et al. [CAZ15] provide a more thorough review of planar arrays. This paper describes a 2D planar array designed with first order microphones - such as figure-8 or cardioid microphones - which can suitably sample vertical elements of the sound-field. The benefit of planar arrays is that for certain applications, spherical arrays simply might not be feasible - such as in cellphone or tablet designs. One noteworthy process implemented in that publication is the use of simulated responses to validate the design prior to implementation - this is recurring theme in microphone design literature.

[23] 3D audio should include height (*periphonic*), 2D audio is captured on just the horizontal plane (*pantophonic*).

The main problem we found with this design was the limited bandwidth, which had a maximum frequency of 1000Hz due to spatial aliasing. For high-quality audio purposes, we seek instead to operate on the audible frequency range. The authors make no mention of correction filters to address this problem. Secondly, in order to sample vertical harmonics, the number of transducers had to be increased from nine, the minimum number required for 2OA[24], to 16, which increases the cost/complexity of the final design.

Meyer and Elko Meyer and Elko of *mh acoustics*, the company responsible for the *em32 Eigenmike*[25], described a second order augmented circular microphone in a 2008 publication [ME08]. In their paper, Meyer et al. described using a single central sensor to provide vertical control of the desired *beam-pattern*. The beam-patterns are controlled by adjusting the weights in the linear combination of spatial harmonic signals - also referred to as *eigenbeams* or *eigenmodes*.

Their design has 7 capsules "flush mounted into the top surface of a puck-like housing". There are 6 capsules along the diameter of the puck, with a radius of 3cm, along with one central capsule. This design seems suitably promising for play-back over dome-like (e.g. hemispherical) loudspeaker systems; very few sound systems around the world have below-listener speakers. Unfortunately, the design might be unsatisfactory for applications which use binaural synthesis featuring HRTFs below the listeners' ear level, since that part of the sound-field would not be properly reconstructed. Two other details stand out about the design:

1. The microphones are described as pointing upwards rather than outwards. Mounting them on the side of the puck would have likely improved the performance of the array. This was likely due to mechanical simplicity.

[24] Second order ambisonics.
[25] mh acoustics

2. The author never specify what capsules are used and there is little information regarding how to recreate the experiment[26].

The authors remark that omni-directional microphones tend to be better matched than directional ones. This becomes important in HOA arrays were ideally we want all capsules to have identical responses.

There are two metrics mentioned by Meyer et al. we should also discuss: *Directivity Index* (DI), and, *White Noise Gain* (WNG). The DI is the difference in *decibels* between the *Sound Pressure Level* (SPL) in a given direction and the average SPL from an omni-directional source. DI is provided by [Lon14] and reduces to:

$$\mathrm{D}(\theta,\phi) = \mathrm{L_p}(\theta,\phi) - \overline{\mathrm{L}}_\mathrm{p}$$

where:

$\mathrm{D}(\theta,\phi)$ = directivity index (*gain*) for a given direction (dB),

$\mathrm{L_p}(\theta,\phi)$ = sound pressure level for a given direction (dB),

$\overline{\mathrm{L}}_\mathrm{p}$ = sound pressure level averaged over all angles (dB), and,

(θ,ϕ) = some specified direction.

WNG is another quality metric of beam-forming algorithms. It is defined as the beam-former's ability to suppress spatially uncorrelated noise. Note that self-noise and external white noise may be statistically indistinguishable, so WNG constraints should consider self-noise as well. Mismatch between sensor characteristics and position errors can also result in spatially uncorrelated white noise [MSK09]. Both of these measures are applicable to SMA design, and can be used to analyze the quality of the final design.

[26]This is likely because *mh acoustics* is a for profit entity.

Middlicott et al. Middlicott et al. [MW19] recently published a paper regarding HOA microphone calibration approaches, in which they described the fabrication of a planar[27] ambisonic array. The main theme of the paper is not the fabrication of the microphone itself, but rather evaluating different calibration approaches for A-format signals in HOA. The A-format signals are subsequently encoded to determine how much the pre-encoding calibration improves the response of spherical harmonics.

A number of important procedures are elucidated by the authors. Like many other authors involved in this work, the first consideration is the room acoustics. For this experiment a *hemi-anechoic* room was used (e.g. not perfectly anechoic). As a result, some processing will need to be applied to the measured IRs. Lopez-Lezcano [LL19] describes windowing applied to the measurements in order to reduce reflection effects from room surfaces. This entails: measuring the closes surface to the microphone, calculating in samples that amount of time, trimming the IR, and using a windowing function to remove discontinuities - which might cause FFT artifacts. This process will be covered in more detail in Section 2.2.2.

The authors noted that only half of the impulses responses were necessary for analysis given the symmetry of polar patterns[28]. This effectively cuts the measurement time in half. The authors also noted the necessity to compensate for the speaker response. For this, a flat frequency response microphone is used, and, as is common, the Nelson/Kirkeby inversion approach is implemented. One should also trim and window this impulse response to compensate for the room acoustics. Equation 2.8 shows the Nelson/Kirkeby algorithm, used to generate the inverse filter (e.g. the compensation filter), including a frequency-dependent regularization parameter.

$$H_{inv}(\mathrm{k}) = \frac{H_{\mathrm{Target}}\,(k) \cdot \mathrm{Conj}(H(k))}{\mathrm{Conj}(H(k)) \cdot H(k) + \epsilon(k)} \tag{2.8}$$

[27] It is cylindrical but with an open design, we call it planar because all capsules lie on a single plane.

[28] The same principle was used in our own work [Zal19]. Unfortunately, assuming symmetry can lead to errors.

where:

H_{inv}(k) is the inverted filter,

H_{Target} (k) is the ideal frequency response,

$H(k)$ is the measured amplitude response, and

$\epsilon(k)$ is a regularization parameter.

The value, k is the frequency index (e.g. the bin) and the $conj()$ operator corresponds to the *complex conjugate*. Without the regularization parameter, the inversion of the filter is idealized, but this can lead to over-emphasis in certain regions of the spectra. The authors use a vector with various regularization parameters depending on the region of interest for A-format calibration, but seem to suggest using no regularization for the speaker inversion. The resultant inverse filter is convolved with each A-format IR in order to negate the effect of the speaker. In this case the target filter would most likely be the FFT of a delta function, which is perfectly flat. Another approach is using software such as DRC[29] to generate such a filter.

The remainder of the paper discusses four different approaches used for calibrating the capsules, which produce our A-format signals. These four approaches are called:

1. Calibration by 1/3rd Octave Average Gain Matching,
2. Calibration to a Specific Capsules On-Axis Frequency Response,
3. Calibration to a Flat Frequency Response, and
4. Calibration by Diffuse Field Equalization.

The first method involves using just one on-axis response for each capsule, and splitting the FFT into 1/3rd octaves[30]. The average magnitude is measured for all

[29]Digital Room Correction.

[30]In electronics, an octave is a doubling of frequency. For example, between 40 and 80Hz there is exactly one octave.

these sub-bands and then the lowest value is used to attenuate all other microphones to that same level.

The second method involves selecting the capsule with the best response by visual inspection and subsequently trying to match all other capsules using Equation 2.8. In this method, once more, a single IR for each capsule is required. Note that this method means user input is required (e.g. this process cannot be automated without some modifications). One could alternatively calculate the deviation from a flat frequency response, instead of selecting the target response by visual inspection.

The third method is very similar to the second, however, rather than using an on-axis response from one of the capsules, all the capsules are calibrated using a perfectly flat frequency response as the target. The frequency response of the *Dirac delta function* (δ) function serves as this target. Much like with the second method, frequency-dependent regularization is employed (e.g. the $\epsilon(k)$ parameter changes over the frequency range).

The last method, called Diffuse Field Equalization, is the most time consuming of all, as it requires the full range of IRs. Much like with the other methods, each capsules gets a customized/individual calibration filter. However, to find our target filter, we take all measurements (e.g. all directions), for each capsule, and average them. In this case, only horizontal measurements were used. Middlicott et al. provide the following equation to generate the *Diffuse Field Response* (DFR).

$$\mathrm{DFR(c)} = \sqrt{\frac{1}{D}\sum_{d=1}^{D}\left|FFT_{(c,d,k)}\right|^2} \tag{2.9}$$

where:

DFR(c) is the Diffuse Field Response for capsule c,

d is the measurement position index (1 to D), and,

k is the frequency bin.

As we can see, this equation is closely related to the RMS[31], although, in this case, we use these values to derive a filter over specific frequency bins. With these filters acquired, the authors use the frequency-dependent regularized inverse filter technique, much like with methods 2 and 3, with the $DFR(c)$ filters as targets. We should note that the author does not make any mention of the phase of the inverse filters, only the magnitude is considered[32].

After the four methods were implemented, the author undertook an evaluation of these different A-format calibration approaches by examining the spherical harmonic polar plots. Before doing so, however, they also generated simulated B-format polar plots using perfect cardioid responses. Even then, the simulated harmonics differed greatly from the ideal harmonic due to the distance between transducers. The authors' particular design featured a 25mm radius and 72° capsule spacing. The calculated spatial aliasing frequency was found to be 4.3kHz using the following equation[33]:

$$\text{Aliasing Frequency } = \frac{Nc}{2\pi r} \tag{2.10}$$

where N is the ambisonic order (2OA in this case)[34], c is the speed of sound, and r is the array radius. The authors also plot the uncalibrated B-format polar responses for comparison with simulated, ideal, and calibrated responses. The ideal responses correspond to perfect coincidence and response, which is impossible in real-world conditions.

In order to encode the raw A-format signals captured directly from the mic into the SH domain the, real-valued SHs are evaluated at the capsule angles according to

[31] Root Mean Square.

[32] In [LL18] filters are modified for minimum-phase.

[33] This theoretical frequency has been found to be imperfect in other publications. Visual inspection can be used to determine transition frequency [LL19].

[34] For 3D sound capture of 2OA we'd generally need 9 capsules. This design is for horizontal 2OA (or cylindrical harmonics only). The general formula for this is $(2N + 1)$ capsules are required, rather than $(N + 1)^2$.

the array geometry. Since this particular array is uniform, the frequency-independent SH encoding matrix can be applied as is to generate the B-format signals.

Figure 2.4 shows an example using FOA SHs. The matrix of SHs contains four rows corresponding to four capsule positions, and four columns corresponding to a particular ambisonic order and degree. The capsule's signals are multiplied by each element in the corresponding row, and summed to generate the desired signals. Alternatively, one can perform point-wise multiplication using a vector of capsule signals with each column in the matrix, followed by integration.

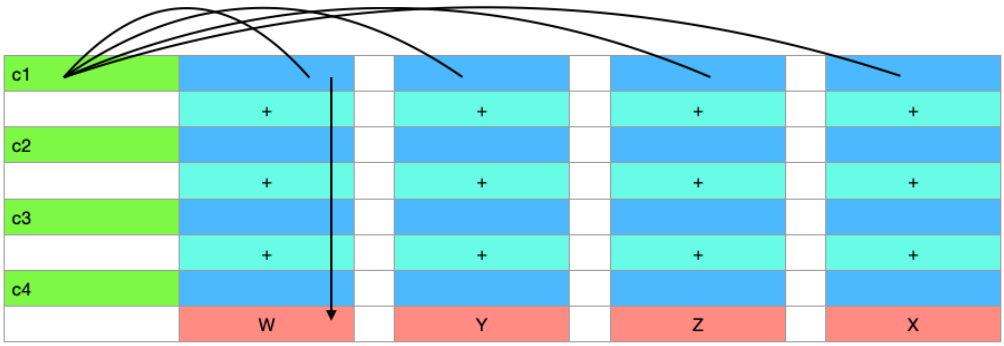


Figure 2.4: FOA Mic Encoding Diagram

Middlicott et al. concluded that the DFR method (method 4) and the average gain matching (method 1) yielded the best results. Considering that method 1 requires far fewer measurements, and yields similar results, it would be interesting to evaluate the time savings this could result in - especially in an automated system, which is ultimately desired.

Unfortunately, in this publication, all the measurements were taken only on the horizontal plane, and there was no subjective evaluation of the systems proposed; a subjective evaluation of methods 1 and 4 could help decide which method is superior, if any. Finally, the authors noted the need for post-filtering after encoding to further improve the integrity of the spherical harmonics at all frequencies. The figures in the publication also reveal that despite the absence of radial filters, such as those

described in [BPSW11], proper SH responses were achievable at multiple frequencies.

This paper offers great insight into the calibration of capsules for SMAs. Unfortunately, the design of the microphone could not be found. The capsules used were the JLI-140A-T, which only cost about $5 per capsule. Although these capsules have many attractive features, for HOA arrays, we would likely need a costly ADC to record the analog output of these cardioid mics. In addition, the current design is only capable of sampling SH in 2D, and the parts are not optimized for low-cost 3d printing.

2.2.2 Spherical Microphone Arrays

Spherical microphone arrays (SMAs) are particularly interesting, since they represent a simple way of capturing *periphonic* (e.g. 3D) sound in both a studio setting as well as out in nature. In contrast to other recording techniques in which musicians'[35] coordinates need to be known in order to re-create an adequate sound-field, spherical microphone arrays inherently capture the spatial information of the sound scene. These recordings are then encoded into the SH domain, using one of several techniques, for reproduction over an arbitrary loudspeaker configuration, or binaural system.

SMAs can be separated into three types of arrays: rigid, hollow, and tiered. Tiered arrays are arrays in which various radii are superimposed in order to capture different regions of interest of the sound-field [DRS15]. In general, these systems have gotten less attention due to the difficulty of integrating these models with camera systems. Rigid and hollow models are more popular, and corresponding radial filters exist for both. As their names suggest, a rigid spherical microphone array has no cavities exposed to the outside of the array, while the hollow designs have open spaces over which sound may propagate. Rigid microphones can further be separated into

[35] Or stationary microphones'.

those with protruding capsules, and those which have capsules flush-mounted on the surface of the sphere. An example of a hollow design is shown in Figure 2.5.

Figure 2.5: Octathingy Version 1 [LL19]

Over the years, several authors, including the present author, have proposed spherical microphone array designs - generally for the purpose of recording and reproducing ambisonic signals. The degree to which these designs have been documented and made open to the public varies from author to author. The following projects describe some attempts made over the years to create open-source spherical microphone arrays. In the commercial sector, there are also designs by companies such as: Sennheiser, Zoom, and Zylia.

Modular Spherical Microphone Array (MSMA) Gonzáles et al. [GPL18] described in 2018 a design the authors defined as modular, given the system's ability to change radius on-the-fly. Their design can be considered both multi-tiered, or rigid, since in certain configurations it can have more than one operating radius. The design has a rigid body, but capsules protrude from the main sphere by varying amounts, depending on the attachment selected. The project site can be found here[36].

[36] Accessed: Feb. 2021.

In the cited publication, the authors compared the performance of their design in three configurations: small radius, large radius, and mixed-radius.

Some of the key features of this design, in addition to the modularity aspect of the microphone, is the use of FOSS for the *Computer Assisted Design* (CAD)[37] and use of MEMS microphones, in lieu of ECMs. Unfortunately, the code used to generate the CAD models does not seem to be accessible, however, the files for manufacturing the *Printed Circuit Boards* (PCBs), and the rendered 3D designs can be downloaded and modified.

In order to validate their designs, the authors used a simulation approach in which they compared a simulation of their system to the ideal SH representation. The "Spherical Array Processing" library by Archontis Politis, publicly downloadable from Github[38], and explained in his doctoral dissertation [P+16], was used for this assessment. The spatial correlation measured was given by the following equation:

$$R_{|S^{sim}|,|S^{SH}|} = \frac{\left\langle S^{sim}, S^{SH} \right\rangle}{\|S^{sim}\| \cdot \|S^{SH}\|}. \tag{2.11}$$

The top part of the fraction involves taking the dot product of the simulated and ideal spherical harmonics. The bottom part involves taking the norm of each vector and then calculating their product - these are both classic operations in linear algebra. The spatial correlation is calculated over the range of audible frequencies for each ambisonic order. For the three given arrangements (e.g small radius, big radius, and mixed radius), it was found that the smaller radius provided the best spatial correlation. Unfortunately, these simulations can only give us a prediction of the performance of this system. Ideally, the authors would have also measured the response of the microphone in anechoic conditions, and performed some listening experiments.

[37] The authors used OpenSCAD, which is distinct from other systems in that the user programs the design using code, rather than manual manipulations.

[38] Link to repository.

Notwithstanding, the design remains promising given that the parts used for its manufacturing can be printed with low-cost 3D printers. This *Modular Spherical Microphone Array* (MSMA) also features MEMS capsules, making it more affordable to reproduce than other designs. Unfortunately, the 19 analog capsules require a rather costly interface in order to record the signals. MEMS microphones with internal ADCs might provide a cheaper alternative than the proposed analog MEMS devices. In addition, the selected capsules in the MSMA design do not offer the best SNR available for analog MEMS capsules.

Higher Order Spherical Mic Array (HOSMA) TH Köln[39] has published several papers describing the development of a rigid SMA. In [MDLP20] Moschner et al. described their latest efforts to create a 7th Order Ambisonic (7OA) microphone with a total of 64 capsules. As the authors noted, spherical harmonic processing in VST plug-ins is limited to 64 channels (i.e. SPARTA or IEM suite)[40]. There are currently no 7OA microphones on the market, the highest order commercial system is the Eigenmike by *mh acoustics* which is of 4th order.

Similar to Gonzáles et al. [GPL18], this microphone system has an inner sphere to which various tubes are attached, however, in the case of the design by Moschner et al., the 64 channels are all comprised of ECM capsules instead of MEMS microphones, and there is an additional rigid baffle which changes the acoustics of the array[41].

HOSMA uses the same capsules as the commercial FOA microphone by Sennheiser called Ambeo VR Mic[42]. The CAD modeling is done in Blender and the analysis is done using the SOFiA toolbox[43].

[39] TH Köln - University of Applied Sciences, is an institute of higher education located in Cologne, Germany.

[40] The ambiX library also contains a multi-channel convolution plug-in for this purpose.

[41] Rather than a rigid array with protruding capsules, it becomes a rigid array with flush mounted capsules.

[42] It is unclear from our search if these capsules are commercially available, this makes the experiment difficult to replicate.

[43] Link to HOSMA repository (accessed: Feb 26, 2021)

Another similarity with the Gonzalez et al. publication, is that partially automated modeling is possible using Blender and Python scripts. Automated modeling is done in [GPL18] and [LL19] using the OpenSCAD modeling software, which uses a strictly scripting-only approach to 3D modeling. This means other researchers can easily re-purpose the design for systems with: fewer microphones, smaller radii, or different capsules[44].

The HOSMA design is based on a Fliege Grid [FM99], created by Jörg Fliege from the University of Southampton's School of Mathematics. The integration formulae for this grid can be found here. There are coordinates and weight for spheres with 4, 9, 16, 25,..., 900 nodes. According to Moschner et al. [MDLP20], this sampling grid allows an "an optimal spatial resolution can be archived with a minimum number of microphones." The Fliege grid is particularly useful for ambisonics because, as we can see, the number of points on the sphere follow the $(N+1)^2$ requirements for ambisonic spherical sampling.

According to [Arc]: "the Fliege-Maier nodes are another example of nearly-uniform arrangements that along with their respective integration weights can be used for direct integration through summation." This is one of four "special" sampling arrangements[45], which allow for direct integration. When the function is sampled using non-uniform nodes, then a least squares solutions is necessary to encode the array - such as in [LL19].

The parts for this project were 3D printed using *Fused Filament Fabrication* (FFF)[46] via low-cost 3D printers. This is one of the considerations also addressed by [LL19] and [GPL18]. Two other, more robust, 3D printing technologies are: SLS and SLA. SLA is used for prototyping; these printers cost around $500. SLS is used for final products, and uses stronger materials. SLS printers cost around $5,000 dollars, which not all institutions might have access to. FFF printers vary in price

[44] The authors note the process is not fully automated in HOSMA.
[45] Including Gauss-Legendre quadratures, Lebedev grids, and t-designs.
[46] Also known as Fused Deposition Modeling.

depending on the quality of the print - a cheap one can cost $300. FFF, therefore, is the most accessible type of 3D printing. Unfortunately, it also means designs must be geometrically simpler, and will not be as strong[47].

The HOSMA design uses the Sennheiser KE14 ECMs and PCBs with a dedicated voltage divider and converter, which outputs a balanced signal to a standard XLR-pin connector. Additionally, for improved EMI-shielding[48], copper tape and graphite spray are applied to the tubes enclosing the PCBs. In [DLAP19] by Dziwis et al. (e.g. the first publication associated with this project), the EMI-shielding was measured to quantify any improvements. It was shown that the 50Hz hum produced by a voltage outlet was mitigated by using these methods. The authors also used a conductive PLA[49] 3D printing material to further improve EMI-shielding.

The most expensive part of this build is unfortunately the recording interface. HOSMA requires 8 microphone amplifiers, each converting eight A-format signals to the ADAT format. Another interface converts eight ADAT streams to the MADI format for recording. This makes the ultimate design, unfortunately, well outside the price range of independent musicians.

The final part of the paper describes the evaluation of the system. The array was measured via impulse response methods in an anechoic chamber at TH Köln. For comparison, Moschner et al. also simulated a free-field plane wave impinging upon a virtual array with the same Fliege grid. For this simulation the SOFiA (Sound Field Analysis) Matlab toolbox was employed [BPSW11]. The simulation and real measurement were well-matched for the frequency of 3kHz. This design is expected to show spatial aliasing artifacts above 3.2kHz. In this paper, the simulation and recorded IR are of a single direction.

The authors employ a *Plane Wave Decomposition* (PWD) method to analyze

[47] After prototyping is complete, one can also send their designs to a 3D printing service, which can return stronger prints at relatively low prices.

[48] Electro-Magnetic Interference.

[49] Polylactic Acid.

the results. According to [PPQ16] in spherical beam-forming literature PWD refers to higher-order hyper-cardioids used to "maximize the directivity factor for a given order"[50].

Radial filters in this publication were limited to 0dB. These are used to compensate for any scattering occurring from the surface of the sphere and also account for the directionality of the sensors. These radial filters are implemented in the SOFiA toolbox.

As noted in Lösler and Zotter [LZ15], simply using a frequency-independent matrix to encode A-format signals into the SH domain does not yield proper ambisonic recordings. This is because "low-frequency signals predominantly map to a non-directional pattern on the surface of the microphone array". This, in turn, causes higher-order signals to require amplification. Holographic/radial filters, are therefore required to equalize the B-format signals.

Radial filters are designed based on the nature of the microphone array (open/-rigid), and the individual capsules' responses (omni/cardioid). The formulas themselves are based on spherical Bessel and Hankel functions, both of which describe sound propagation in a frequency-dependent manner. According to McCormack et al. [MDMF+18]:

> "The modal coefficients[51] take into account whether the array construction is open or rigid and the directivity of the sensors (cardioid, dipole or omnidirectional). In the case of a rigid baffle construction, these modal coefficients can also take into consideration the frequency-dependent acoustical admittance of the surface of the array and also the distance between the surface of the array and the sensors; in order to cater for cases in which the sensors protrude from the surface of the array."

In order to derive these filters, we first need to define and calculate the spherical Bessel functions of the first and second kind. All of these are related to Bessel

[50] This means essentially that the analysis is that of a virtual microphone, not the full sound-field.
[51] Of the radial filters.

functions, and are all solutions to the same differential equation [Teu07]. They can also all be defined in terms of Bessel functions of the first kind, which are often called cylinder functions. These are described in Appendix A.

Using these spherical Bessel functions various radial filters can be implemented, based on the configuration of the SMA. Bernschutz et al. [BPSW11][52] provide good formulations for some of the filters for various different configurations. McCormack et al. [MDMF+18] provide more information regarding the inversion methods examined in the literature, as the actual filters we will need to use are the inverse of these functions - which require regularization to limit sensor noise amplification.

The HOSMA analysis revealed that the array performs similarly to the simulation at some frequencies but deviates substantially at others. Notably, the array does not perform well at 500Hz, well below the aliasing frequency of 3.2kHz. This could perhaps be improved with some equalization of the B-format signals. The other two reported measurements at 3kHz and 7kHz appear to match the simulation.

While this design is promising given the very high order of the array, and high quality of the components used, it is not suitable for independent musicians or sound engineers working on a budget. Additionally, certain parts do not seem to be available without a partnership with the manufacturer, which makes this project unreproducible for most people.

Nonetheless, the Fliege grid design does offer a great HOA design that could be modified and used with other capsules; one could swap these out for MEMS to create a cheap version of this array. The license of the project, and use of open-source tools, such as Blender and Python, make this possible. The SOFiA toolbox has recently been partially ported to Python [HA17], which means one does not need a MATLAB license to perform these simulations.

[52] See Section 3.2 of [BPSW11].

Spherical Harmonic EAR (SpHEAR) Fernando Lopez-Lezcano, from CCRMA, has also written about Free and Open Source SMA designs in [LL16] [LL18] [LL19]. To this date, two designs have been proposed by the author: the TinySpHEAR, and the Octathingy. The TinySpHEAR is a FOA microphone featuring four capsules in a tetrahedral configuration, while the Octathingy, inspired by [Ben12], is a 2OA microphone with 8 capsules arranged in octahedral geometry - capable of sampling 8 out of the 9 SHs required for 2OA.

One of the key features of the SpHEAR project is the OpenSCAD routines written for the generation of 3D models. The OpenSCAD scripting language, as already noted, allows researchers to modify existing models to fit the dimensions of different capsules, and also adjust the array radius without hundreds of manual modifications. The code for the OpenSCAD modeling of SpHEAR is public and can fully automate changes to the design.

The second important part of this work is the Octave GNU calibration routines which allow one to encode microphone signals into the SH domain, and also perform equalization of both A and B format responses. Finally, the author also provides PCB files and diagrams describing how to create these designs, and the accompanying publications explain how to calibrate the systems.

In [LL18], Lopez-Lezcano describes the use of a 5-degree-of-freedom robotic arm for the calibration of the Octathingy[53]. The arm allows automatic measurement of the system with 4096 different points of resolution. With 5-degrees-of-freedom the author is able to calibrate the microphone using either horizontal only IRs, or a combination of horizontal and vertical IRs. Certain spherical harmonics are not well suited for calibration using horizontal measurements only, as the response is completely null at $\theta = 0°$[54].

[53] The robotic arm costs $2,500.

[54] Consider for example the Z harmonic. θ is our elevation variable.

Mechanical Design and 3D Models One of the key features of this microphone array, similar to [GPL18], is the ability to 3D print these designs using fused filament fabrication (FFF)[55] - in lieu of more expensive methods such as SLA or SLS. This allows institutions with cheaper equipment to reproduce these designs without having to outsource the printing. The designs are created using FOSS, and the entire project is licensed under GPL/CC licenses.

In [LL19] Lopez-Lezcano shared an update of the project. In this publication, the author described various mechanical designs which were measured and acoustically validated. A second version of the capsule holder was designed to improve the cardioid response of the capsules - which was perturbed by the enclosure. The author noted that the empty space inside the initial array (shown in Figure 2.5) resulted the offending resonance. This affected the response of the sensors at the frequency corresponding to the size of the cavity.

As a result, in the second design, a new geometry was designed, which no longer featured an open array configuration, but instead positions each sensors at the base of a cone attached to a rigid octahedron. Figure 2.6 shows Octathingy Version 2, featuring this new design. Each cone has apertures for incident sounds arriving from behind the capsules[56]. Unfortunately, although the individual capsule response of this new design is better, it also requires a slightly larger overall radius - resulting in a lower spatial aliasing frequency. The TinySpHEAR is based on the capsule holders in Version 1.

Electronics Both the TinySpHEAR and Octathingy rely on active ECMs requiring phantom power for operation. The operating voltage of sensors vary depending on the package, and must be suitably powered for proper performance. Both the 10mm Primo EM182 and 14mm Primo EM200, used in these designs, have an

[55] Also known as extrusion method.
[56] The desired directivity is that of a cardioid microphone.

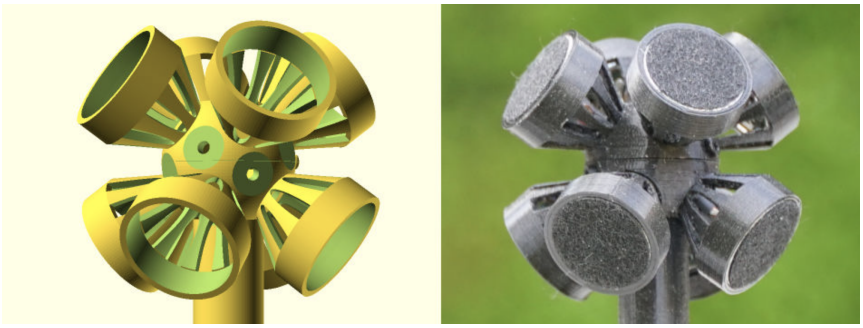


Figure 2.6: Octathingy Version 2 [LL19]

operating voltage of 5V, which means the standard 48V output from phantom power needs to be stepped down[57].

In both the the FOA and 2OA designs, this is accomplished using a Zapnspark circuit. The larger 14mm capsules were reported to provide better low-end performance, highlighting a trade-off between capsule diameter, noise performance, and array radius. Larger diameter capsules have been shown to have better noise performance, which becomes important during the SH equalization step.

The Octathingy[58] uses a standard DB25 connector, as opposed to the 12-pin Amphenol DIN connector from the TinySpHEAR, which is not common in audio applications. The ECMs used in this project can be found online and are of exceptional quality, however, they are also quite expensive - reducing the availability of this system.

Calibration By far the most laborious aspect of designing a proper HOA microphone is the calibration process. There are several steps involved for complete HOA array calibration. Below is a reduced list, describing the minimum requirements:

1. Determine truncation size based on shortest reflection path (unless anechoic

[57]The operating voltage can be found in their respective specification sheets.
[58]Also called OctaSpHEAR.

chamber is available).

2. Create compensation filter for speaker response, taking into account the measurement from step 1.

3. Find suitable system for IR generation and acquisition. The system should ideally be automated.

4. Determine how many poses are needed for A-format equalization and generate A-format correction filters.

5. Encode corrected A-format signals into SH domain, after they have been equalized.

6. Determine how many poses are needed for B-format measurements, and analyze B-format responses to create SH equalization filters[59].

7. Ideally, download all filters and apply them via a VST or other easy-to-use tool.

Determine Truncation Size

Much like with other authors (i.e. [MW19]), the first step towards calibration is to simulate a free-field environment by compensating for the effects of the speaker and room. The shortest reflection path will determine the truncation size of our IRs[60], as well as the minimum frequency for reliable measurements. In [LL18], the 4.5 msec direct path translates to a 220Hz limit[61]. This shortest reflection path will also determine the quality of the compensation filter we create for the speaker response.

[59] Alternatively, use a modeling approach to equalization.

[60] Shortest reflection path corresponds to the closest surface relative to the sensor array.

[61] $1/f = T$ corresponds to the period of longest wave. We can use $f/c = 1/\lambda$, where λ is the wavelength, to calculate f based on measured distance. For Lopez-Lezcano it was 1.5 meters roughly or 5 feet.

Create Compensation Filter for Speaker Response

One may either chose to create this filter themselves[62], or use a software library, such as Digital Room Correction (DRC) to create this filter. Providing as much detail as possible regarding room dimensions, reverb time, and noise floor will help analyze the final design. If an anechoic chamber is not available, then, in order to achieve accurate results, a large room with little to no reflections, and low noise floor, is suitable.

Using the truncation time, re-calculated in samples, all measurements will need to be trimmed after the impulse response has been extracted from an *Exponential Sine Sweep* (ESS)[63]. Canfield-Dafilou et al. [CDCJ18] note that the ESS should ideally be re-recorded using a direct line to the audio interface, to compensate for any effects the interface might have on the signal[64]. This, in practice, is often not performed because of IO[65] limitations. Instead, we consider the interface's effects to be negligible, especially if the bit-rate is high. Thus, in practice, the sine-sweep is commonly saved in memory until it is needed for the deconvolution. Equation 2.12 shows the deconvolution process.

$$h(t) = \mathcal{F}^{-1}\left(\frac{\mathcal{F}(r(t))}{\mathcal{F}(c(t))}\right) \tag{2.12}$$

In this equation $h(t)$ corresponds to the IR in the time domain, $\mathcal{F}^{-1}$ is the iFFT operation, $\mathcal{F}(r(t))$ corresponds to the FFT of the recorded signal, and $\mathcal{F}(c(t))$ corresponds to the FFT of the original ESS.

Since convolution in the frequency domain corresponds to multiplication, deconvolution corresponds to division. Once the response of the speaker has been found using a flat microphone, an inverse filter can be generated, such as in [MW19], using the Nelson/Kirkeby method. Lopez-Lezcano does not use this method, but instead

[62] Using for example JOS's imp_meas.
[63] The ESS technique is the most common method today for capturing IRs.
[64] The interface belongs to the transfer function path.
[65] Input/output.

inverts the response directly, and subsequently uses the *Minimum-Phase Spectrum* (MPS) method - by Julius O. Smith [Con]. Alternatively, Lopez-Lezcano uses the Digital Room Correction software package.

After the IR is found, it is truncated - and a windowing centered at the impulse is applied in order to avoid discontinuities in future FFT analyses. The same will be done for all future IRs captured by the array. Lopez-Lezcano recommends using a Blackman window as he states it provides: "empirically better ripple in the passband, which translates to flatter response at low frequencies." The speaker compensation filter will be convolved with all subsequent measurements, negating the effects from the speaker.

Build Automated IR Acquisition System

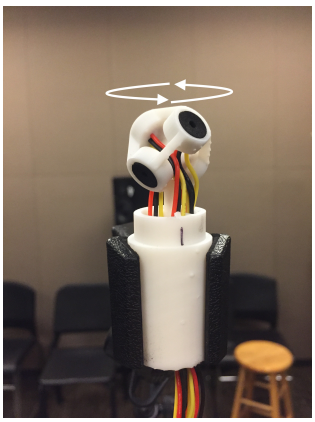
Once the compensation filter for the speaker response has been generated, we must acquire a system capable of capturing a dense grid of IRs to analyze and ultimately equalize our SMA. Lopez-Lezcano [LL18] uses SuperCollider (SC) to send serial data to the aforementioned robotic arm, and then transfers the recordings to Octave GNU for analysis and filter design. The robotic arm uses an "Arduino based built-in control processor". Unfortunately, the SC code was not found, however, recreating it would likely not be too hard since all the synthesis involves is an ESS.

The other option is to use Aliki, which is a Free and Open Source IR generator, however, there is no direct way we know to interface Aliki with Arduino; this means the person has to manually move the array, possibly hundreds of times, which could lead to substantial errors. This process would also take a very long time, especially if the sampling grid is very dense. In Section 2.2.2, we will talk about the solution the present author has implemented in the past, and what possible changes we adopt to make it more accessible.

Calibrate A-format IRs

Once a proper IR measurement system has been acquired, and the IRs of the array have been captured, the next step is to create the equalization filters for the A-format

(a) Pose 1: Single Capsule - Horizontal



(b) Pose 2: Full Array - Horizontal

(c) Pose 3: Full Array - Vertical

Figure 2.7: Three Poses for SMA Calibration

signals. There are two purposes to this: to make the capsules responses flatter, and to make the capsule responses match each other. In contrast to cylindrical arrays, such as the one in [MW19], for spherical arrays we need to use both horizontal and vertical responses.

Let's consider the *pose* of the array - with respect to the speaker - and the axis of rotation (e.g. the center of gravity), in order to demonstrate this process. Figure 2.7 shows the three poses the present author has previously implemented in the field. Image 2.7a shows a FOA microphone array centered for single capsule measurements, to analyze the directivity of a single capsule. Figure 2.7b shows a different FOA array centered for horizontal measurements of the entire SMA, the axis of rotation is directly below the center of the tetrahedron[66]. Figure 2.7c shows a third FOA microphone, rotated 90° in preparation for vertical measurements of the Z harmonic.

As noted, the purpose of these filters is to fix any irregularities in the frequency

[66]This pose was used for X and Y harmonic sampling.

response of each capsule. Ideally, A-format calibration filters should be measured using Pose 1, repeating the process for each sensor in the array. Unfortunately, this substantially increases the amount of time required for calibration, since each capsule requires a whole new set of measurements.

In the A-format Octathingy calibration, horizontal measurements of the entire array are used to calibrate the A-format signals (e.g. using Pose 2). Lopez-Lezcano, unfortunately, does not provide much information about the generation of these filters in the associated publications. From inspecting the open source code, we discover that the inverse filter is generated using three measurements from a direction proximal to the angle of the capsule in question - in the azimuthal sense. The measurements are also weighted according to a cardioid response and averaged. The resulting response is then inverted directly without regularization, and converted to minimum-phase using the technique described by JOS:[67].

> "Any spectrum can be converted to minimum-phase form (without affecting the spectral magnitude) by computing its cepstrum and replacing any anticausal components with corresponding causal components. In other words, the anticausal part of the cepstrum, if any, is "flipped" about time zero so that it adds to the causal part. [Con]"

Finding suitable methods for A-format calibration in SMAs is an open area of research. The method chosen by Lopez-Lezcano works reasonably well using only horizontal measurements, however, the 8-channel system is relatively small[68] compared to other arrays. Additionally, the elevation difference, relative to the horizontal plane, is the same for all capsules; for more complex arrays, where multiple elevation angles are featured, it is not clear that horizontal only measurements would be the best choice. In the Octathingy design, the calibration of A-format is designed to optimize reproduction of horizontal signals; this means the optimal recording position of the microphone, would be perpendicular to any musicians.

[67] Using the technique in [Con], makes the filter causal.
[68] In terms of the number of sensors.

Encode Calibrated A-format IRs

With the A-format equalization filters derived, the original IRs[69] are filtered and subsequently encoded using a matrix of coefficients derived from the SH representation of the capsule positions. Once the matrix is initialized, it is inverted to produce the encoding matrix [MDMF+18].

The inversion matrix for non-uniform distributions can either be calculated using the Moore-Penrose technique or SVD[70]. Equation 2.13 shows the encoding matrix calculation given uniform and non-uniform distributions. The non-uniform solution in Equation 2.13 describes the Moore-Penrose approach.

$$\mathbf{Y}_e = \begin{cases} \frac{1}{Q}\mathbf{Y} & \text{uniform} \\ \mathbf{Y}^\dagger = \left(\mathbf{Y}^T\mathbf{Y}\right)^{-1}\mathbf{Y}^T & \text{non-uniform} \end{cases} \tag{2.13}$$

If using the SVD method (e.g. $Y = U\Sigma V^T$), the pseudo-inverse will be calculated by:

$$Y^\dagger = V\Sigma^\dagger U^* \tag{2.14}$$

The star superscript indicates conjugate transpose. If a matrix has all real components then the conjugate transpose is just the transpose. If the matrix has complex entries, you take the conjugate and transpose the matrix [Sin]. $\Sigma^\dagger$ is formed by calculating the reciprocal of each eigenvalue[71] in Σ. Both methods should yield the same matrix.

The encoding matrix is only valid below the aliasing frequency where the wavelength of the impinging sound is large enough to be accurately sampled in the SH domain by the mic array[72]. For this reason, additional equalization must be applied,

[69] This might include multiple elevation angles.
[70] Singular Value Decomposition.
[71] One over the original value. The original σ entries are called singular values.
[72] One should also consider the source's relationship to the receiver. Various authors opt for 2m and ESS times of 10 seconds. 20Hz is roughly 17 meters.

especially above the critical spatial aliasing frequency where the responses begin to deteriorate.

In SpHEAR, the author uses a slightly different approach; Lopez-Lezcano derives the encoding matrix as a function of the measured IRs - much like in [MDB06] or [JEP13]. The matrix of encoding coefficients is not just the inverse of the projection matrix, rather, it is the optimal matrix solving the system of linear equations involving the specific IR powers[73] and the ideal SH representations. Since this array is irregular, the solution is also inverted. The final matrix is frequency-independent; the powers in the critical bandwidth is integrated and averaged to find these values. Unfortunately, the reference given for this process is unpublished. We discovered this by inspection of the code.

In Moreau et al. [MDB06], the authors compare the frequency-independent matrix approach[74] versus the frequency-dependent inversion approach - which yields a matrix of encoding filters. One can use simulated responses, or idealized responses, in order to find the desired filter matrix. Lopez-Lezcano uses a idealized responses and real measurements[75] to find a matrix of scalar coefficients, and subsequently derives custom equalization filters based on measurements, rather than applying radial filter theory.

Find Equalization Filters for B-format Signals

As aforementioned, no radial filters or a matrix of filters are derived in this project. Instead, the author analyzes a set of initial B-format signals, resulting from the A-format equalization, plus the transformation from the encoding matrix, in order to create custom B-format equalization filters.

According to the author:

> "Our software defines the shape of the B-format equalization filters by

[73] At a specified frequency range, where the capsules are considered co-located.

[74] Using a matrix of scalars plus radial filters.

[75] The RMS power is calculated over a specific frequency range determined by the size of the array.

> measuring the power in logarithmically spaced bands for measurements near the peak of each recovered lobe. The inverse of this power profile is used to create the FIR filters.[LL19]"

The measurements corresponding to the points where the SH features a maximal response are strictly used to define this equalization profile. The B-format equalization filters are then generated much the same way as the A-format ones via the MPS method inverting these maximal responses.

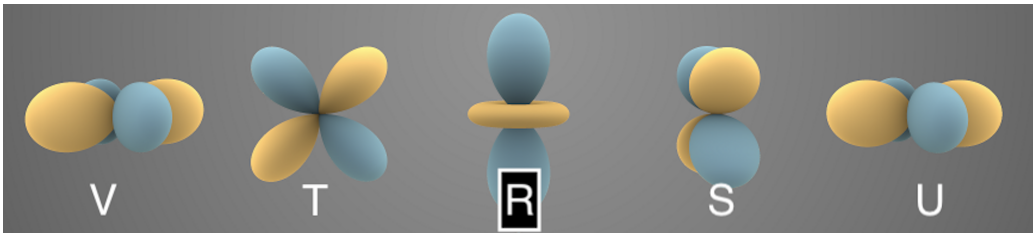


Figure 2.8: SH of Ambisonic Order 2

Unfortunately, the horizontal only measurements do not provide any information suitable for B-format equalization at certain harmonics. Consider Figure 2.8, here we see clearly that certain harmonics (e.g. T and S) contain nulls at elevation $\theta = 0$. This is for example why Pose 3 was implemented in [Zal19], which allowed us to determine if our own array could sample the Z harmonic[76].

While harmonics Z, T, and R[77] can suitably be equalized with a set of measurements using Pose 3, due to the orientation of harmonic S, it cannot be equalized using either of these measurements[78]. Figure 2.9 shows the horizontal cross section of harmonic V, using a 2D polar plot, illustrating the result of sampling this harmonic using horizontal only measurements (e.g. Pose 2).

Assuming symmetry between harmonics T and S, we could apply the same equalization to both harmonics. This theory can be further expanded to determine the

[76]Since Lopez-Lezcano has access to many elevation angles, this is not a problem. However, if we don't have access to a robotic arm, we will have to figure out some other method.

[77]2, 5 and 6 in ACN ordering.

[78]It is rotated 90°compared to harmonic T.

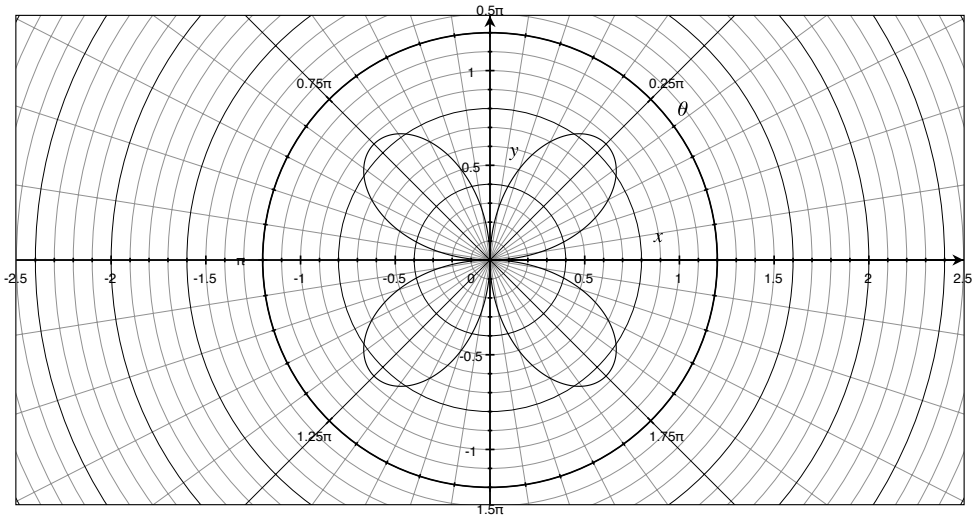


Figure 2.9: V Harmonic - Horizontal Cross Section - Polar Plot

fewest number of filters required for calibration of all harmonics, however, in the real world, these arrays are never perfectly symmetrical, so the compromise comes at a cost.

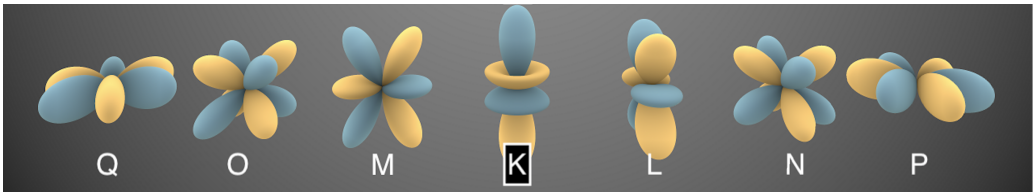


Figure 2.10: SH of Ambisonic Order 3

Consider now Figure 2.10, which shows SHs of ambisonic order 3 (note in particular harmonic O). At higher orders, certain harmonics have nulls using both Poses 2 and 3, which means a different angle needs to be accessed for the calibration filters to be derived. For 3OA, we would need at least Pose 2 and 3, and a third measurement where either the speaker or microphone had a 45° tilt. This demonstrates that the calibration processing time, for this method, has a direct relationship with the order of the array.

Lopez-Lezcano also notes that second order components (e.g. V, T, S, U)[79] drop in volume at 6dB per octave with decreasing frequency. This means that the B-format correction filters provide intense amplification at those frequencies, which results in significant self-noise. As a result, it is necessary to add a high-pass filter to the design. Alternative equalization techniques such as those discussed by Middlicott et al. [MW19] might be worthwhile exploring as an alternate means of B-format correction.

Radial filters also tend to boost the low-end of these arrays, which can further exacerbate the noise problem. Care should be taken if one wishes to use both radial filters and equalization filters. Due to the aggressive nature of radial filters, other authors have also suggested high-passing harmonics in the encoding process (i.e [Ven14]). The high-pass filters should be commensurate with the ambisonic order, which affects the magnitude response of the radial filters (which are order dependent).

Export Filters and Encoding Matrix

In order for musicians and recording engineers to use this system, an easy-to-use interface is required. Having to import recordings into Octave GNU for processing every time would be cumbersome. As a result, the author suggests exporting all the equalization filters and the encoding matrix for use in a plug-in. In the SpHEAR project, the Faust library is used to create an Octathingy encoder that can run on a traditional DAW.

Lopez-Lezcano notes the inclusion of a expander in the final encoder, to compensate for the SNR of 2nd order SHs. The expander works inversely to a dynamic compressor - instead of making loud signals quieter, and weak signals louder, it expands the dynamic range by doing the opposite. The expander is only applied to the 2nd order harmonics, which distorts the spatial image - as conceded by the author. Other dynamic compression systems, such as those discussed in Chapter 1,

[79] We don't actually sample R, because there aren't enough sensors.

apply the same attenuation to all harmonics in order to preserve spatial acuity. This time-variant system is implemented in the final real-time encoder[80].

Summary As we can see, while SpHEAR offers a great deal of insight into the design and calibration of SMAs for FOA and HOA recording, the calibration process, in particular, make these systems difficult to implement. The use of high-quality ECMs also make the design expensive, especially if considering 3OA or 4OA. The analog nature of these capsules implies the need for multi-channel audio interfaces, which further increase the cost, especially as we increase the ambisonic order.

Despite these limitations, SpHEAR is undoubtedly the most comprehensive, accessible, and high-quality design we have found. Every piece of software used for the design is FOSS. This makes it possible for anyone with enough patience and perseverance to develop one of these microphones at home. The models are designed for low-cost 3D printers and the electronics are distributed publicly[81].

One of the main problems with the 2OA mic in particular, is that the missing harmonic R might make it incompatible with other VSTs and effects. The calibration process, is not only laborious, unfortunately, it is also necessary, since these capsules have a ±3dB manufacturing tolerance. In contrast, the MEMS capsule we will talk about in the next section, have a ± 1dB tolerance, arguably making individual microphone calibration less critical.

Ambisonics Z Array Over the last few years, the present author has been investigating the viability of MEMS sensors for the development of a free and open-source ambisonic array. In our latest publication [Zal19], we exploited the small size of the MEMS to increase the coincidence of capsules, thus increasing the spatial aliasing frequency to good effect.

[80]The code for this process was unfortunately not located by the present author, so not much more can be said about it.

[81]The authors even created additional models for a windscreen and microphone holder.

The polar plots resulting from acoustic measurements of the constructed device, in this case a tetrahedral FOA MEMS array, showed improvement in the polar response of dipole elements as a result of increased coincidence - as predicted by the theory. No phase compensation was needed in order to generate the velocity elements in the system, however, we did encounter the unexpected consequence that the omnidirectional response[82] was difficult to generate. Further investigation is required to evaluate if any phase corrections can be introduced to this system in order to obtain a more pure omni-directional element (e.g. W channel).

In order to validate the results, two arrays with different diameters were built. Our latest efforts, not yet published, involved using recordings taken with these two arrays to perceptually evaluate their sound quality in a listening experiment. The project was diligently submitted to the International Review Board (IRB) for approval of study with human subjects, however, due to the pandemic[83], the experiment has been delayed. We also built an Octathingy prototype with MEMS capsules, which we intended to use to test the calibration routines by Lopez-Lezcano, unfortunately, this also has been delayed.

MEMS MEMS sensors offer many benefits compared to ECMs:

1. They are very cheap.
2. The manufacturing tolerance is very small, so the frequency responses are already fairly well-matched.
3. They are small, which means they can be placed closer together to increase the aliasing frequency.
4. Digital MEMS can be directly recorded without the need for an external ADC.

[82] Channel W.
[83] 2020-2021, SARS-CoV-2

Unfortunately, they also have many compromises:

1. Their SNR is worse than ECMs (64dB SNR v. 78dB)[84].
2. As a result of the poor SNR the dynamic range is limited.
3. They need to be surface mounted (e.g. they are soldered using a reflow oven or something similar).
4. They have a Helmholtz resonance above 10kHz.
5. They are omni-directional.
6. They are very fragile, which means the sound port needs to be protected. Acoustically transparent foam might solve this.

The commercial Zylia microphone uses MEMS capsules internally. This is promising as it indicates that MEMS capsules can be used for HOA in professional recording conditions.

Automatic Rotating Microphone Mount (ARMM) In our experiments, we have been using the ScanIR [VGR19] MATLAB toolbox to generate and capture our impulse responses. Our automatic rotating system is based on a cheap stepper motor with a custom attachment, which is driven by an Arduino. This system only rotates along one axis. As a result, for vertical sampling of our array, we manually adjust the pose of the microphone. This is trivial for FOA, but might be less accurate and cumbersome for HOA.

One of the future goals of the author is to better describe the rotating microphone system and provide a 3D model attachment that one can print to create a similar system. Currently, the attachment that allows us to connect the motor to

[84]For the ICS43434 or ICS-41350, two capsules we have chosen, and the Primo Capsules.

the microphone stand boom pole was mechanically created using expensive equipment. However, creating a 3D printed model for this purpose should not be difficult.

In our next experiment, we hope to adopt the philosophy and methodology of Lopez-Lezcano, and attempt to use more FOSS. Since the present author does not have experience with SC, we might instead opt for Pd - which has Arduino support.

Open Questions The designs proposed by Lopez-Lezcano provide a good starting point for building and calibrating HOA microphones of various sizes, unfortunately, the calibration software is far too complicated for most musicians to use - even with example files provided. Beyond using different components (i.e. MEMS capsules and a MicroController), we would also like to investigate how the tolerance of MEMS contributes to the need for individual array calibration.

In other words: *is individualized microphone calibration noticeable by listeners given the improved tolerance of the capsules?* If it's possible to demonstrate that using MEMS results in imperceptible microphone calibration, we could distribute a single software solution that works for all people wishing to build this design. This would mean that the musician would only have to worry about the mechanical aspects of the design.

An additional open question regards the SNR of the system, which as we have noted is inferior to those with ECM capsules. An easy experiment to conduct in the short-run would be to record various stimuli using our existing FOA designs, and professional FOA microphones available to us, and compare their sound reproduction with music samples featuring wide dynamic ranges. In addition, we should evaluate the amount of noise present in higher order SHs, to validate the proposition of MEMS for HOA.

Since the directionality of MEMS capsules is omnidirectional, we must also investigate how this affects the array responses. Finally, we would like to look into radial filters to improve the performance of these arrays, which one can combine with

equalization methods.

Future Work While there are SMAs in the literature with MEMS capsules, we have yet to see any HOA arrays that use digital MEMS. We believe this might be one of the possible ways in which we can contribute to the community. Additionally, it would appear as though not enough research has been done in A-format, or B-format, calibration using SMAs.

While it is possible to make these microphone more accessible, the individual calibration process greatly complicates matters. Ideally, we'd like to create a design that is simple enough to build in an afternoon, with little to no engineering experience. This way, people interested in HOA microphones can experiment with this recording technology as a way to make spatial music.

As we noted, the SNR of these capsules is quite poor compared to professional ECMs. In order to improve the SNR we might consider using a surplus of capsules to record lower order ambisonic sound-fields. For example, a dodecahedron (e.g. 12-faced platonic solid) could be used for a full 2OA design, which only requires 9 capsules. Alternatively, one can use multiple capsules per sampling node to improve SNR, as suggested by Backman [Bac06a].

An unfortunate part of the MEMS-based designs is that they require surface mounting. Over the years we have been lucky enough to find industrial reflow ovens at different fab-labs[85]. In our next designs, we are considering using a hotplate or other cheap DIY solution[86] This would allow others to safely solder these components at home.

The 10kHz Helmholtz resonance caused by the inner dimensions of these capsules must be treated for proper performance. We believe the approach implemented by Middlicott et al. [MW19], using the response of the Dirac delta function as a target,

[85]Fabrication Labs at Tandon (NYU), and EnVision (UCSD).
[86]Using a small grant from our University, we were able to buy one of these for $30.

can be used to address this problem. This can then be compared to the other methods proposed in that same publication.

Considering the omnidirectional nature of these capsules, various array geometries should be evaluated to determine how these affect the directionality of the sampling node (i.e. rigid, hollow). We believe one of the strengths of MEMS is their smaller size, so the geometry should be adapted to maximize capsule coincidence. This reduces the nefarious effects of spatial aliasing which distort SHs.

The capsules also need to be protected by acoustically transparent foam to prevent dust from affecting the response of the array. In the past we have used laser cutters at various fab-labs to make foam rings. However, these machines are extremely inaccessible without membership to a fab-lab or access to University equipment. Therefore, in future designs, we intend to use a specific material that can also be found from online laser cutting service. Another alternative is to offer stencils, which the person can use to cut these by hand.

2.3 Conclusion

This chapter has discussed the considerations regarding the development of Free and Open Source SMAs. In the introduction, we discussed the basic principles of ambisonic, including normalization and ordering standards. In the second section, we discussed some planar microphone arrays. In particular, we discussed in depth the calibration of capsules in HOA arrays.

The main focus of this chapter, however, has been on SMAs designed using open source software. The MSMA, HOSMA and SpHEAR projects were discussed in depth. These three projects constitute the most comprehensive open source microphone arrays we found in the literature. Finally, we discussed the development of the authors own microphone array, which uses MEMS capsules to increase capsule coincidence in a FOA configuration.

The goal of this chapter, was to explore methods to democratize access to HOA arrays. In this manner, artists wishing to use this technology in their musical compositions will not be obstructed by the cost of these devices. The calibration process makes it very difficult to create these designs. Therefore, a standing project involves evaluating the perceptual effects of personalized calibration filters on binaural ambisonic reproduction.

Chapter 3

Equitable Distribution of Spatial Music Using WebXR

Question: *Your various research projects during the past years have revolved around capture, production, development, and presentation of 3D audio and music. A recent important topic has been providing access to communities with less resources to produce and consume multi-channel music and audio (usually in ambisonic format). Present your research on the history of distribution of surround and 3D audio and music in general, and examine and compare contemporary technological infrastructures which may serve your aspirations of wide distribution of such music without the extensive equipment requirement on the consumer side. Pay special attention to the quality of the algorithms of such approaches. Finally present your plans for the use of some of these approaches in your own work or the work of other artists whose works you have produced or presented.*

3.1 Introduction

Spatial audio has been a subject of great interest in the electro-acoustic domain for many decades. Ever since Schaeffer, Stockhausen, and Cage, began experimenting with tape players as a means of separating the musician from the sound source, composers have ardently explored the possibilities that new technological developments afford them in this domain. The last few years have seen an explosion in the field of *Extended Reality*[1] (XR), and, as a result: spatial audio. Major tech companies have rolled out increasingly affordable *Head-Mounted Displays* (HMDs) along with software solutions to author media for these devices.

Despite the technological advancements in the commercial sector, there are countless compositions being written today that are seldom experienced with any spatial information by the general public. A substantial number of composer generate works routinely which elegantly make use of multi-channel arrays; unfortunately, many of these works still get *down-mixed* to stereo in publishing - destroying most of the pertinent spatial information. While the stereo mix-down generally results in a high-quality standardized playback format, the option to disseminate these works with additional spatial information warrants further exploring; this is especially true in cases where composers have a made a career out of employing spatial sound as a primary compositional feature of their work.

In this chapter, we are examining: libraries, players, and, musical works using browser-based spatial audio solutions. These libraries and players can be leveraged to allow for improved distribution of musical works on the web, while the musical works that use them, serve as examples of what is already possible today. Traditionally, only a small number of individuals who attend live events are privileged enough to listen to these musical works as intended. Our hope is to create a culture of documenting and sharing our spatial works online, using *Free and Open Source Software* (FOSS)

[1] XR includes Virtual Reality, Augmented Reality and Mixed Reality.

and low-cost hardware.

3.1.1 Framing the Problem

There are four basic questions we seek to answer:

1. How do modern composers generally distribute their spatial music to the general public?
2. What are the potential problems with these methods?
3. What tools/system might work better?
4. What are the problems with the proposed solution?

In this chapter we will attempt to methodically address these different questions. Before we can begin this process, however, we want to present a brief overview of the attempts that have already been made by the private and public sectors with regards to this problem. This should provide us with a starting point to begin comparing and contrasting different technologies available today.

In our history of 3D audio distribution, we are focusing specifically on direct-to-consumer distribution systems that do not rely on intermediary parties[2]. In the commercial music world, the traditional formats that we have all come to know about are listed below. We purposefully left out radio as it is not an "on-demand" play-back system - the listener cannot control what will be played back.

1. **Vinyl records:** 12-inch Long Play (LP) format disks being perhaps the most popular. These were common in the 1940s.
2. **Cassette tapes:** magnetic tapes which were popular in the 70s and 80s with inventions such as *The 8-track* and the *Sony Walkman* reaching mass market in automobile audio and personal audio respectively.

[2]Such as in a movie theatre setting.

3. **CDs:** optical media developed by Phillips and Sony in the 90's. The first digital medium for recording and reproducing audio. Their loss in popularity started in the 2000s, with the advent of MP3 players.

4. **MP3 files:** compressed digital files which were typically stored in mp3 players like the iPod. Began a contentious era of copyright infringement arguably still going on today. An mp3 player was capable of holding thousands of songs in a more portable medium than a CD player.

5. **Streaming:** today streaming is the most popular playback method. With streaming, songs are only temporarily buffered into memory. After they are heard they are erased from the computer memory.

The preceding list originated from an article by Sioni [Sio18]; as we can see, it does not reveal much at all in the distribution of spatial audio - this is because not much has been written on the subject.

In the 70's, when *quadraphonic* audio was first being developed, there was a good effort to commercialize four channel speaker systems to the public, along with tape players capable of reproducing multiple channels at once [Pos90]. Unfortunately, the quadraphonic system never gained much popularity, perhaps in part because of its insufficient coverage of the listening area. The major breakthrough for personalized spatial audio came in the 90's, when commercial 5.1 surround sound system began being distributed to the general public by Dolby and THX [MP09]. This allowed people to sell DVDs of musical performances, created using this surround sound format. Unfortunately, while many composers have created recordings for 5.1, very few people are willing to invest the money for such sound systems. A number of important spatial music scores can be experienced at movie theatres, however, this leaves composers beholden to the interest of directors, and, ultimately, unable to compose freely.

3.1.2 Head-Tracked Binaural Audio

Head-Tracked Binaural Audio[3] (HTBA) provides a way for listeners to experience a *virtual surround sound* experience without anything other than headphones, and some tracking system to determine the listener's head orientation. This is accomplished by acoustically measuring the effect the human head has on sound, and applying these features to musical sounds. This allows for individuals without any specialized equipment, or skills, to consume spatial music from the comfort of their homes with relatively little financial overhead. Modern processors are ubiquitous and fast enough to perform the necessary processing in real-time, even adapting as the user changes perspective.

Unfortunately, these are also some drawbacks with HTBA in contrast to surround-sound formats. Namely:

1. Latency between the head-tracking system and the audio reproduction system can lead to playback inaccuracies,
2. The frequency response of the playback system (i.e. headphones or earbuds) may degrade the spatialization efforts,
3. The lack of personalized HRTFs may result imperfect performance, and,
4. The head-tracking system may be cumbersome or require a complicated set-up procedure.

Despite these potential problems, we believe that HTBA is the most cost-effectiveness and equitable method for distribution of spatial music. Consider in contrast some of the problems associated with surround-sound:

[3] In contrast to Fixed-Perspective Binaural Audio (FPBA), in which users' head movements do not affect sound reproduction.

1. While latency is not a problem, incorrect speaker placement will greatly distort the spatial image. Precise speaker placement may not be possible in certain environments,

2. Even if frequency response of the speakers is excellent, the acoustics of the room will still affect the spatial balance. Thus, additional acoustic treatment might be needed, and,

3. Surround-sound systems are a form of specialized equipment, only affordable to a small minority, or those using them for their work.

In contrast, HTBA can be done with common tools people rely on daily. Smartphones, for examples, are capable of performing HTBA using internal *Inertial measurement Units* (IMUs)[4]. Another way of performing HTBA, is to use the camera sensor in a computer, or tablet, to track the users' head-movements [DW20].

There are cheap and easy-to-build head-trackers, that can interface both with DAWs[5] and web-based solutions. Finally, even though the *Human Computer Interaction* (HCI) is not perfectly ergonomic, one can also use a mouse to rotate an ambisonic scene[6], which is preferable to having no control over perspective at all. Studies have shown the HTBA offers improved externalization, localization and reduction of front-back confusions compared to *Fixed-Perspective Binaural Audio* (FPBA), in which the audio is technically binaural, but the listener has no agency over the listening perspective. More importantly, however, this method allows people to experience spatial music without any lavish equipment.

[4]For example, while using Google Cardboard.
[5]Digital Audio Workstations.
[6]Ambisonics is a common technique used for HTBA.

3.1.3 Spatial Music Composers

During research into spatial audio in the contemporary music domain, we found that there is an important number of artists working routinely with spatial audio - as a way to articulate their compositional ideas. A very short list of notable composers, along with their methods, includes:

1. **Natascha Barrett:** HOA granular synthesis,
2. **Fernando-Lopez Lezcano:** HOA virtual auralization using *Room Impulse Responses* (RIRs) and HOA microphone designer,
3. **Eric Lyon:** creator of *LyonPotpourri*[7] library, featuring a number of useful Pd externals for multi-channel music (i.e. `splitspec~`: split an incoming sound into complementary spectra),
4. **Kerry Hagan:** real-time noise-like compositions featuring maximally uniform sequences,
5. **Denis Smalley:** known for his theory of *spectro-morphology* [Sma97][8].

As one will notice when investigating these different composers, despite the fact that they have intensively worked on multi-channel sound in their creative practice, it is not possible at this present time to listen to their repertoire without attending a live event, or performing some rigorous preparations involving expensive equipment.

Many of these artist opt for FPBA, using binaural synthesis renderings, as a means to present their works online. Alternatively, some artists record their works with a binaural microphone, and a multi-channel sound system, yielding an equivalent effect. Unfortunately, this means that much of the careful spatialization work is

[7] Link.
[8] This is a very short list. Used for demonstrative purposes.

ultimately only experienced by a small number of attendees at *High-Density Loudspeaker Array* (HDLA) concerts.

Another approach is to provide listeners with a HOA recording, or a patch that performs the music; the former being the case with Barrett, and the latter with Hagan. This leaves us with the problem that a costly loudspeaker array is required for each individual to experience the work. The HOA recording can also be played back by the listener binaurally - if they have experience with ambisonic playback - but for the general public, this will simply be too burdensome.

Luckily, a number of possible solutions to this problem have already been created. *Javascript* (JS) libraries and players have already been developed allowing composers to present their works in a more dynamic fashion. Specific musical works we will discuss highlight the possibilities of these tools. In the proceeding sections, we will focus on these technologies, and works, paying particularly close attention to FOSS solutions. Additionally, we will attempt to highlight low-cost hardware options, which we believe can address the equity problems surrounding the development, and presentation, of this type of material.

3.2 Alternative Distribution Systems

While this paper focuses primarily on WebVR as a solution to the equitable distribution of spatial music, we should note that there are other systems that could be viable alternatives. As expected, these have their own benefits, and limitations, when compared to web-based solutions. The two other possible solutions we found, correspond to the delivery of spatial audio using *native* XR experiences consumed using a: dedicated HMD[9], or, mobile HMD.

Dedicated HMDs, in contrast to mobile HMDs, are costly devices with the sole purpose of presenting XR content. Given their proprietary nature and high-cost,

[9]Head-Mounted Display.

we don't see these as being a proper solution for equitable distribution of spatial music today. In a distant future, it is possible these systems may become part of our everyday lives in the same way that personal computers have, but, as it stands, these entertainment systems remain a luxury for most people.

Stand-alone mobile XR apps (e.g. native applications), created using the Google Cardboard SDK, for example, are far more accessible, and provide an proper alternative solution to our distribution problem. The benefit of native experiences is that the content is entirely downloaded prior to playback. This makes it possible to create more sophisticated works that are larger in size. In contrast, due to the allocation structure of data using web applications, WebVR pieces tend to be relatively small[10].

There is one other benefit of web-based solutions, over native mobile solutions - web-based solutions can be experienced over a multitude of platforms, including: mobile devices, personal computers, and even dedicated HMDs - without needing to recompile these for different systems. These solutions are, as a result, not entirely hindered by the *Operating System* (OS) of the mobile device or personal computer. Most personal computers support WebXR over the right browser[11]. This lies in contrast to stand-alone mobile XR apps that only work on modern, and often expensive, mobile devices.

While WebXR offers some improved flexibility, the largest drawback is that an active and stable internet connection might be required throughout the experience for smooth performance. Rather than downloading assets into hard memory, browsers instead store data in *Random Access Memory* (RAM) for temporary usage; this means that projects are generally required to have a smaller data footprint. In the next section, as we discuss various WebVR spatial audio solutions, we will highlight how different projects address the optimization challenges created by this architecture.

[10] In terms of data.

[11] As long as the CPU, GPU and RAM can handle the task.

3.3 WebVR Audio

While there are certainly numerous spatial audio techniques available today, ambisonics has unarguably become one of the most popular for spatial audio on the web. Ambisonics has clear bandwidth efficiencies over *Object Based Audio* (OBA) solutions, as a large number of sources can be encoded into a lower number of channels - called ambisonic harmonics - and subsequently decoded binaurally. An ambisonic sound-field can also be rotated in a non-destructive fashion, making it ideal for HTBA.

This sound-field synthesis technique varies in quality depending on the order of the ambisonic system. A *First Order Ambisonics* (FOA) system, reduces the amount of data required to represent a sound-field. However, *Higher Order Ambisonics* (HOA), provides a more realistic experience given the improved spatial resolution of the system. This is one of the key trade-offs in the delivery of ambisonic music: the higher the ambisonic order, the better the quality, but also the larger the file sizes.

Fortunately, advances in data compression have resulted in improved 3D audio CODECs[12], which are able to reduce this data load substantially without major degradation in the *Quality of Experience* (QoE). Herre et al.[HHKP15] provide a good overview of multi-channel data compression systems, which informed the design of MPEG-H - one of the more popular proprietary CODECs for spatial audio. In contrast to stereo, where only two channels can be examined for redundancies, MPEG-H is designed to account for dozens of channels in OBA, HOA, and other formats simultaneously. Narbutt et al. [NOA+17] provide a good summary of the perceptual effects of bit-rate on QoE, using an open source CODEC[13].

Three minutes of raw uncompressed audio at 44.1kHz, and 24 bits per sample,

[12] COder/DECoder.

[13] In our own work we have adopted the OGG container format and OPUS CODEC, much like Narbutt et al.

results in roughly 2.2GB of audio data in 9OA[14]. In contrast, a 3OA system would result in 6 times less amount of space required; only 0.35GB of data is required to reproduce the sound field. A FOA file, with 3 minutes of music, would only require 0.09GB of data (90MB).

The size of the music file is an important bottleneck in these types of works. In cases where sound is transmitted live in an ambisonic format, it's important to determine the *bit-rate* of the audio stream[15]. It's also possible the data is dynamically downloaded from a server, rather than downloaded all at once. Dynamic streaming reduces the need for a lot of RAM, however, it also means that the internet connection needs to remain stable throughout the duration of the piece, rather than only when loading.

Finally, in order to perform the binaural synthesis required to create immersive *Virtual Auditory Environments* (VAE), web audio frameworks must perform numerous simultaneous convolutions (e.g. filtering). Convolution, even in the frequency domain, is a *Central Processing Unit* (CPU) intensive operation. In a *virtual speaker* (VLS) approach to binaural decoding, two convolutions are needed for each virtual speaker. The number of virtual speakers will affect the QoE, thus, a balance needs to be struck between high virtual speaker count, and low-CPU requirements. If the *Head-Related Impulse Response* (HRIRs)[16], are represented in the *spherical harmonic* (SH) domain, then the ambisonic order will determine the number of convolutions.

[14]Ninth order ambisonics. Anything over FOA is considered HOA.

[15]The bit-rate is the amount of data per unit time, generally seconds, that is required for proper reproduction. It also determines the size of the file during compression, and is correlated to the QoE.

[16]The filters containing the acoustic effects of the head, torso, and ears on the sound.

3.4 The Web Audio API

Many of the libraries we will talk about in the following sections build upon the *Web Audio API*[17] (WAA). The WAA itself, however, already contains certain functions for spatializing sound, and, is reasonably comprehensive.

The WAA [Web], in contrast to external JS libraries, describes the capabilities of any browser using this framework[18]. In contrast to the ambisonic libraries we will discuss, the WAA does not need to be imported into ones' project, as it is internally bundled with various browser distributions.

There are three main functions for spatialization within the WAA:

1. **Panner Node**: which allows for: *equal-power* (e.g. traditional stereo panning), or HRTF panning[19],

2. **Distance Effects:** which calculates gain values based on the position of listener(s) and source(s); sounds are amplified as a listener approaches a source as in real life,

3. **Sound Cones**: which determine the directivity of the sound source (e.g. omnidirectional, cardioid, etc.).

Unfortunately, the workflow for creating spatial music using the WAA is slightly more complicated than other tools, since it requires one to program every sound source in JS, rather than simply importing a sound-field created using more conventional tool - such as a DAW. *A-Frame* and *THREE.js*, two WebXR JS tool-kits, provide WAA-based positional audio which allow us to map geometries to sound sources. This makes the WAA a very powerful option, especially in circumstances

[17] Application Programming Interface.
[18] The WAA defines how these methods must be implemented to conform to the standard, it is not a library per se.
[19] Using interpolation methods rather than ambisonics.

where the number of sound sources is small. Unfortunately, for most musicians, setting up their own server and programming these works in JS is not realistic.

3.4.1 Inside the WAA

Carpentier [Car15] provides a good summary of the capabilities and limitations of the WAA. According to the author, both Mozilla's Firefox and Google's Chrome browsers are based on the Blink engine [Bli], which has the following deficits:

1. Filter kernels are not partitioned, which causes extra latency of half the FFT (e.g. *Fast Fourier Transform*) size,
2. The IRCAM Listen HRTF data-set [LIS], is hard-coded into the panner method, and,
3. The documentation does not explain how the HRTF set was selected, or why the *Impulse Responses* (IRs) were truncated to half their size - both of which likely affect the psycho-acoustic character of the system.

As a result of his investigation, Carpentier developed some routines to allow HRTF sets to be swapped. JSAmbisonics [PPQ16], a library we will discuss later, also borrows from Carpentier's work (e.g. The *BinauralFIR Node* project)[20]. Carpentier's publication talks more about HRTF personalization using the WAA. Details about Carpentier's work will be provided in Section 3.5.3.

Since the WAA is not a suitable solution for most composers, a number of libraries have been developed which allow for ambisonic reproduction via the web. Ambisonic is particularly flexible with regards to reproduction since the audio signals don't represent speaker channels, but, instead, define different parts of the sound-field. This means that beyond allowing composers to showcase their work online, these can be shared among institutions and played back using various speaker layouts[21].

[20] Part of the BiLi project with IRCAM.

[21] This quality of ambisonics is called isotropic. In contrast, surround sound is anisotropic.

This also make ambisonics an ideal format for equitable distribution of spatial audio, since both institutions with HDLAs, and those with smaller speaker arrays (i.e. quadraphonic), can enjoy these spatial works.

3.5 JS Libraries

Numberless *JavaScript* (JS) libraries have been created in the open-source domain that allow people to create immersive VAEs. These sets of functions extend the capabilities of the WAA, providing state-of-the-art spatial audio algorithms not found natively on the browser. We should note that there are certainly commercial JS libraries that are very promising, and just as important as the projects we will discuss.

These particular libraries were selected due to their licensing, which allows developers to contribute to them, and because they are free. The latter makes them attractive for independent musicians working on a limited budget, and helps address the equity issue surrounding the creation and dissemination of this type of music. The academic publications that accompany them also provide great insight into the inner workings of these algorithms, in contrast to proprietary software, which we could only discuss superficially.

3.5.1 Resonance

As noted, there are several frameworks available for creating binaural spatial audio online today. Among them, the most popular open-source solution is likely Resonance[22]. Gorzel et al. [GAK+19] describe some of the techniques used to optimize the HOA encoding and playback implemented in Resonance, which allow it to run on cheap hardware and older devices.

[22] From Google.

Encoding Optimization

In their publication, the authors describe the implementation of a *Look Up Table* (LUT), instead of using *cos* or *sin* functions in JS. The *spherical harmonic* (SH) coefficients, needed to encode sources into ambisonics, are pre-computed and retrieved based on the current angle of the sound source. Encoding in ambisonics, is the process of taking a raw audio signal and creating a sound-field from it, with the audio at a particular position[23]. Since the angle of the source is not known a priori, an entire quadrant of the SH is computed. In order to provide smooth transitions between coefficients interpolation is applied. The use of a LUT is a common technique in audio software applications which improves the speed of the algorithm on slower processors.

Exploiting Symmetry

This paper points out the possibility of exploiting SH symmetries in order to reduce the memory requirements of the system. Gorzel et al. note that SHs have well-known symmetries, making it possible to derive all coefficient from a single quadrant, determined by the azimuth and elevation angle of the sound source. By simply calculating the front-left-top quadrant of the sphere, all the other values can be calculated simply by applying sign changes. This memory reduction means faster loading times for the user, and consequently allows people with less powerful devices to experience these works.

Assuming Head Symmetry

Another technique employed by the authors, involves reducing the number of convolutions required for binaural synthesis by assuming left/right hemispherical symmetry. Consider a binaural sound source at 45° to the left of the subject, with

[23]Encoding ambisonics should not be confused with parametric compression encoding.

an elevation of 0°. If we were to simply swap the left and right channels then, in essence, it would be like presenting the sound source at 45° to the right of the subject[24]. By extension, a single HRIR is used for sources with azimuth 0°. By assuming symmetry, the number of convolutions is reduced by half, which allows for a smoother experience on cheaper hardware. This method is also adopted iby other authors (e.g. [PPQ16]).

HRTF Expansion

Gorzel et al. use a spherical domain representations of the SADIE HRTF dataset [KD15][25] in order to further reduce the number of convolutions required. In this approach, a *decoding matrix* is first calculated based on the positions of the desired IRs. A matrix of impulse responses, corresponding to the same directions, is then multiplied by the transpose of this matrix, which effectively encodes the IRs into the spherical domain. This method, ubiquitous today in binaural ambisonic reproduction, is called the *Least Squares* (LS) solution to the system of linear equations.

The decoding matrix is found via a *Moore-Penrose pseudo-inverse* of the matrix **L**, shown in Equation 3.1[26]. As we can see, each column of our matrix corresponds to a direction, and each row corresponds to a harmonic. The values are calculated using our standard real-valued ambisonic equation[27]:

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

$N_n^{|m|}$ corresponds to the normalization[28], and $P_n^{|m|}$ corresponds to the associated

[24] As noted, this assumes that the head is exactly symmetrical; further studies are required to see if this feature degrades the sound quality of the renderer substantively, especially when using personalized HRTFs.

[25] In lieu of the traditional virtual speaker approach.

[26] Matrix **L** is the SH representation of our HRTF directions.

[27] The equation is unnumbered since it is discussed in detail in Chapter 2.

[28] SN3D in this case.

Legendre polynomials, which are solutions to the wave equation, generally computed using numerical libraries[29].

$$\mathbf{L} = \begin{bmatrix} Y_0^0(\Phi_1, \Theta_1) & Y_0^0(\Phi_i, \Theta_i) & \dots & Y_0^0(\Phi_N, \Theta_N) \\ Y_1^{-1}(\Phi_1, \Theta_1) & Y_1^{-1}(\Phi_i, \Theta_i) & \dots & Y_1^{-1}(\Phi_N, \Theta_N) \\ \vdots & \vdots & \vdots & \vdots \\ Y_n^m(\Phi_1, \Theta_1) & Y_n^m(\Phi_i, \Theta_i) & \dots & Y_n^m(\Phi_N, \Theta_N) \end{bmatrix} \tag{3.1}$$

Computing the Moore-Penrose pseudo-inverse is given by [GAK+19] as:

$$\mathbf{D} = \mathbf{L}^\dagger = \mathbf{L}^\mathbf{T} \left(\mathbf{L}\mathbf{L}^\mathbf{T}\right)^{-\mathbf{1}}. \tag{3.2}$$

Any matrix, regardless of its dimensions, can be multiplied by its transpose to yield a square matrix - transposing is as trivial as swapping rows and columns. The inverted matrix[30] is subsequently multiplied with a transposed version of the original matrix $\mathbf{L}$. The transposed matrix $\mathbf{D}$, the decoding matrix, is used to expand the HRTFs into the the spherical harmonic domain. Equation 3.3 shows the HRFT spherical harmonic expansion.

$$\hat{\mathbf{H}} = \mathbf{H}\mathbf{D}^\mathbf{T} \tag{3.3}$$

The result is a matrix with $(N+1)^2$ columns, and as many rows as there are filter coefficients. This matrix can directly be convolved with the B-Format ambisonics spherical harmonics (e.g. these two matrices have the same number of harmonics). The binaural signals are then the result of the inner product (e.g. convolution) between the ambisonic coefficients and the SH coefficients of the HRTFs [PPQ16]. In the case of a 3OA[31] signal and a 26-point Lebedev grid[32] the number of convolutions

[29] Refer to Chapter 2 for more information on both normalization and Legendre polynomials.

[30] Inverting the square matrix involves Gaussian elimination. This can be done by hand but for large matrices we usually use computing libraries to perform the task.

[31] Shorthand notation for third order ambisonics.

[32] Lebedev quadrature is a special sampling scheme for spherical harmonics in which the number

is reduced from 26 to 16 resulting in a ∼62% reduction in the number of convolutions.

An added benefit of this method is that any *shelf-filtering*, such as a *MaxRe* shelf-filtering, can be done offline, rather than real-time during decoding. This reduces the required amount of CPU and as a result improves speed on low-end mobile devices [GAK+19] (e.g. since the decoder is known ahead of time, we can filter the $\hat{\mathbf{H}}$ matrix in advance). The same process is implemented in Politis et al. [PPQ16], however, instead of using *MaxRe* decoding filters, the authors describe using the AllRAD [ZF12] method for creating the decoding matrix, which is then binauralized using a VLS approach.

It should be noted that Gorzel et al. make no mention of the well-known timbral issues with the LS method (see [SZH18]). They also make no mention of alternate decoding strategies such as *mode-matching*, *AllRAD*, or EPAD [ZPN12][33]. A possible improvement to Resonance, therefore, could be to incorporate better binaural filters, based on the *Magnitude LS* (MagLS)[34] described in [DMKH+20], and test the effect of other decoding methods described in [PPQ16].

Additional Functions

In addition to all these optimizations, Resonance also has methods for controlling the sound source spread, simulation of near-field sources, and, the reverberation of simulated rooms. One of the contributions Gorzel et al. suggest is allowing for personalized HRTFs; the *Spatially Oriented Format for Acoustics* (SOFA) has become the standard for personalized HRTF integration - it is described in [MIC+13] by Madjak et al. The source code for Resonance includes MATLAB, JS, and C++ code, and the Github repository shows how to build these components to target

and location of points, along with integration weights, are determined by enforcing exact integration of spherical harmonics up up to a given order. T-designs and Fliege grids are other examples of sampling schemes that enforce exact integration.

[33] Although for regular designs these might be equivalent.

[34] [SZH18] also describes the *Time Alignment* (TA) and *Spatial Re-sampling* (SPR) methods, which Resonance does not yet offer.

different platforms making it highly flexible. The code is open-source and licensed under Apache 2.0.

Conclusion

While the Resonance web audio solution is extremely comprehensive, unfortunately, it also requires in-depth knowledge of web development to be implemented. While this lower level representation does provide more flexibility, a set of web tools and examples that simplifies the creation of spatial audio using Resonance could be useful. Much like *THREE.js* provides a higher-level implementation of the underlying OpenGL methods, we believe a set of examples for environments like *A-Frame* or *Babylon.js* could help more artists include Resonance into their web audio projects.

Unlike other libraries, Resonance goes much further in terms of optimizing its functions for lower-end devices. The other benefit of this framework is that it goes far beyond standard ambisonic functions, including much more complex processing such as reverbs, source-shaping, and *near-field compensation* (NFC). Unfortunately, since Resonance is tied to a commercial entity, there is always the possibility that other products by the same company, such as mobile devices, might be exclusively supported.

3.5.2 JSAmbisonics

Politis et al. [PPQ16] describe another implementation of an open-source WebVR solution in their 2016 publication. In this paper, the authors outline an ambisonic library built using the WAA, written in JS, for interactive spatial audio rendering. The library supports HOA, and implements the most fundamental processing blocks for generating and reproducing sound scenes, such as rotations and binaural decoding. In contrast to Resonance, this project was not developed by any company, but rather students from various academic institutions. In the next few sections we will

provide more detail regarding the transforms implemented in JSAmbisonics, and how these can be used to produce and deliver 3D web audio.

Ambisonic Rotation

Ambisonic rotation is an integral part of any binaural ambisonic system. This operation allows the sound to change dynamically in response to users' head movements. Rotating the sound-field in response to user head movement, in binaural decoding systems, has been shown to improve externalization, reduce front-back confusions, and generally improve localization [DW20]. Ambisonic rotation in FOA is trivial, and reduces to the following equations in $\mathbb{R}^3$ [KZ14][35].

R_z defines the counterclockwise rotation matrix about the z-axis.

$$R_z(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.4}$$

R_y defines the counterclockwise rotation matrix about the y-axis.

$$R_y(\beta) = \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix} \tag{3.5}$$

R_x defines the counterclockwise rotation matrix about the x-axis.

$$R_x(\gamma) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{pmatrix} \tag{3.6}$$

[35] We opt for α, β, γ to avoid confusion between rotation angles and ambisonic panning angles. The reference uses a different notation. Note also that we are referring to sound-field rotation after encoding.

The letters α, β, γ correspond the to the *yaw*, *pitch* and *roll* angles respectively[36]. When multiplying with our signal matrix, the order of harmonics will affect the results. In this case, the signals should be ordered $[x, y, z]$, and then re-ordered if necessary[37]. These three rotation matrices can also be combined into a single larger matrix by multiplying them together. Note that the order in which these transforms are combined is also important - the convention we use in Equation 3.7 is roll, pitch, yaw. The combined formula, resulting from multiplying these three matrices in this order yields [LaV06]:

$$R(\alpha, \beta, \gamma) = R_z(\alpha) R_y(\beta) R_x(\gamma) = \begin{pmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{pmatrix} \tag{3.7}$$

Care should be taken to understand the direction of rotation. Counter-intuitively, what we seek is for the ambisonic rotation to compensate for the listener's head movement, not follow it. Picture a sound source at 45° clockwise, if the user moves his or her head 45° clockwise, then, from their new perspective - if there is no rotation applied - the source will remain at 45°. However, the new position should now be 0°. Thus, the clockwise rotation of the user, should result in a counterclockwise ambisonic rotation.

Rotations in $\mathbb{R}^4$ and above are far more complicated, but necessary to move beyond FOA. There are three common methods used to accomplish this task. The first involves computing a single higher order rotation of ± 90 in R_y. R_z rotation matrices of higher orders are relatively easy to compute. Kronlachner's thesis [Kro14b] shows the following derivation of this matrix up to second order.

[36] The coordinate system for these matrices has the x-axis pointing towards the front, the y-axis pointing towards the left, and the z-axis pointing to the right.

[37] Rotating the 0th order harmonic has no effect.

$$\boldsymbol{R}_z(\alpha) = \left(\begin{array}{c|ccc|ccccc|c} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{0} \\ \hline 0 & \cos\alpha & 0 & \sin\alpha & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{0} \\ 0 & -\sin\alpha & 0 & \cos\alpha & 0 & 0 & 0 & 0 & 0 & \\ \hline 0 & 0 & 0 & 0 & \cos 2\alpha & 0 & 0 & 0 & \sin 2\alpha & \\ 0 & 0 & 0 & 0 & 0 & \cos\alpha & 0 & \sin\alpha & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \mathbf{0} \\ 0 & 0 & 0 & 0 & 0 & -\sin\alpha & 0 & \cos\alpha & 0 & \\ 0 & 0 & 0 & 0 & -\sin 2\alpha & 0 & 0 & 0 & \cos 2\alpha & \\ \hline \mathbf{0} & & \mathbf{0} & & & & \mathbf{0} & & & \cos 3\alpha \\ & & & & & & & & & \ddots \end{array}\right) \tag{3.8}$$

Kronlachner's technique, adopted from Zotter's thesis [Kro14b], involves using a single R_y rotation matrix to position the spherical harmonic for variable rotation using the more easily computable R_z matrix in intermediary steps[38]. E.g.

$$R(\alpha, \beta, \gamma) = R_z(\alpha + 90) \cdot R_y(90) \cdot R_z(\beta + 180) \cdot R_y(90) \cdot R_z(\gamma + 90) \tag{3.9}$$

These 90° rotation matrices can be found at IEM's website[39]. Figure 3.1 shows the head rotations according to Equation 3.9 when α, β, γ are all set to 0°.

We see clearly that the starting position on the left, returns to its original position after these 5 respective rotations[40]. From a strict implementation perspective, we highlight this technique because we find it easier to understand than the derivation

[38] See Equation 3.9. Please note the matrix in Equation 3.8 assumes your signals are ordered according to ACN.

[39] Site accessed April 1, 2021.

[40] The positive angle should have likely been a counterclockwise rotation. The image is purely demonstrative.

of HOA rotation matrices. It is not clear at this moment from our research if there is significant latency created by the added rotations, or if in fact this method is faster than its counterparts[41] due to the reduced number of R_y calculations, and zero R_x calculations.

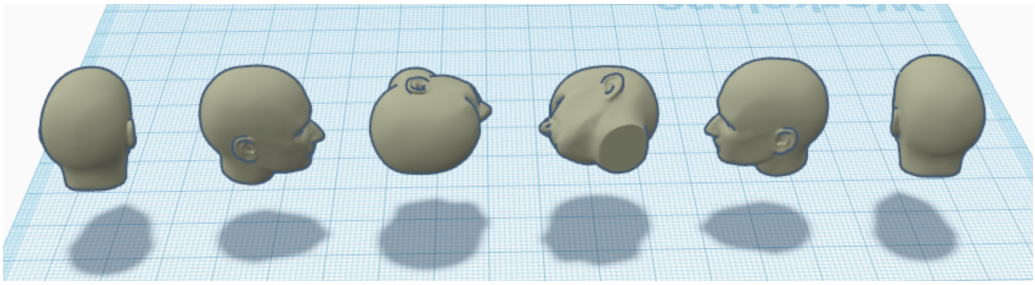


Figure 3.1: Rotation Using Single R_y Matrix

Politis [PPQ16] and other authors (i.e. [GAK+19]), instead compute all real-valued rotation matrices using a recursive algorithm. Spherical harmonics are not only found in the audio domain, but are pertinent in chemistry, physics, and mechanics. The rotation matrices derived in Politis's work come originally from Ivanic and Ruedenberg's paper in "The Journal of Physical Chemistry" [IR96].

The details of these calculations are outside the scope of this chapter, however, implementations of these formulas can be found in Politis's "Spherical Harmonic Transform" (SHT) library[42]. Politis has implemented these formulas both in JS and in MATLAB. Gorzel et al. [GAK+19] have implemented these in numerous other computer languages.

For those wishing to write these from scratch, we recommend the paper by Blanco et al. [BFB97], which contains pseudo-code and reasonably straightforward formulas. We should also note the existence of rotation matrices using complex SH, which have been shown to be easier to compute. These however, require bringing the entire sound-field to the frequency domain, which might prove computationally expensive;

[41] Including rotation in using complex SHs.
[42] See link.

achieving low latency is crucial for smooth, error-free, delivery of binaural audio [DW20].

These HOA rotation matrices allow us to binaurally decode HOA audio, taking into account the dynamic head movements of the user. This substantially increases the realism of the works, and represents modern approach to the dissemination of spatial audio. More studies are required to evaluate how the three different techniques discussed affect the quality of the reproduction[43]. This operation is pivotal for the low-cost dissemination of spatial music featuring similar properties as those accomplished by speaker arrays.

Ambisonic Reflection

Reflection is a simple operation in the ambisonic domain given the symmetry of spherical harmonics. Reflection, or mirroring, refers to the ability to rotate the sound-field 180° along any of the three coordinate planes (yz, xz, or xy). This simplifies to polarity changes of specific harmonics, based on whether they are symmetric or anti-symmetric[44]. Politis provides Equation 3.10 to accomplish this task.

$$\begin{aligned} (m < 0 \wedge m \text{ even }) \cup (m \geq 0 \wedge m \text{ odd }) &: \mathbf{yz} \\ m < 0 &: \mathbf{xz} \\ (n+m) \text{ odd } &: \mathbf{xy} \end{aligned} \tag{3.10}$$

This might be desired, for example, if a microphone is suspended from the ceiling in a concert hall, and we now wish to use the recording in a different orientation. Another other case scenario might be when 360° camera footage and a sound-field are misaligned by exactly 180°. For each case when the desired harmonics are found, a multiplication by -1 results in a change of polarity, giving us our desired result. In

[43] Kronlachner's technique has not yet been implemented in a web-based solution, to the best of our knowledge.

[44] All harmonics are one or the other.

other words:

1. For rotation about the **yz** plane, we must find all harmonics with ambisonic degree m that are smaller than 0 and even, as well as those that are greater than or equal to 0 and odd. This is a front-back flip.

2. For a rotation about the **xz** plane, we must find the harmonics that have ambisonic degree m smaller than 0. This is a left-right flip.

3. For a rotation about the **xy** plane, we must find the harmonics where the sum or ambisonic degree n and m is odd. This is a up-down flip.

Ambisonic Beam-forming

Beam-forming refers to the ability of steering an audio signal, either at the capture or reproduction stage, by using a series of delays and gain adjustments. In the spherical harmonic domain it is possible to create adjustable *virtual microphones* pointing in arbitrary directions using a linear combination of SHs. This is useful for example when we want to isolate a particular sound inside the sound-field. This technique is also the basis of ambisonic decoding. This transform results in a monophonic signal, these monophonic signals, however, can be re-encoded into the SH domain[45] - if desired - or be used for a conventional stereo mix-down. Equation 3.11 shows the equation provided by Politis [PPQ16] for this task:

$$x_{\mathrm{vm}}\left(t, \boldsymbol{\gamma}_0\right) = \mathbf{w}^{\mathrm{T}}\left(\boldsymbol{\gamma}_0\right) \mathbf{b}(t) \tag{3.11}$$

where,

$\boldsymbol{\gamma}_0$ corresponds to the orientation of the virtual microphone,

$\mathbf{w}(\gamma_0)$ corresponds to the $(N+1)^2$ vector of beam-forming weights[46], and,

[45] This is done for example in the Sparta suite discussed in Chapter 1.
[46] As a reminder, N, is the ambisonic order.

$\mathbf{b}(t)$ corresponds to the ambisonic spherical harmonics, otherwise known as B-format signals (at time t)[47].

The library allows for various different types of microphone patterns with different front/back *rejection ratios*[48]. Politis [PPQ16] provides additional equations for the derivation of specific polar patterns including: *cardioid*, *hypercardioid* and a *max-rE* which "maximizes the acoustic intensity vector in an isotropic (e.g. equal in all directions) diffuse field." The max-rE measure is prevalent in the ambisonic literature.

The $\mathbf{b}(t)$ term is obtained by taking the outer product of your original source signal and the vector $\mathbf{y}$ of SH coefficients, evaluated for the azimuthal and elevation angle desired. If the two vectors have dimensions n and m, then their outer product is an n by m matrix[49]. $\boldsymbol{\gamma}_0$ may either be a triplet of Cartesian coordinates, or a duple containing azimuth ϕ and elevation θ.

The outer product can be defined as:

$$\mathbf{s} \otimes \mathbf{y} = \mathbf{B} = \begin{bmatrix} s_1 y_1 & s_1 y_2 & \dots & s_1 y_n \\ s_2 y_1 & s_2 y_2 & \dots & s_2 y_n \\ \vdots & \vdots & \ddots & \vdots \\ s_m y_1 & s_m y_2 & \dots & s_m y_n \end{bmatrix} \tag{3.12}$$

where $\mathbf{s}$ is the raw signal to be encoded, $\mathbf{y}$ is the vector corresponding to the sampled SHs for the given coordinate, and $\mathbf{B}$ is the resulting B-format signal. The weight vector $\mathbf{w}^T$ is arranged to match the indexing of the spherical harmonics and can be expressed as a "pattern-dependent" part, and a "rotation-dependent" part [PPQ16]. E.g.

[47] Using vector notation to denote a block size of 1.

[48] Measure of how much sound incident from the front is captured relative to sound incident from behind.

[49] n and m here should not be confused with indices for spherical harmonics. These values refer instead to the number of rows and columns of the product $\mathbf{B}$.

$$[\mathbf{w}(\gamma_0)]_q = w_{nm} = c_n Y_{nm}(\gamma_0) \tag{3.13}$$

There are (N+1) c_n coefficients, which are derived according to the desired properties of the virtual microphone. For each ambisonic SH order n, all harmonics of ascending degree share the same "pattern-dependent" coefficient. The "rotation-dependent" coefficient, is unique for each harmonic[50]. The subscript q refers to the ACN index of the spherical harmonic, which can be expressed as:

$$\begin{aligned}[\mathbf{y}(\boldsymbol{\gamma})]_q = Y_{nm}(\boldsymbol{\gamma}), \text{ with } q = 1, 2, \ldots, (N+1)^2 \\ \text{and } q = n^2 + n + m + 1.\end{aligned} \tag{3.14}$$

In JSAmbisonics a number of different types of virtual microphone responses have been implemented including cardioid, hypercardioid and *max-rE*. A few formulas are provided by the author to recreate these patterns via spherical domain beam-forming. Such as:

$$\begin{aligned}\text{cardioid} \quad &: c_n = \frac{N!N!}{(N+n+1)!(N-n)!} \\ \text{hypercardioid} \; &: c_n = \frac{1}{(N+1)^2} \\ \max -\text{rE} \; &: c_n = \frac{P_n(\cos\kappa_N)}{\sum_{n=0}^{N}(2n+1)P_n(\cos\kappa_N)}\end{aligned} \tag{3.15}$$

where,

$\kappa_N = cos(2.407/(N + 1.51))$ given in [ZF12],

N is maximum the ambisonic order,

n is the ambisonic order of harmonic with index q, and

[50] Politis uses the standard ambisonic coordinate system $\gamma = [\cos\phi\cos\theta, \sin\phi\cos\theta, \sin\theta]^{\mathrm{T}}$.

P_n are the associated Legendre functions[51].

Some possible uses cases for these virtual-microphones include: defining custom decoders, isolating a speaker during live web-conference, or, to musically modify an individual part of the sound-field. The beam-forming technique is used in other libraries together with a re-encoding to create a whole new sound-field from the decomposition of an original one. One can also localize source directions automatically, and thus create a system that allows one to listen to individual musicians within a sound-field.

Decoding

Three main decoding strategies are discussed in [PPQ16]: *Sampling* (SAD), *Mode-matching* (MMD) and *AllRAD*[52]. Most web browsers do not support multi-channel audio past 5.1 [Web], however, these methods can still be implemented in the generation of binaural decoding matrices offline. Politis provides three equations which correspond to three well-known decoding methods:

$$\begin{aligned} \text{Sampling} &: \mathbf{D}_{\text{ls}} = \frac{1}{L}\mathbf{Y}_L^{\text{T}} \\ \text{Mode} - \text{matching} &: \mathbf{D}_{\text{ls}} = \left(\mathbf{Y}_L^{\text{T}}\mathbf{Y}_L + \beta^2\mathbf{I}\right)^{-1}\mathbf{Y}_L^{\text{T}} \\ \text{AllRAD} &: \mathbf{D}_{\text{ls}} = \frac{1}{N_{\text{td}}}\mathbf{G}_{\text{td}}\mathbf{Y}_{\text{td}}^{\text{T}} \end{aligned} \tag{3.16}$$

For all of these $\mathbf{D}_{\text{ls}}$ corresponds to the decoding matrix which generates loudspeaker signals[53] via a linear combination of SH ambisonic signals. E.g.

$$\mathbf{x}_{\text{ls}}(t) = \mathbf{D}_{\text{ls}}\mathbf{b}(t)$$

[51] Solution to the Legendre equation, encountered when solving Laplace's equation in spherical coordinates. Laplace's equation is a special case of Helmholtz equation which is used to describe wave and diffusion behaviors.

[52] All Around Ambisonic Decoding.

[53] In this case virtual loudspeakers, or speaker directions corresponding to HRTF measurement coordinates.

.

Where $\mathbf{x}_{\text{ls}}(t)$ is a vector with a single sample for each loudspeaker.

The sampling approach (SAD) is the least sophisticated, and only works well for regular arrangements. Given its well-known deficiencies, it won't be elaborated upon any further. In the mode-matching approach, the solution involves a *Least Square* (LS) solution with a regularization value β. Zotter [ZFP13] suggests a value of $\beta = \frac{1}{4}\frac{\text{G}}{\pi}$, where G is the number of speakers. They describe the regularization parameter β as the "reciprocal of a suitable scalar times the mean angular segment covered by each loudspeaker." $\mathbf{I}$ corresponds to the identity matrix which will diagonalize β.

The AllRAD method [ZF12] refers to the *All Round Ambisonic Decoding* method which combines *Vector-Based Amplitude Panning* (VBAP) and ambisonic decoding into a single unified system. Rather than using the real directions of the speakers, the SH matrix is designed using coordinates from a t-design[54].

An additional $\mathbf{G}$ matrix is designed based on the VBAP method, also using the same t-design coordinates as initialization values for the VBAP algorithm. Finally, we generate G vectors which adjust the gain of our "real" speaker signals, located inside these t-design triplets - using a "virtual" t-design configuration[55]. This is done in order to achieve direction-independent energy and spread in arbitrary ambisonic layouts [ZF12].

All these designs differ in sound quality. Having them all available in the JSAmbisonic kit, allows users to build state-of-the-art binaural decoders using the best sounding decoding methods. It also allows for experiments to be conducted to further assess the quality of each under different constraints.

Two objective measures are used to validate these decoders: *total signal power* and *angular spread*. Equations 3.17 and 3.18 show these two metrics. Total signal power is given by:

[54] [HS96], t-designs can be found here.

[55] The real coordinates of sources are passed to the VBAP algorithm, to calibrate the gain producesd in the initial decoding.

$$\hat{E} = \frac{4\pi}{\mathrm{L}} \sum_{l=1}^{\mathrm{L}} g_l^2 = \frac{4\pi}{\mathrm{L}} \|\boldsymbol{g}\|^2, \tag{3.17}$$

where g_l correspond to the gain weights for our L loudspeakers. Angular spread is given by:

$$\hat{\sigma}_{\mathrm{E}} = 2 \arccos \left\{ \|\hat{\boldsymbol{r}}_{\mathrm{E}}\| \right\} 180^\circ / \pi, \tag{3.18}$$

where,

$$\hat{\boldsymbol{r}}_{\mathrm{E}} = \sum_{l=1}^{\mathrm{L}} \boldsymbol{\theta}_l g_l^2 / \sum_{l=1}^{\mathrm{L}} g_l^2. \tag{3.19}$$

$\boldsymbol{\theta}_l$ corresponds to the direction vector triplet for each loudspeaker. The $\boldsymbol{r}_{\mathrm{E}}$ vector is a recurring measure in ambisonic literature[56], described in [ZF12] as the center of gravity, or *centroid*, of the sound-field. These measures have been used in simulated experiments to determine the predicted performance of these various decoders. In ambisonics, in contrast to other methods, there is both an encoding and decoding stage, thus, in order to find our speaker weights, we need to account for both stages[57].

The AllRAD method was ultimately selected in JSAmbisonics [PPQ16] due to its improved sound quality. It is included in the $\mathbf{D}_{\mathrm{bin}}(f)$ binaural decoding matrix described subsequently using Equation 3.20. In Section 3.5.1 we discussed binaural decoding using SH expansion of HRTF data-sets. For sake of completeness, in this section we will talk about the *virtual loudspeaker* (VLS) approach. As noted earlier, both methods can be used to generate the binaural decoding filters offline, since the decoder layout is known a priori. These filters can also be equalized, if need be, beforehand.

In the virtual loudspeaker approach, we imagine that we are decoding to an array of real loudspeakers, and find a suitable decoding matrix according to this criteria.

[56] Used in the design of MaxRe decoders.
[57] See Section 6.1 in [ZF12].

Any of the three aforementioned decoding methods are valid here.

Subsequently, we derive our matrix of decoded signals, and match each speaker direction with its corresponding HRTF from our data-set. Naturally, this means that the decoder will be designed according to the HRTFs present in the set. The main constraint in this method is that the number of decoding directions must be larger than the number of harmonics, in order to accurately represent the sound-field[58]. With the SH HRTF expansion, the spherical sampling scheme must contain more points than the number of harmonics at that order, or at least be equal.

Politis et al.[PPQ16] provide Equation 3.20 describing the *virtual loudspeaker* (VLS) approach to binaural decoding in the frequency domain. E.g.

$$\mathbf{x}_{\text{bin}}(f) = \mathbf{H}_{\text{LR}}(f)\mathbf{D}_{\text{vls}}\mathbf{b}(f) = \mathbf{D}_{\text{bin}}(f)\mathbf{b}(f) \tag{3.20}$$

where,

$\mathbf{x}_{\text{bin}}(f)$ corresponds to the 2 by 1 binaural bins at frequency f[59],

$\mathbf{H}_{\text{LR}}(f)$ is the 2 by G[60] matrix of L/R HRTF filters suitably arranged,

G is the number of decoding directions, and

$\mathbf{D}_{\text{vls}}$ is the G by $(N+1)^2$ virtual loudspeaker matrix (*not in the frequency domain*).

Additionally, $\mathbf{b}(f)$ is the $(N+1)^2$ by 1 encoded audio signal in the frequency domain, where each value is a single bin of a single harmonics[61], and $\mathbf{D}_{\text{bin}}(f)$ is the binaural decoding matrix for the virtual loudspeaker approach. Much like in Gorzel et al. [GAK+19], the number of HRTFs is reduced by 50% in this implementation by

[58] If the decoder does not meet this criteria, multiple orders of SHs can be discarded at the expense of spatial resolution.

[59] One for each ear.

[60] Where each row corresponds to a direction, and each column to an ear.

[61] The paper uses **a** but we prefer **b** to denote B-format signals.

assuming L/R symmetry. The binaural signals filtered with the proper HRTFs are then delivered to the client using the *Web Audio API* (WAA) browser specification.

As already discussed, SH expansion of HRTF data has timbral problems which have since then been addressed in other publications [SZH18], but not implemented in the JSAmbisonic kit. The VLS method provides us with the flexibility to decode HOA using alternate strategies, which might provide better sound quality. This library, therefore, provides us with a way to test out and compare various binaural decoding methods, in order to determine if optimized SH methods, for example, can achieve improved sound quality compared to VLS binaural decoding. It remains to be seen if the additional computational load of VLS can take precedence over the efficiency of the direct approach (e.g. SH HRTF expansion).

Acoustic Visualization

Politis also implemented some examples of sound-field visualization to help the user verify that their sound-field is suitably composed. In "Parametric Spatial Audio Effects" [PPP12], Politis et al. describe how to analyze sound-fields in order to visualize them[62]. Politis notes one possible improvement to JSAmbisonics could be to use the biquad filter structure in the WAA to perform frequency-dependent visualizations - using the *intensity* and *diffuseness* analysis. Various authors have pointed out that sound localization is a bi-modal task, requiring both sight and vision. Visualizing the sound-field can help listeners better localize and track moving sources during a musical performance, and provide an additional layer of stimuli to engage viewers of these multi-media works.

[62]The mathematical details of this are covered in Chapter 1.

Decoding Filter Generation (SOFA)

The very last section of this paper discusses the generation of decoding filters which allows for flexible HRTF selection. The included binaural decoding filter is based on the LISTEN HRTF sets [LIS] and derived using the AllRAD method [ZF12], as noted before. The author tested both the VLS approach and the HRTF SH expansion method, and found that the former was subjectively superior.

The *MagLS* method has been proposed to improve these timbral deficiencies [SZH18]. Another possible addition to this toolkit could be the Energy Preserving Ambisonic Decoder (EPAD) [ZPN12], which could be measured up against AllRAD. Finally, the author highlights the possibility of importing SOFA formatted HRTF sets dynamically using the OpenDAP SOFA server proposed in [Car15]. Currently, one can import HRTFs for local development or select from a small included database. The goal of the OpenDAP SOFA server is to have access to hundreds of HRTFs, via a remote server, which one could select from.

Conclusion

This toolkit is one of the most popular ways researchers are creating ambisonic tools online. It is comprehensive, has good documentation, and is openly licensed. Using these tools, other authors have already contributed to the development of systems for the dissemination of ambisonic audio (e.g. [DMKH+20]). Since the toolkit makes use of the WAA, it can be suitably integrated with hundreds of WAA projects already public online, including musical instrument apps, web-based DAWs, or interactive experiences.

Much like other JS toolkits, some programming experience is required in order to make use of these functions. The demos provided by the author almost all worked well, with the exception of a few including the WebGL integration; this might be one possible way to contribute to the project.

One of the demos we found promising was the multi-channel HOA player, which allows one to stream to speaker arrays from the browser, however, that demo also seems to need support/maintenance. This feature would allow us flexibility when users have surround sound systems, as they would not be limited to 3D audio via binaural synthesis. Future sections of this chapter will highlight some projects that have made use of JSAmbisonics.

3.5.3 BiLi: JS Binaural Listening

As noted in Section 3.4.1, Carpentier [Car15] provides a good overview of the strengths and deficits of the WAA. In his paper, the author discusses how to extend the WAA to allow for personalized HRTFs, one of the perceived deficits of the API. He also discusses how to generate simulated spatial audio via IIR[63] filters, in what he calls a parametric approach to virtual surround sound, within the context of HRTF interpolation-based methods for spatial audio[64].

In addition, the author provides a lengthy discussion on suitable methods for retrieving HRTF data-sets from servers using the SOFA architecture [MIC+13]. This work is part of the BiLi project which is a collaboration between IRCAM and several other French institutions. The HRTF personalization improvements rely on native nodes from the WAA. IRs are loaded into *AudioBuffers*, convolved with *ConvolverNodes*, and cross-faded using *GainNodes*. In contrast to binaural decoding of ambisonic signals, in this implementation the author is employing a *nearest neighbor* approach that interpolates between HRTFs as the source moves in space. Carpentier notes that a *k-d tree* would improve the performance of this algorithm[65].

The internal WAA HRTF spatialization method also uses interpolation rather

[63] Infinite Impulse Response.

[64] Since this method is vastly different than the techniques we are researching, a deep discussion into this technique will be omitted. However, this is another important way that spatial audio can be optimized for the web.

[65] Complexity $\mathcal{O}(\log n)$ instead of $\mathcal{O}(n)$.

than ambisonics to position sound sources in space. Other popular tool-kits, such as the *Spat* package from IRCAM, written for MAX/MSP, also have an option for spatialization via nearest neighbor. The efficiency of this algorithm depends largely on the number of sources, in contrast to the ambisonic method.

The *Nearest Neighbor Search* (NNS) algorithm is commonly used in classification problems to find the closest data point in a group that satisfies a set of conditions. In the case of HRTF interpolation, the NNS is used to find the HRTF(s) closest to a particular direction, given the available set. The k-d tree is an extension to NNS, in which the group of points are arranged using a binary tree, in order to more efficiently find the solution to the search problem. For example, the right side of the tree might correspond to a particular hemisphere, causing the algorithm to immediately ignore half the possible solutions, rather than calculating every single distance in the data space.

The final part of the paper discusses the SOFA [MIC+13] architecture in greater detail, and how it can be used to archive and retrieve personalized HRTFs from a remote server. Carpentier describes how a web server can be configured in order to accept URL requests from networked applications. Specifically, the author notes that the SOFA specification makes it compatible with OpenDAP (Open Data Access Protocol), which "allows for data transport over HTTP."

> "More specifically an OpenDAP server can host a collection of data (typically in HDF or netCDF[66] format) and serve them to remote clients such as web browsers, web applications, graphics programs or any DAP-aware software."

HRTF personalization is an ongoing subject of research, however, improvements in localization have been shown when using personalized HRTFs - especially with regard to elevation [KJM08]. It has also been shown that subjects can adjust to generic HRTFs after a certain period of time. However, being able to import your

[66]SOFA relies on netCDF.

own equalization profiles is undoubtedly better acoustically, if possible, for immediate immersion.

Unfortunately, the process of creating one's own HRTF set is still very expensive and time-consuming. Recently, in 2020, Guezenoc and Séguier [GS20] published a complete survey of HRTF Individualization discussing the most up-to-date work involving four different methods:

1. **Acoustic measurement:** impulse responses are gathered using in-ear microphones and grids of loudspeakers.

2. **Numerical simulation:** using 3D scans of human subjects. 3D scans can be taken using low-cost consumer grade smartphones [KSS16].

3. **Indirect individualization based on anthropometric data:**

   (a) **Adaptation:** using anthropomorphic features of the subject, a generic, or pre-existing HRTF set is modified.

   (b) **Selection:** using a data set which includes anthropomorphic information, a specific pre-measured HRTF set is selected.

   (c) **Regression:** the HRTF set is estimated using morphological measurements. Statistical approaches are used to derive the set. Some neural network approaches have been considered.

4. **Indirect individualization based on perceptual feedback:**

   (a) **Selection:** perceptual evaluation task is undertaken by the subject to help select the best HRTF set from the data set.

   (b) **Adaption:** based on responses from subjective evaluation, a HRTF set is adapted using frequency scaling, filter-tuning, or statistical-tuning. [67]

[67] Refer to [GS20] for more details on these three methods.

As we can see, HRTF personalization has received substantial attention in the last few years. We expect that all these JS libraries will improve in terms of their ability to suggest or generate HRTF sets over time. The BiLi project is not a suitable player or library for composers to utilize, but the discussion regarding the server architecture for requesting HRTFs is particularly useful for future FOSS solutions. The parametric methods for HRTF simulation also warrants more exploring, especially since it has been shown to reduce computational loads when compared to convolution-based spatialization. Unfortunately, Comunita et al. [CGLP19] also noted that the IIR simulation[68] compromises on sound quality at the expense of lower processing.

3.6 Players

As we noted earlier, many of these JS Libaries are not well suited for composers with limited technical backgrounds. As a result, some authors have developed different players for the dissemination of spatial music based on HOA. The goal of these projects is to simplify the process of sharing these musical works online making use of HTBA as a means of spatializing the recorded or synthesized sound-fields. In contrast to commercial players by large companies, these players can be hosted by Universities giving the authors full control of the materials disseminated.

3.6.1 HOAST

Deppisch et al. [DMKH+20] published a paper in 2020 describing the development of a Higher Order Ambisonic STreaming (HOAST) platform. This collaboration between IEM, Aalto, Hofer Web Solutions and Zylia, resulted in an open source[69] web player for 360° video and HOA. As the authors noted, most video players for

[68] Using the 3DTI Toolkit.
[69] GNU GPLv3 licence.

this type of media only allow up to 2OA[70]. The goal of this project was to create a video player capable of producing 4OA, with an associated VR video.

HOAST was built using the WAA, which is compatible with most modern browsers. One of the problems with all these systems is that despite their superior flexibility, there are still only certain browsers that support immersive media; luckily all these browsers are free to install, which removes any financial barriers. When it comes to support on mobile devices, not all platforms are equally supported, even when using the right browser[71].

In the case of HOAST, much like with other WebXR projects, Firefox and Chromium-based browsers are recommended. An added problem is that due to the infancy of this technology, there are still API changes happening consistently, which means that support is required to maintain these applications. Deppisch et al., for example, describe the deprecation of *ScriptProcessorNodes* for *AudioWorklets*, which despite their superiority, can cause glitches on some devices. Recently, the entire WebVR API was deprecated and replaced by the WebXR API.

Another consideration with HOAST, or any 360° video player for that matter, is the need for a large amount of data and processing power to deliver a proper experience. A single minute of 4K[72] video at 90fps[73] requires roughly 5.5GB of data if uncompressed. This means that HOAST, much like other players, might only work well in some high-end mobile devices.

The Web Audio API has limits on the number of audio channels that it supports for audio nodes. According to the authors of HOAST: "all Web Audio API implementations establish a 32-channel limit"; this, in turn, limits the Ambisonic audio to 4OA - which requires 25 channels. While it might possible to use multiple nodes for reproducing even higher orders, this is likely not suitable for web-based apps, which

[70]Second order ambisonics.
[71]WebXR is better supported currently in Android than in other mobile device OSs.
[72]This is the typical resolution of 360° videos.
[73]This is the preferred frames-per-second setting for VR content, as it reduces motion sickness.

require small files sizes and low CPU overhead.

CODECs, Container, and Signal Flow

In order to provide reasonable small file sizes a number of different compression algorithms were evaluated by Deppisch et al. The OPUS CODEC is one of the preferred compression systems for ambisonic audio on the web, given its large channel count and open-source nature. Some other popular file types which allow for multi-channel audio files include: WAV, FLAC, AAC, and VORBIS. All major browsers support the OPUS codec, with the exclusion of Apple's Safari Browser (version 13.1).

After an audio CODEC was selected, the next step was to determine which streaming platform would be adopted. The authors in this case used the MPEG-DASH player, which was selected due to its CODEC-agnostic nature. MPEG-DASH is an *adaptive streaming technology* which automatically adjusts media quality based on the users available bandwidth to ensure smooth playback. The VP9 video CODEC was also selected, along with the WebM container for combining both audio and video. The actual ambisonic manipulation in JavaScript is accomplished using the JSAmbisonics library by Politis et al. [PPQ16] discussed in Section 3.5.2.

Figure 3.2 shows the signal flow of HOAST. 4OA files, encoded with OPUS, are packaged in WebM and streamed using MPEG-DASH. On the client side, the DASH stream is received via *dash.js* and sent to an HTML[74] *audio element*. The OPUS decoded audio signals from the HTML audio element are then passed to the WAA's *audio context* via a `MediaElementAudioSourceNode`. These signals are then passed to custom processing nodes from JSAmbisonics [PPQ16], which perform the rotation and zoom based on the user's FOV. Finally, the Web Audio API convolver nodes are used for binaural decoding. The final two signals are sent to the `AudioDestinationNode`, the last node in the signal path, preceding the user. The ambisonic zoom relies on the cropping/windowing algorithm combined with the

[74]Hyper-Text MarkUp Language.

warping transform developed by Kronlachner [KZ14]. Both of these transforms are covered in more detail in Chapter 1.

Binaural Decoding

The *binaural decoding* implemented in HOAST is based on the solution to a *Least Squares* (LS) problem, much like the implementation by Gorzel et al. [GAK+19].

Deppisch et al. provide Equation 3.21, which is another way to symbolize the LS solution.

$$\omega_{\mathrm{LS}} = \arg\min_{\omega} \|\boldsymbol{Y}_{\Upsilon}\boldsymbol{\omega} - \boldsymbol{h}_{\Upsilon}\|_2^2 \tag{3.21}$$

The $(N+1)^2$ decoding filters ω_{LS} will minimize the least squares error between HRTF measurements in $\boldsymbol{h}_{\Upsilon}$ and the spherical harmonics in $\boldsymbol{Y}_{\Upsilon}$ evaluated at all HRTF directions in the measurement grid Υ [DMKH+20].

In this case a Lebedev grid[75] of HRTF measurements is used, and, in order to improve the perceptual decoding of binaural signals at high frequencies, the *Magnitude Least Squares* (MagLS) method is implemented. For low frequencies, a LS solution is applied.

In the MagLS method, the objective is to "approximate only HRTF magnitudes while ignoring the phase error [SZH18]." In that same publication, Schorkhuber et al. noted that other solutions had been proposed by various authors, including: diffuse-field equalization, using a sparse number of sampling points [76], and time-alignment using all-pass filters.

Schorkhuber et al. found that when evaluated for *Composite Level Loudness* (CLL), the MagLS method performed better than the post-equalized LS method, or the Time-Aligned method. In addition, the authors performed a listening experiment

[75] Link to Lebedev code.

[76] This Spatial Re-Sampling method is also discussed in Chapter 1 in more detail, as well as the Time Alignment (TA) method.

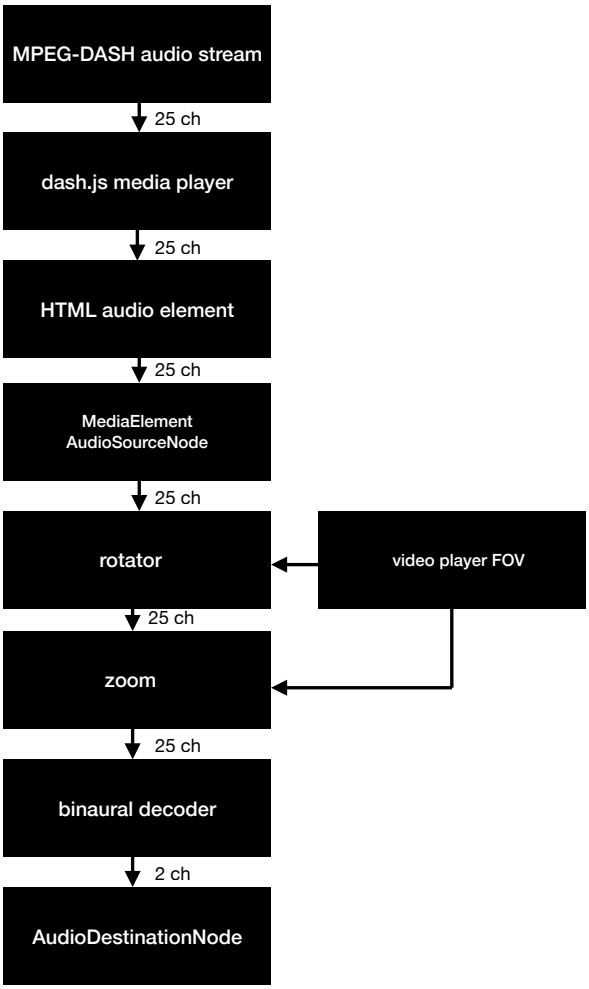


Figure 3.2: HOAST Signal Flow

to determine the cut-off frequency - since the MagLS approach performs a magnitude-only optimization only at high frequencies[77]. In HOAST, the MagLS optimization is pre-computed in the provided IRs[78].

Zotter's book [ZF19] offers a look into the implementation of the MagLS approach[79]. The *argmin* representation is purely demonstrative, instead, to yield the desired result, a modified version of the process described by Gorzel et al.[GAK+19] in Section 3.5.1 must be performed. Namely, for frequencies below 2kHz [SZH18] the traditional SH expansion from Equation 3.3 is performed. Above the critical frequency, the modified LS solution ignores the phase of the signal since ITDs[80] at high-frequencies do not significantly contribute to localization. Thus, for higher frequencies, the magnitude of the HRTF SH is used in lieu of the complex frequency values in the harmonic expansion. To avoid phase discontinuities, the phase of the bin last bin before equalization is extracted and applied to all subsequent equalized bins.

In addition to this MagLS optimization, the HOAST publication also describes the use of a FOA impulse response, generated using the *Image Source Method* (ISM), in order to improve externalization. This ISM impulse response is applied only to the first-order[81] decoding filters to reduce latency. The length of the higher-order decoding filters is 256 samples[82]. The ISM is a technique that can be used to generate synthetic room impulse responses[AB79][83].

[77] The authors found this frequency to be as low as 2kHz for most signals.
[78] There is no public code for the generation of these filters yet.
[79] Section 4.11.2
[80] Interaural Time Differences (e.g. phase).
[81] W, Y, Z, X harmonics in ACN ordering, namely.
[82] Shorter than the ISM filters.
[83] Detailed discussion of this technique is outside the scope of this chapter.

Video Rendering

In order to display 360° video along with the ambisonic audio, *Video.js* was used in combination with the `videojs-contrib-dash` plug-in [DMKH+20]. This allowed the authors to receive and play back MPEG-DASH audio/video streams while supporting equirectangular video[84] via a custom made plug-in called *videojs-xr* - an update to the pre-existing *videojs-vr* relying on updated WebXR API.

A polyfill[85] makes HMDs usable in Firefox and Chromium-based browsers. In Chromium-based browsers enabling the *#webxr* flag, setting the *#webxr-runtime* and disabling *#xr-sandbox* might be required. These can be found by typing `chrome://flags` in the URL text box.

Website Implementation

On the backend side, the HOAST web application allows basic CRUD (Create, Read, Update, Delete) tasks of the data. The database holds two main types of data: the meta data of the media files, and, the information related to the "channel" owner (i.e. IEM channel, Aalto channel, etc.). The python-based web framework Django was used for the back end, while for the front end, the Bootstrap toolkit was employed.

Currently only the creators are allowed to manage the data, but the code is freely available so anyone is capable of loading it on a new server to start their own library. One could also add a more complex back-end allowing any user to create their own "channel", but this has not yet been implemented.

[84] Equirectangular projection is a typical transform used to display 360 degree content in 2D.

[85] Polyfill is a library that: injects an XR implementation into a browser if one does not exist, patches browsers that have incomplete implementations of the API, and, provides a synthesized VR display when WebXR is not supported.

Conclusion

From the numerous projects we found involving distribution of ambisonics online HOAST is certainly one of the most promising. While it does not offer much in the way of experimentation or customization, this system allows one to distribute 360° videos with HOA without needing to rely on any media companies. This means viewers won't be bothered by advertisements, and publishers can retain full ownership of their content.

Upon exploring the numerous examples on the HOAST website, we found that the sound quality was very good, with the exception of some noticeable clicks when we rotate the sound-field too quickly[86]. The low quality of the video was distracting, but likely a good compromise given the reduced load-time. In addition, we noticed that many of the examples used a sound-field visualization, which helped localization, but could be improved by focusing the DoA image better and using a higher video resolution.

The drawback of this system is that the user requires very specific CODECs and file types for proper results. For an average musician, using **ffmpeg** to convert audio and video files to the right format might prove difficult[87]. Additionally for users that do not have experience with equirectangular images or video editing, the system does not allow one to submit music without a video. This makes it necessary to find a suitable image and convert it to a video of the same length as the music to be presented. A possible contribution, therefore, could be to adjust this player to allow for audio only presentation, and use the visualization tools described by [PPQ16] to depict the sound-field in real-time.

[86] We found the same problem with other WebXR tools, such as Omnitone.
[87] Thankfully the authors do have some information regarding this in the repository.

3.7 Other Relevant Projects

This section corresponds to WebXR projects that don't fall in either the player or library category but we believe are worthwhile mentioning as they address different issues of spatial audio distribution on the web. The first project, WHAM, describes the development of a head-tracking system using camera data, which can be used to generate HTBA without external hardware. The second project describes PlugSonic, which contains an online editor for soundscape generation. Finally, we discuss an audio walk-through project which allows one to place multiple sound-fields in a virtual environment.

3.7.1 WHAM

Webcam Head-tracked Ambisonics (WHAM) is a recent contribution by Dring and Wiggins, from the University of Derby, which combines facial recognition on the web with binaural ambisonic decoding. As the authors note, and as formerly stated, head-tracking has been shown to improve externalization, source localization, and reduce the number of front-back confusions [DW20]. Thus, the ability to head-track the user as they experience the work substantially improves the quality of the experience.

Common methods used for head-tracking include using an external gyroscope - which send data either via MIDI, Bluetooth, or OSC[88] - or, using an infrared camera that can track the user over 6DoF[89]. According to Dring and Wiggins, head-trackers need to have low-latency and angular resolution of 2° for accurate binaural synthesis. The latency will dictate how quickly the rotation of the sound-field can be applied - too much latency will break the illusion of immersion.

In order to circumvent the need for a gyroscope, the authors used the *Beyond*

[88] Open Sound Control.
[89] Such as TrackIR.

Reality Face (BRF) SDK[90] to track listeners' faces via their webcam feeds. This SDK uses 68 facial landmarks that are displayed on top of the webcam image, allowing the user to see when the tracking is working or not[91]. The downsides of using a webcam is that proper tracking requires good lighting and camera positioning. The system also does not seem to recognize angles correctly when the user is facing away from the camera.

According to the authors: "observations show that the BRF tracking points reach an approximate maximum at 40° to the left or right before failing to track." Facial hair and other objects in front of the face also lead to poor performance. Various head-tracking studies have shown that sub-30ms latency is necessary in order to maintain proper synthesis. Dring and Wiggins noted that the latency varies according the listener's GPU[92], however, with their set-up, they were able to find a webcam-only latency of 128ms at 30fps. Despite the poor latency, the authors did not notice any drop in quality, however, no formal studies were conducted.

Future work proposed by the authors included the development of a listening test which could automatically download BRIRs[93], the main subject of Dring's research, from a remote server. The authors also stated that the system could benefit from a simple SOFA integration, allowing those with personalized filters to upload them to the site. Finally, the authors pointed out the need for an objective comparison of the WHAM method versus other head-trackers, considering the latency difference of the two systems. This could be done, for example, using the open-source MIDI head-tracker from IEM[94] and a JS MIDI library.

In our own tests, we found that the head-tracking was very promising as a method to track the user's orientation. We were able to import a musical work in ambisonics,

[90]Software Development Kit.
[91]One can try out the demo here.
[92]Graphical Processing Unit, in this case responsible for capturing the camera feed.
[93]Binaural Room Impulse Responses, or HRIRs with an additional diffuse field.
[94]Repo found here.

both in FOA and HOA, and listen to the piece while the camera performed its necessary task. The result felt more comfortable that using a cursor to navigate the sound-field. Unfortunately, all the deficits the authors noted were also present in our test. In addition, we found that we needed to sit uncomfortably close to the laptop camera for the tracking to work well. Therefore, a zoom (or cropping) function for the camera feed might be worthwhile implementing. Future research, not elucidated by Dring and Wiggins, could include using these feature points to select the HRTF sets[95], or simply measuring the interaural time difference based on the head geometry.

We believe facial tracking is likely to be an important part of the future of HTBA. In poor lighting conditions, light sensors based on infrared technology might become more important. This current system could easily be re-purposed for delivery of spatial audio on the web in a library format. Currently, however, the site is not set-up in such a way that most users can enjoy music using WHAM. We had the pleasure of enjoying this system only because we had access to the right file-types, which won't be true for most users. Ideally, a *Graphical User Interface* (GUI) like HOAST's [DMKH+20] might be developed. Furthermore, a method for anybody to upload sounds permanently should be added[96].

3.7.2 PlugSonic

For people with limited sound design experience, PlugSonic aims at providing a way to create spatial audio on the browser. PlugSonic [CGLP19] is a *cultural heritage* project focused on accessible spatial audio tools on the web. Cultural heritage is a popular topic in spatial audio research today; one example of such a project is the Hagia Sophia auralization by Abel et al. [AWK+13], in which the acoustics of the sacred site are analyzed and recreated.

[95] Using the anthropometric data.
[96] Provided a large enough server can be secured.

> "Cultural heritage often brings to mind artifacts (paintings, drawings, prints, mosaics, sculptures), historical monuments and buildings, as well as archaeological sites. But the concept of cultural heritage is even wider than that, and has gradually grown to include all evidence of human creativity and expression: photographs, documents, books and manuscripts, and instruments, etc. either as individual objects or as collections." [Wha]

Many spatial audio cultural heritage projects aim at recreating the acoustics of current or lost structures. PlugSonic, in contrast, is about recreating the sonic feeling of a particular site, by providing listeners with soundscapes of important places around the world, without an emphasis on architectural acoustics.

The PlugSonic system is composed of multiple web and phone apps that work in conjunction, allowing the creator and/or spectator to easily make and digest spatial soundscapes. The PlugSonic ecosystem is separated into two main sections called: PlugSonic Sample, and, PlugSonic Soundscape. The PlugSonic Sample software is an audio editor meant to allow editing of sound-files, for later importing into PlugSonic Soundscape. It allows one to: trim, listen, apply fades, and change the volume of sound samples - without needed to leave the browser[97]. The PlugSonic Soundscape system is split into the Create and the Experience apps.

The Create app is used to generate sound-fields, which are later accessed via the Experience app during museum visits or other cultural settings. With the Create app, users can place point-sources in space, and move around to hear them from different perspectives. They can also control the room size, and the circumference of each audio source's radiation pattern. Once the audio experience has been properly edited, the user exports either the soundscape meta-data on its own, as a JSON file, or the JSON file along with associated assets. The JS Object Notation (JSON) file contains the metadata required for the Experience app to reproduce the sound-field. These JSON files can also be re-imported into the Create app for further editing.

[97] These are regular mono files.

Comunita et al., the developers of this project, employed two types of tests to determine the effectiveness of their system. The first was an objective evaluation which observed the load time and CPU/RAM usage of the app using various numbers of sources, and using several listening conditions. Namely, the authors tested a: stationary listener, a rotating listener, a listener moving in circles, and, a listener moving in circles while using the performance mode. The performance mode from the internal 3D Tune-In (3DTI) Toolkit [CRPGT+19] is an IIR-based binaural simulation which reduces computational load at the expense of quality. A similar concept is explored in the BiLi project [Car15]. While the 3DTI Toolkit, used to create PlugSonic, was written in C++, the authors were able to compile a JS version using the emscripten toolchain.

One can test their own web-apps' CPU/RAM performance in Chromium-based browsers by opening the "Task Manager" and check the load time by opening the "Inspect" panel, and selecting the Performance tab. We believe the authors employed the same process, although this was not explicitly stated. Figure 3.3 shows a summary of the load time analysis generated by a Chromium-based browser, showing the different stages involved in the loading[98]. Comunita et al. [CGLP19] used a Lenovo computer with a 2.8GHz chip and 16GB of RAM, and found the maximum number of sources they could render was 35.

The second type of evaluation the author employed was a subjective evaluation, in which a small number of participants used all these tools and then provided feedback to the authors. The subjective evaluation concluded that the apps were easy to use and responsive, however, various improvements were suggested by subjects which the authors intend to incorporate. Some of these regarded GUI improvements, flexibility and compatibility with mobile devices.

While the PlugSonic ecosystem seems promising as an easy-to-use tool for creation of spatial audio content, the format also forces the user to use the Experience

[98]The site that was analyzed was arbitrary, this is just demonstrative.

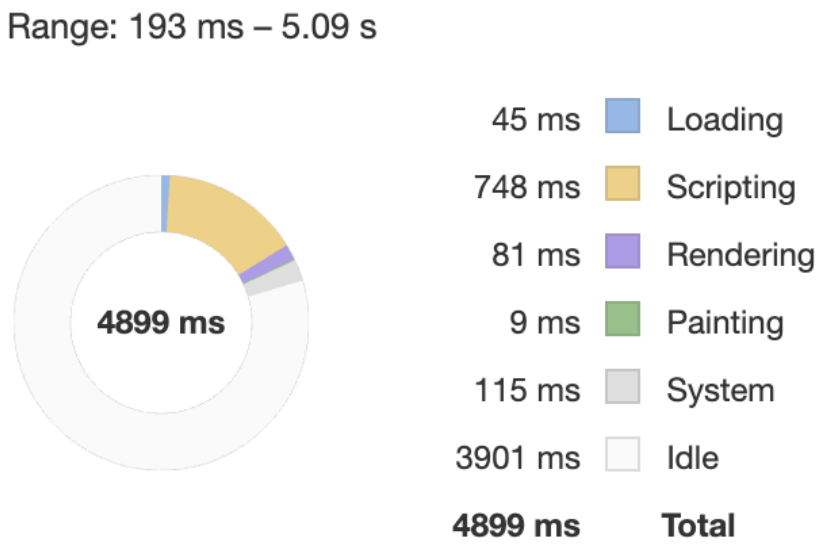


Figure 3.3: Inspect Performance: Load Time Example - Chromium Browser

app in order to present the sound-fields they create. In contrast to other systems, the PlugSonic Create software has limited processing capabilities and seems to be meant for straightforward narrative content rather than musical material. Since the Experience app makes use of the iOS ARKit, it is also limited by the OS.

The two most noteworthy aspects of PlugSonic are the sound-field composition on the browser, and the fact that hearing aid data can be incorporated into the binaural rendering. While DAW-based tools might be better for musicians, for the general public, this system provides a way to easily create and audition soundscapes using very simple controls. The application requires no installation, or background knowledge of spatial audio research - making it more accessible than many other tools. Unfortunately, as a flexible solution for distribution of this content, a different Create app would be needed that could be used on other OSs. Finally, PlugSonic stands out as one of the only projects we looked at which addresses the effects of hearing aids on the localization of sound sources; this is made possible using the 3DTI

toolkit [CRPGT+19] which has filters designed by analyzing the effects of hearing aids on HRTFs.

3.7.3 Virtual Audio Walk-through

Deppisch et al. [DS17] describe a web project in which listeners are given the ability to move inside a scene containing multiple ambisonic sound-fields and independent sound objects. The project is built using JSAmbisonics [PPQ16], and allows the user to define a space using JSON to determine the positions of various FOA arrays and spot microphones[99]. "During playback, every microphone capsule is interpreted as a virtual speaker object which then gets placed in the room according to its original position."

The "scene file", written in JSON, defines the: position, gain, orientation, distance gain function, directivity, and sound file name - associated with each mono or FOA signal. For mono objects, one can also activate *Near-Field Compensation* (NFC) filters, which simulate frequency variations for ambisonics signals proximal and distal to the listener. Namely, as source approach the listener, the low-frequency is boosted, and vice-versa.

NFC (HOA) was developed by Daniel [DM04], whom described method as a series of frequency-dependent distance-controlled filters based on the behavior of spherical Hankel functions. Every ambisonic order receives a separate filter; unfortunately, the compensation is unrealizable directly using Hankel functions, as they result in excessive low-end frequency boosting. Derivation of spherical Hankel functions are shown in Appendix A.

Vennerod [Ven14] defines the frequency-dependent NFC function without correction as:

[99] Mono microphones used to record direct sources.

$$F_n(\rho,\omega) = \sum_{m=0}^{n} \frac{(n+m)!}{(n-m)!m!} \left(\frac{-\mathrm{i}c}{2\omega\rho}\right)^m \tag{3.22}$$

where,

ρ is the distance of the source from the origin,

ω is the angular frequency,

i is the imaginary unit, and,

c is the speed of sound.

The filter correction entails creating a second filter of identical nature replacing the source distance with speaker distance R, used for reproduction. In binaural systems we can use the distance at which the HRTFs were measured[100]. The resulting filter $H_n(\omega)$ is found by dividing both filters. This filter should be applied for sources where ρ is smaller than R (e.g. in the Near-Field). This results in a boost of low frequencies when sources are proximal, which aims at simulating diffraction around the head.

$$H_n(\omega) = \frac{F_n(\rho,\omega)}{F_n(R,\omega)} \tag{3.23}$$

In this Virtual Walk-through, objects of type *fourChannelArray*, contain four separate mono channels that are mapped to four virtual speaker objects inside the scene. An additional directivity parameter allows these sources to vary from omni-directional to figure-8. The directivity function is defined as:

$$g_{dir} = \gamma + (1-\gamma)\cos(\varphi) \tag{3.24}$$

[100]For real speaker arrays we don't know the speaker distance a priori, which requires additional considerations.

where γ is the directivity factor between omni ($\gamma = 1$), and figure-8 ($\gamma = 0$), and φ is the angle between: the vector pointing from the listener to the virtual speaker, and the vector corresponding to the direction of the virtual speaker.

The authors also provide a distance gain function:

$$g_{dist} = \begin{cases} g_0 + (1 - g_0)\, r & , \text{ if } r \leq 1 \\ \frac{1}{r^\alpha} & , \text{ if } r > 1 \end{cases} \tag{3.25}$$

Where α and g_0 are the adjustment parameters. The distance gain "equals 1 for distance $r = 1$, linearly interpolates to g_0 for distances $r < 1$ and decreases by $1/r^\alpha$ for distances $r > 1$" [DS17].

Deppisch and Sontacchi [DS17], the authors of this paper, also implemented a reverb design in the walk-through based on the *Image Source Method* (ISM). Room reflections are simulated using delay lines whose delay time is calculated based on the distance of sources to boundaries (e.g. walls) defined by the user. First and second order image sources are provided in the application. Unfortunately, while the ISM does provide realistic reverb, it has also been shown to be computationally taxing. In order to generate the directivity of the reflections, virtual speakers are placed at the boundaries, and the delay times are updated according to:

$$\Delta t = \frac{r}{343 \frac{m}{s}} \tag{3.26}$$

where Δt is the delay time. This is calculated as a function of the speed of sound and the distance r of the listener to the speaker. Image sources are also low-passed in order to simulate high-frequency loss caused by absorption during reflection. Since delay times are continually updated, the ISM method also yields a *Doppler-shift* for reflected sources. Finally, the *fourChannelArray* files are normalized in order to maintain an even distribution of sound.

All these virtual objects are then decoded using a regular distribution of virtual speakers (e.g. a t-design). SOFA formatted HRTFs are supported. The user is

provided a user interface allowing them to walk around the room while rotating their head, simulating a 6DoF experience. The authors note the possibility of using a head-tracker to further improve the experience, which could be supported via MIDI with the Web MIDI API or WebMidi.js[101].

While this paper provided an interesting proof-of-concept for a virtual walk-through based on ambisonic spatial audio, we believe it could use a number of improvements.

1. All virtual objects shows in the UI (mono objects) use the same icon, it would be good to have different icons representing the response of the microphone, and a parameter to denote their orientation.

2. The system only counts with a single perspective (birds-eye), a second vantage point could be provided to allow users to show the height of each capture point.

3. In this additional vantage point, one could also depict the elevation angle of the head, which is currently not visible.

4. Finally, the documentation leaves us with many questions regarding *fourChannelArray*, more details could be provided in future publications about the implementation of this sound-field interpolator.

While this work is quite rudimentary for what it could be, it is one of the only projects that attempts to generate 6DoF audio via ambisonics on the web. For more sophisticated recording set-ups, this could allow one to combine multiple HOA recordings, and represent them all at once within a single scene. A system such as this one could benefit from a different interface, perhaps based on modern WebXR frameworks. Unfortunately, while the idea of 6DoF ambisonic is exciting, we should recall the existing problems associated with large file sizes in streaming applications.

[101] Chrome and Opera support the Web MIDI API.

Deppisch et al. do not mention any adaptive streaming in this publication, but, since HOAST was also authored by Deppisch, we expect this is something we might see from the author in the future.

3.8 Musical Works

The HOAST library features a number of great musical works with video that can be experienced either on desktop or mobile devices. Many of these have real world footage that accompany them while some have visualizations instead. The authors did not provide any insight into how these visualizations were generated, however, we can see from the player that they are videos that were pre-rendered, not real-time animations.

The "Experiments with Google" site has a few interesting artworks. *Inside the Music* is a visualization and spatial audio experience that lets you listen to stems from various musicians work. Unfortunately, we could not find the demos for the famous musicians like Phoenix anywhere. We did manage to find this demo and the code. This is another method one could use to present ambisonic music without a 360° video. A potential problem for composers wishing to use this method could be the splitting of stems into 30 second chunks, which Google recommends to perform adaptive streaming. This requires familiarity with the command line, and Python.

The other project we wish to highlight from Google is *The Musical Forrest*, which is a playground for making music by hitting simple geometries with a virtual representation of a HMD controller. In the past, the present author was involved in a similar project called Synth-esthesia which was built using Unity. This project was abandoned due to the pandemic, and unfortunately has not been academically documented yet. While these projects are both interesting conceptually, neither is very accessible, since they both rely on stand-alone HMDs, which cost many hundreds of dollars.

The final piece we would like to highlight is one of our own, which we worked on with two undergraduates students over the course of 2020-2021. The project is titled after the song produced by Tim Gmeiner: Pigments of Imagination[102]. The piece was created using some of the IEM plug-ins discussed in Chapter 1, as well as A-Frame. We recently submitted the formal documentation and piece, as a music submission, to the Audio Mostly conference. The piece's extensive documentation can be found at this link. The piece works on Chrome and Firefox and makes use of Google's *Omnitone* JS library for HOA. In order to accomodate cheaper devices and slower connections, we provided a HOA version, a FOA version, and a FPBA[103] version.

3.9 Future Work

In this chapter we have examined a number of relevant libraries and players that can be used to disseminate spatial music on the web. We also discussed a few additional projects that support head-tracking using webcam data, authoring sound-fields on the web, and a 6DoF prototype based on ambisonic sound theory. Throughout the overview of these projects, we highlighted possible ways of contributing to the development of these tools. At this stage, these open-source libraries feature enough functions for high-quality dissemination of spatial music. However, certain permutations of decoders with particular frameworks have not yet been implemented or evaluated.

It could also be useful to combine one, or more, of these projects, into an integrated system that provides more features and control for the listener. For example, the HOAST system could benefit from an automated visualization engine, similar to that described in the JSAmbisonic publication. In this way, musicians who don't

[102]A second undergraduate student, Eito Murakami, provided feedback and support throughout the process.

[103]Fixed-perspective Binaural Audio.

own 360° cameras, or who don't have experience with 360°video in general, could still share their music. The visualization is not only a way to maintain the audience engaged, but also facilitates localization. The WHAM project could also be included in this player, so that people who wish to navigate the sound-field using their web-cam could chose to do so, in lieu of relying on their mouse or keyboard. While, this step is irrelevant for mobile or HMD-based viewing, it could enhance the desktop experience substantially.

Finally, as we mentioned before, it would be also be useful to develop a back-end that would allow people without programming experience to create their own channel, and upload their ambisonic music. Certain elements from Resonance [GAK+19] could also be incorporated into the player, such as the reverberation or certain optimization, which improve performance on low-end devices. Finally, it would be proper to have the capacity to upload SOFA formatted HRTFs to the site, so users can enjoy personalized HTBA using their own measurements, or, alternatively, have access to a large data-base such as the one proposed by the OpenDAP SOFA project.

3.10 Conclusion

This chapter has examined how contemporary web-based XR systems can be exploited to distribute spatial music en masse. Emphasis has been given to the ways in which XR can be used as a dissemination tool, allowing greater equity of access to this type of music. As a way to contrast the access issues surrounding XR, we also discussed some XR tools that are not low-cost or open-source - despite these two factors being of primary interest.

A large motivation for the creation and dissemination of spatial music stems from the growth of the XR industry in the last decade. Any complete discussion involving spatial music, therefore, should address how spatial music is being shaped by the development of these new tools. Over the years a number of Universities

and companies have developed increasingly sophisticated XR systems; unfortunately, many of these XR system still remain too costly for most people. We believe WebXR, in particular, is a powerful distribution tool which should be adopted by more artists working in the spatial audio domain. Online publishing is an important part of any artists work - allowing patrons to preview artists' works before committing to a live experience. WebXR provides musicians working with spatial music the opportunity to reach a wider audience, while preserving some of the depth that was carefully crafted into their music.

Appendix A

Bessel Functions

A.1 Cylindrical Bessel Functions of the First Kind

Bessel functions arise when finding solutions to Laplace's equation and the Helmholtz equation in cylindrical or spherical coordinates and are important in problems of wave propagation.

When n is an integer, these functions can be calculated by [oM]:

$$J_n(z) = \left(\frac{z}{2}\right)^n \sum_{k=0}^{\infty} \frac{(-1)^k}{k!(n+k)!} \left(\frac{z}{2}\right)^{2k} \tag{A.1}$$

where,

n is the ambisonic order, and

z is a vector of frequency values.

However, spherical Bessel functions are easier to define as a function of half-integer order cylindrical Bessel functions. Luckily, the first two half-integer order cylindrical Bessel functions are elementary. They are given by:

$$J_{\frac{1}{2}}(z) = \sqrt{\frac{2}{\pi z}} \sin(z) \tag{A.2}$$

and,

$$J_{-\frac{1}{2}}(z) = \sqrt{\frac{2}{\pi x}}\cos(z) \tag{A.3}$$

using the recurrence formula:

$$J_{\nu-1}(z) + J_{\nu+1}(z) = \frac{2\nu}{z}J_{\nu}(z) \tag{A.4}$$

we can calculate other half-integer order cylindrical Bessel functions. For example, for $v = 1/2$:

$$\sqrt{\frac{2}{\pi \cdot z}}\cos(z) + J_{v+1}(z) = \frac{2(1/2)}{z}\left(\sqrt{\frac{2}{\pi \cdot z}}\sin(z)\right) \tag{A.5}$$

Isolating $J_{v+1}(z)$ gives:

$$J_{3/2}(z) = \frac{1}{z}\left(\sqrt{\frac{2}{\pi z}}\sin(z)\right) - \sqrt{\frac{2}{\pi z}}\cos(z). \tag{A.6}$$

Which further simplifies to:

$$J_{3/2}(z) = \frac{1}{z}\cdot\sqrt{\frac{2}{\pi}}\cdot\sqrt{\frac{1}{z}}\cdot\sin(z) - \sqrt{\frac{1}{z}}\cdot\sqrt{\frac{2}{\pi}}\cdot\cos(z). \tag{A.7}$$

Finally we obtain:

$$J_{3/2}(z) = \sqrt{\frac{2}{\pi}}\cdot\left(\frac{1}{z^{3/2}}\sin(z) - \frac{1}{z^{1/2}}\cos(z)\right). \tag{A.8}$$

Additional properties of Bessel functions can be used to find negative half-integer order solutions. Various authors use built-in functions to calculate Cylindrical Bessel functions. They can be found in MATLAB, for example.

A.2 Cylindrical Bessel Functions of the Second Kind

Provided the Cylindrical Bessel Functions of the First Kind have already been calculated, the Cylindrical Bessel Functions of the Second Kind can be calculated using:

$$Y_\nu(z) = \frac{J_\nu(z)\cos\nu\pi - J_{-\nu}(z)}{\sin\nu\pi} \quad \text{for } \nu \notin \mathbb{Z} \tag{A.9}$$

This equation is valid for values of v that are not part of the set $\mathbb{Z}$, which corresponds to real integers. We can use these two definitions to derive our Spherical Bessel Functions of the First and Second kind, as well as the Spherical Hankel functions of the First and Second Kind.

A.3 Spherical Bessel Functions

To calculate our Sphericak Bessel Functions we use

$$\begin{aligned} j_n(z) &= \sqrt{\frac{\pi}{2z}} J_{n+1/2}(z) \\ y_n(z) &= \sqrt{\frac{\pi}{2z}} Y_{n+1/2}(z) \\ h_n^{(1)}(z) &= j_n(z) + iy_n(z) = \sqrt{\frac{\pi}{2z}} \left[J_{n+1/2}(z) + iY_{n+1/2}(z)\right] \\ h_n^{(2)}(z) &= j_n(z) - iy_n(z) = \sqrt{\frac{\pi}{2z}} \left[J_{n+1/2}(z) - iY_{n+1/2}(z)\right] \end{aligned} \tag{A.10}$$

$j_n(z)$ and $y_n(z)$ represent the spherical Bessel functions of the first and second kind respectively. $h_n^{(1)}(z)$ and $h_n^{(2)}(z)$ define the Hankel functions of the first and second kind respectively.

Alternate solutions to the four spherical Bessel functions can also be found in the Appendix of [ZF19]. Implementations of these can also be found in Politis's "Array-Response-Simulator"[1] or the SOFiA toolbox used in Moschner et al.[MDLP20]. In our research we Bessel functions appeared when discussing Near-Field Compensation (NFC), Radial Filter Design for HOA encoding, and a particular VST application (e.g. Kronlachner's widening algorithm).

A.4 Practical Spherical Bessel Functions

Generally in periphonic ambisonic research we are interested in spherical Bessel functions of the first and second kind, which we can use to calculate Hankel functions applicable to NFC and Radial filter design. Here we give the spherical Bessel functions up to 3OA, using simple trigonometric functions, which we find suitable for most recording and reproduction purposes:

$$\begin{aligned} j_0(x) &= \tfrac{\sin x}{x} \\ j_1(x) &= \tfrac{\sin x}{x^2} - \tfrac{\cos x}{x} \\ j_2(x) &= \left(\tfrac{3}{x^2} - 1\right) \tfrac{\sin x}{x} - \tfrac{3\cos x}{x^2}, \\ j_3(x) &= \left(\tfrac{15}{x^3} - \tfrac{6}{x}\right) \tfrac{\sin x}{x} - \left(\tfrac{15}{x^2} - 1\right) \tfrac{\cos x}{x} \end{aligned} \tag{A.11}$$

The relationships above are for Spherical Bessel Functions of the First Kind.

$$\begin{aligned} y_0(x) &= -j_{-1}(x) = -\tfrac{\cos x}{x} \\ y_1(x) &= j_{-2}(x) = -\tfrac{\cos x}{x^2} - \tfrac{\sin x}{x}, \\ y_2(x) &= -j_{-3}(x) = \left(-\tfrac{3}{x^2} + 1\right) \tfrac{\cos x}{x} - \tfrac{3\sin x}{x^2}, \\ y_3(x) &= j_{-4}(x) = \left(-\tfrac{15}{x^3} + \tfrac{6}{x}\right) \tfrac{\cos x}{x} - \left(\tfrac{15}{x^2} - 1\right) \tfrac{\sin x}{x} \end{aligned} \tag{A.12}$$

While the second equation refers to Spherical Bessel Functions of the Second Kind.

[1] Found here (Accessed: May 14, 2021).

Bibliography

[AB79] Jont B Allen and David A Berkley. Image method for efficiently simulating small-room acoustics. *The Journal of the Acoustical Society of America*, 65(4):943–950, 1979.

[APSV19] Benoit Alary, Archontis Politis, Sebastian Schlecht, and Vesa Välimäki. Directional feedback delay network. *Journal of the Audio Engineering Society*, 67(10):752–762, 2019.

[Arc] Politis Archontis. Spherical harmonic transform library. http://research.spa.aalto.fi/projects/sht-lib/sht.html. (Accessed on 04/23/2021).

[AWK+13] Jonathan S Abel, Wieslaw Woszczyk, Doyuen Ko, Scott Levine, Jonathan Hong, Travis Skare, Michael J Wilson, Sean Coffin, and Fernando Lopez-Lezcano. Recreation of the acoustics of hagia sophia in stanford's bing concert hall for the concert performance and recording of cappella romana. In *International Symposium on Room Acoustics*, pages 1–13, 2013.

[Bac06a] Juha Backman. Gradient microphone design using miniature microphone arrays. In *Audio Engineering Society Convention 121*. Audio Engineering Society, 2006.

[Bac06b] Juha Backman. Miniature microphone arrays for multi-channel recording. In *Audio Engineering Society Convention 120*. Audio Engineering Society, 2006.

[Ben12] Eric M Benjamin. A second-order soundfield microphone with improved polar pattern shape. In *Audio Engineering Society Convention 133*. Audio Engineering Society, 2012.

[BFB97] Miguel A Blanco, Manuel Flórez, and Margarita Bermejo. Evaluation of the rotation matrices in the basis of real spherical harmonics. *Journal of Molecular Structure: THEOCHEM*, 419(1-3):19–27, 1997.

[BGPA14] Benjamin Bernschütz, A Vázquez Giner, Christoph Pörschmann, and Johannes Arend. Binaural reproduction of plane waves with reduced modal order. *Acta Acustica united with Acustica*, 100(5):972–983, 2014.

[Bli] Blink (rendering engine) - the chromium projects. https://www.chromium.org/blink. (Accessed on 04/26/2021).

[BPSW11] Benjamin Bernschütz, Christoph Pörschmann, Sascha Spors, and Stefan Weinzierl. Sofia sound field analysis toolbox. In *Proceedings of the International Conference on Spatial Audio (ICSA)*, pages 7–15, 2011.

[Car15] Thibaut Carpentier. Binaural synthesis with the web audio api. In *1st Web Audio Conference (WAC)*, 2015.

[CAZ15] Hanchi Chen, Thushara D Abhayapala, and Wen Zhang. Theory and design of compact hybrid microphone arrays on two-dimensional planes for three-dimensional soundfield analysis. *The Journal of the Acoustical Society of America*, 138(5):3081–3092, 2015.

[CDCJ18] Elliot K Canfield-Dafilou, Eoin Callery, and Christopher Jette. A portable impulse response measurement system. In *15th Sound and Music Computing Conference*, pages 172–176, 2018.

[CGLP19] Marco Comunità, Andrea Gerino, Veranika Lim, and Lorenzo Picinali. Web-based binaural audio and sonic narratives for cultural heritage. In *Audio Engineering Society Conference: 2019 AES International Conference on Immersive and Interactive Audio*. Audio Engineering Society, 2019.

[Con] Conversion to minimum phase. https://ccrma.stanford.edu/~jos/filters/Conversion_Minimum_Phase.html. (Accessed on 05/06/2021).

[CRPGT+19] María Cuevas-Rodríguez, Lorenzo Picinali, Daniel González-Toledo, Carlos Garre, Ernesto de la Rubia-Cuestas, Luis Molina-Tanco, and Arcadio Reyes-Lecuona. 3d tune-in toolkit: An open-source library for real-time binaural spatialisation. *PloS one*, 14(3):e0211899, 2019.

[DLAP19] Damian T Dziwis, Tim Lübeck, Johannes M Arend, and Christoph Pörschmann. Development of a 7th order spherical microphone array for spatial audio recording. *45. Deutsche Jahrestagung für Akustik*, 2019.

[DM04] Jérôme Daniel and Sébastien Moreau. Further study of sound field coding with higher order ambisonics. In *Audio Engineering Society Convention 116*. Audio Engineering Society, 2004.

[DMKH+20] Thomas Deppisch, Nils Meyer-Kahlen, Benjamin Hofer, Tomasz Latka, and Tomasz Zernicki. Hoast: A higher-order ambisonics streaming platform. In *Audio Engineering Society Convention 148*. Audio Engineering Society, 2020.

[DRS15] Matthew Dabin, Christian Ritz, and Muawiyath Shujau. Design and analysis of miniature and three tiered b-format microphones manufactured using 3d printing. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2674–2678. IEEE, 2015.

[DS17] Thomas Deppisch and Alois Sontacchi. Browser application for virtual audio walkthrough. In *Forum Media Technology*, pages 145–150, 2017.

[DW20] Mark Dring and Bruce Wiggins. Wham: Webcam head-tracked ambisonics. 2020.

[EJ16] Nicolas Epain and Craig T Jin. Spherical harmonic signal covariance and sound field diffuseness. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(10):1796–1807, 2016.

[Ele91] Richard Elen. Whatever happened to ambisonics. *AudioMedia Magazine, Nov*, 1991.

[FM99] Jörg Fliege and Ulrike Maier. The distribution of points on the sphere and corresponding cubature formulae. *IMA Journal of Numerical Analysis*, 19(2):317–334, 1999.

[GAK+19] Marcin Gorzel, Andrew Allen, Ian Kelly, Julius Kammerl, Alper Gungormusler, Hengchin Yeh, and Francis Boland. Efficient encoding and decoding of binaural sound with resonance audio. In *Audio Engineering Society Conference: 2019 AES International Conference on Immersive and Interactive Audio*. Audio Engineering Society, 2019.

[Ger76] Michael A Gerzon. Unitary (energy-preserving) multichannel networks with feedback. *Electronics Letters*, 12(11):278–279, 1976.

[GPL18] Raimundo González, Joshua Pearce, and Tapio Lokki. Modular design for spherical microphone arrays. In *Audio Engineering Society Conference: 2018 AES International Conference on Audio for Virtual and Augmented Reality*. Audio Engineering Society, 2018.

[GS20] Corentin Guezenoc and Renaud Seguier. Hrtf individualization: A survey. *arXiv preprint arXiv:2003.06183*, 2020.

[HA17] Christoph Hohnerlein and Jens Ahrens. Spherical microphone array processing in python with the sound field analysis-py toolbox. *Fortschritte der Akustik–DAGA 2017*, pages 1033–1036, 2017.

[Har] Sloane Hardin. Spherical designs. http://neilsloane.com/sphdesigns/. (Accessed on 05/24/2021).

[HHKP15] Jürgen Herre, Johannes Hilpert, Achim Kuntz, and Jan Plogsties. Mpeg-h audio—the new standard for universal spatial/3d audio coding. *Journal of the Audio Engineering Society*, 62(12):821–830, 2015.

[HLB08] Aaron Heller, Richard Lee, and Eric Benjamin. Is my decoder ambisonic? In *Audio Engineering Society Convention 125*. Audio Engineering Society, 2008.

[HN13] Keyong Han and Arye Nehorai. Improved source number detection and direction estimation with nested arrays and ulas using jackknifing. *IEEE Transactions on Signal Processing*, 61(23):6118–6128, 2013.

[HS96] Ronald H Hardin and Neil JA Sloane. Mclaren's improved snub cube and other new spherical designs in three dimensions. *Discrete & Computational Geometry*, 15(4):429–441, 1996.

[IEMa] IEM. Iem plug-in suite. https://plugins.iem.at/. (Accessed on 05/27/2021).

[IEMb] IEM. Plug-in descriptions. https://plugins.iem.at/docs/plugindescriptions/#coordinateconverter. (Accessed on 04/21/2021).

[Ini14] Inigo.quilez. Spherical harmonics - wikipedia. https://en.wikipedia.org/wiki/Spherical_harmonics, 14 May 2014. (Accessed on 02/19/2021).

[IR96] Joseph Ivanic and Klaus Ruedenberg. Rotation matrices for real spherical harmonics. direct determination by recursion. *The Journal of Physical Chemistry*, 100(15):6342–6347, 1996.

[JEP13] Craig T Jin, Nicolas Epain, and Abhaya Parthy. Design, optimization and evaluation of a dual-radius spherical microphone array. *IEEE/ACM transactions on audio, speech, and language processing*, 22(1):193–204, 2013.

[KD15] Gavin Kearney and Tony Doyle. An hrtf database for virtual loudspeaker rendering. In *Audio Engineering Society Convention 139*. Audio Engineering Society, 2015.

[KJM08] Bill Kapralos, MR Jenkin, and E Milios. Virtual audio systems. *Presence: Teleoperators and Virtual Environments*, 17(6):527–549, 2008.

[Kro14a] Matthias Kronlachner. Plug-in suite for mastering the production and playback in surround sound and ambisonics. *Gold-Awarded Contribution to AES Student Design Competition*, 2014.

[Kro14b] Matthias Kronlachner. Spatial transformations for the alteration of ambisonic recordings. *M. Thesis, University of Music and Performing Arts, Graz, Institute of Electronic Music and Acoustics*, 7, 2014.

[KSS16] Shoken Kaneko, Tsukasa Suenaga, and Satoshi Sekine. Deepearnet: individualizing spatial audio with photography, ear shape modeling, and neural networks. In *Audio Engineering Society Conference: 2016 AES International Conference on Audio for Virtual and Augmented Reality*. Audio Engineering Society, 2016.

[KZ14] Matthias Kronlachner and Franz Zotter. Spatial transformations for the enhancement of ambisonic recordings. In *Proceedings of the 2nd International Conference on Spatial Audio, Erlangen*, 2014.

[LA14] Trond Lossius and Joseph Anderson. Atk reaper: The ambisonic toolkit as jsfx plugins. In *ICMC*, 2014.

[LaV06] Steven M. LaValle. Yaw, pitch, and roll rotations. http://planning.cs.uiuc.edu/node102.html, 2006. (Accessed on 04/24/2021).

[LIS] Listen hrtf database. http://recherche.ircam.fr/equipes/salles/listen/. (Accessed on 04/26/2021).

[LL16] Fernando Lopez-Lezcano. The* sphear project, a family of parametric 3d printed soundfield microphone arrays. In *Audio Engineering Society Conference on Soundfield Control*, 2016.

[LL18] Fernando Lopez-Lezcano. The* sphear project update: the tinysphear and octathingy soundfield microphones. In *Audio Engineering Society Conference: 2018 AES International Conference on Audio for Virtual and Augmented Reality*. Audio Engineering Society, 2018.

[LL19] Fernando Lopez-Lezcano. The sphear project update: Refining the octasphear, a 2nd order ambisonics microphone. 2019.

[Lon14] Marshall Long. 2 - fundamentals of acoustics. In Marshall Long, editor, *Architectural Acoustics (Second Edition)*, pages 39 – 79. Academic Press, Boston, second edition edition, 2014.

[LS17] Hugh Lynch and Robert Sazdov. A perceptual investigation into spatialization techniques used in multichannel electroacoustic music for envelopment and engulfment. *Computer Music Journal*, 41(1):13–33, 2017.

[LVJ+14] Mikko-Ville Laitinen, Juha Vilkamo, Kai Jussila, Archontis Politis, and Ville Pulkki. Gain normalization in amplitude panning as a function of frequency and room reverberance. In *Audio Engineering Society Conference: 55th International Conference: Spatial Audio*. Audio Engineering Society, 2014.

[LZ15] Stefan Lösler and Franz Zotter. Comprehensive radial filter design for practical higher-order ambisonic recording. *Fortschritte der Akustik, DAGA*, pages 452–455, 2015.

[McCa] Leo McCormack. research.spa.aalto.fi/projects/compass_vsts/plugins.html. http://research.spa.aalto.fi/projects/compass_vsts/plugins.html. (Accessed on 05/12/2021).

[McCb] Leo McCormack. research.spa.aalto.fi/projects/sparta_vsts/plugins.html. http://research.spa.aalto.fi/projects/sparta_vsts/plugins.html. (Accessed on 05/10/2021).

[MDB06] Sébastien Moreau, Jérôme Daniel, and Stéphanie Bertet. 3d sound field recording with higher order ambisonics–objective measurements and validation of a 4th order spherical microphone. In *120th Convention of the AES*, pages 20–23, 2006.

[MDLP20] Oscar Moschner, Damian Dziwis, Tim Lübeck, and Christoph Pörschmann. Development of an open source customizable high order rigid sphere microphone array. In *Audio Engineering Society Convention 148*. Audio Engineering Society, 2020.

[MDM19] Leo Mccormack and Symeon Delikaris-Manias. Parametric first-order ambisonic decoding for headphones utilising the cross-pattern coherence algorithm. In *EAA Spatial Audio Signal Processing Symposium*, pages 173–178, 2019.

[MDMF+18] Leo McCormack, Symeon Delikaris-Manias, Angelo Farina, Daniel Pinardi, and Ville Pulkki. Real-time conversion of sensor array signals into spherical harmonic signals with applications to spatially localized sub-band sound-field analysis. In *Audio Engineering Society Convention 144*. Audio Engineering Society, 2018.

[MDMP17] Leo McCormack, Symeon Delikaris-Manias, and Ville Pulkki. Parametric acoustic camera for real-time sound capture, analysis and tracking. In *Proceedings of the 20th International Conference on Digital Audio Effects (DAFx-17)*, pages 412–419, 2017.

[MDMP+19] Leo McCormack, Symeon Delikaris-Manias, Archontis Politis, Despoina Pavlidi, Angelo Farina, Daniel Pinardi, and Ville Pulkki. Applications of spatially localized active-intensity vectors for sound-field visualization. *Journal of the Audio Engineering Society*, 67(11):840–854, 2019.

[ME08] Jens Meyer and Gary Elko. Spherical harmonic modal beamforming for an augmented circular microphone array. In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5280–5283. IEEE, 2008.

[MIC+13] Piotr Majdak, Yukio Iwaya, Thibaut Carpentier, Rozenn Nicol, Matthieu Parmentier, Agnieszka Roginska, Yôiti Suzuki, Kankji Watanabe, Hagen Wierstorf, Harald Ziegelwanger, et al. Spatially oriented format for acoustics: A data exchange format representing head-related transfer functions. In *Audio Engineering Society Convention 134*. Audio Engineering Society, 2013.

[MP09] Christos Manolas and Sandra Pauletto. Enlarging the diegetic space: Uses of the multi-channel soundtrack in cinematic narrative. *The Soundtrack*, 2(1):39–55, 2009.

[MP19] Leo McCormack and Archontis Politis. Sparta & compass: Real-time implementations of linear and parametric spatial audio reproduction and processing methods. In *Audio Engineering Society Conference: 2019 AES International Conference on Immersive and Interactive Audio*. Audio Engineering Society, 2019.

[MPP19] Leo McCormack, Archontis Politis, and Ville Pulkki. Sharpening of angular spectra based on a directional re-assignment approach for ambisonic sound-field visualisation. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 576–580. IEEE, 2019.

[MSK09] Edwin Mabande, Adrian Schad, and Walter Kellermann. Design of robust superdirective beamformers as a convex optimization problem. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 77–80. IEEE, 2009.

[MV+17] Leo McCormack, Vesa Välimäki, et al. Fft-based dynamic range compression. In *Proceedings of the 14th Sound and Music Computing Conference, July*, pages 5–8, 2017.

[MW19] Charlie Middlicott and Bruce Wiggins. Calibration approaches for higher order ambisonic microphones. Audio Engineering Society, 2019.

[Net] Nettings. File:naive ambisonic square decoder example.png - wikimedia commons. https://commons.wikimedia.org/wiki/File:Naive_Ambisonic_Square_Decoder_Example.png. (Accessed on 02/18/2021).

[NOA+17] Miroslaw Narbutt, Seán O'Leary, Andrew Allen, Jan Skoglund, and Andrew Hines. Streaming vr for immersion: Quality aspects of compressed spatial audio. In *2017 23rd International Conference on Virtual System & Multimedia (VSMM)*, pages 1–6. IEEE, 2017.

[NZDS11] Christian Nachbar, Franz Zotter, Etienne Deleflie, and Alois Sontacchi. Ambix-a suggested ambisonics format. In *Ambisonics Symposium, Lexington*, page 11, 2011.

[oM] Encyclopedia of Math. Bessel functions - encyclopedia of mathematics. https://encyclopediaofmath.org/wiki/Bessel_functions. (Accessed on 05/14/2021).

[P+16] Archontis Politis et al. Microphone array processing for parametric spatial audio techniques. 2016.

[PLS12] Nils Peters, Trond Lossius, and Jan C Schacher. Spatdif: Principles, specification, and examples. In *9th Sound and Music Computing Conference, Copenhagen, Denmark*, 2012.

[Pos90] Steven R Postrel. Competing networks and proprietary standards: The case of quadraphonic sound. *The Journal of Industrial Economics*, pages 169–185, 1990.

[PPP12] Archontis Politis, Tapani Pihlajamäki, and Ville Pulkki. Parametric spatial audio effects. *York, UK, September*, 2012.

[PPQ16] Archontis Politis and David Poirier-Quinot. Jsambisonics: A web audio library for interactive spatial sound processing on the web. In *Interactive Audio Systems Symposium*, 2016.

[PTP18] Archontis Politis, Sakari Tervo, and Ville Pulkki. Compass: Coding and multidirectional parameterization of ambisonic sound scenes. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6802–6806. IEEE, 2018.

[Pul97] Ville Pulkki. Virtual sound source positioning using vector base amplitude panning. *Journal of the audio engineering society*, 45(6):456–466, 1997.

[Pul07] Ville Pulkki. Spatial sound reproduction with directional audio coding. *Journal of the Audio Engineering Society*, 55(6):503–516, 2007.

[PVP15] Archontis Politis, Juha Vilkamo, and Ville Pulkki. Sector-based parametric sound field reproduction in the spherical harmonic domain. *IEEE Journal of Selected Topics in Signal Processing*, 9(5):852–866, 2015.

[Rud] Daniel Rudrich. Directivityshaper guide. https://plugins.iem.at/docs/directivityshaper/. (Accessed on 04/21/2021).

[RZF] D Rudrich, F Zotter, and M Frank. Efficient spatial ambisonic effects for live audio (effiziente räumliche ambisonics-effekte zur livebeschallung).

[Sha] SharkD. Spherical coordinate system - spherical coordinate system - wikipedia. https://en.wikipedia.org/wiki/Spherical_coordinate_system#/media/File:Spherical_coordinate_system.svg. (Accessed on 02/19/2021).

[Sin] Singular value decomposition (svd) and pseudoinverse. https://www.johndcook.com/blog/2018/05/05/svd/. (Accessed on 05/06/2021).

[Sio18] Nicole Sioni. A brief history of recorded music - cdrom2go. https://www.cdrom2go.com/blog/a-brief-history-of-recorded-music, September 2018. (Accessed on 02/10/2021).

[Sma97] Denis Smalley. Spectromorphology: explaining sound-shapes. *Organised sound*, 2(2):107–126, 1997.

[SP82] John Stautner and Miller Puckette. Designing multi-channel reverberators. *Computer Music Journal*, 6(1):52–65, 1982.

[SZH18] Christian Schörkhuber, Markus Zaunschirm, and Robert Höldrich. Binaural rendering of ambisonic signals via magnitude least squares. *Fortschritte der Akustik–DAGA*, pages 339–342, 2018.

[TC19] Joseph G Tylka and Edgar Y Choueiri. Domains of practical applicability for parametric interpolation methods for virtual sound field navigation. *Journal of the Audio Engineering Society*, 67(11):882–893, 2019.

[Teu07] Heinz Teutsch. *Modal array signal processing: principles and applications of acoustic wavefield decomposition*, volume 348. Springer, 2007.

[Ven14] Jakob Vennerød. Binaural reproduction of higher order ambisonics-a real-time implementation and perceptual improvements. Master's thesis, NTNU, 2014.

[VGR19] Julian Vanasse, Andrea Genovese, and Agnieszka Roginska. Multi-channel impulse response measurements in matlab: An update on scanir. In *Audio Engineering Society Conference: 2019 AES International Conference on Immersive and Interactive Audio*. Audio Engineering Society, 2019.

[Web] Web audio api. https://webaudio.github.io/web-audio-api/index.html. (Accessed on 04/26/2021).

[Wei] Eric W. Weisstein. Associated legendre polynomial – from wolfram mathworld. https://mathworld.wolfram.com/AssociatedLegendrePolynomial.html. (Accessed on 02/19/2021).

[Wha] What is cultural heritage? (article) — khan academy. https://www.khanacademy.org/humanities/special-topics-art-history/arches-at-risk-cultural-heritage-education-series/arches-beginners-guide/a/what-is-cultural-heritage. (Accessed on 05/01/2021).

[Zal19] Gabriel Zalles. Effects of capsule coincidence in foa using mems: Objective experiment. In *Audio Engineering Society Convention 147*. Audio Engineering Society, 2019.

[ZF12] Franz Zotter and Matthias Frank. All-round ambisonic panning and decoding. *Journal of the audio engineering society*, 60(10):807–820, 2012.

[ZF19] Franz Zotter and Matthias Frank. *Ambisonics: A practical 3D audio theory for recording, studio production, sound reinforcement, and virtual reality*. Springer Nature, 2019.

[ZFKC14] Frank Zotter, Matthias Frank, Matthias Kronlachner, and Jung-Woo Choi. *Efficient phantom source widening and diffuseness in ambisonics*. 2014.

[ZFP13] Franz Zotter, Matthias Frank, and Hannes Pomberger. Comparison of energy-preserving and all-round ambisonic decoders. *Fortschritte der Akustik, AIADAGA,(Meran)*, 2013.

[ZPN12] Franz Zotter, Hannes Pomberger, and Markus Noisternig. Energy-preserving ambisonic decoding. *Acta Acustica united with Acustica*, 98(1):37–47, 2012.

[ZSH18] Markus Zaunschirm, Christian Schörkhuber, and Robert Höldrich. Binaural rendering of ambisonic signals by head-related impulse response time alignment and a diffuseness constraint. *The Journal of the Acoustical Society of America*, 143(6):3616–3627, 2018.