The Ambisonic Decoder Toolbox: Extensions for Partial-Coverage Loudspeaker Arrays

Aaron J. HELLER
AI Center, SRI International
Menlo Park, CA, US
heller@ai.sri.com

Eric M. BENJAMIN
Surround Research
Pacifica, CA, US
ebenj@pacbell.net

Abstract

We present extensions to the Ambisonic Decoder Toolbox to efficiently design periphonic decoders for non-uniform speaker arrays such as hemispherical domes and multilevel rings. These techniques include modified inversion, AllRAD, and spherical Slepian function-based decoders. We also describe a new backend for the toolbox that writes out full-featured decoders in the Faust DSP specification language, which can then be compiled into a variety of plug-in formats. Informal listening tests and performance measurements indicate that these decoders work well for speaker arrays that are difficult to handle with conventional design techniques. The computation is relatively quick and more reliable compared to non-linear optimization techniques used previously.

Keywords

Ambisonic decoder, HOA, hemisphere, Faust

1 Introduction

This is a paper about extensions to the Ambisonic Decoder Toolbox to efficiently design decoders for loudspeaker arrays with partial coverage of the sphere, such as domes and multilevel rings. The criteria for Ambisonic reproduction are:

- Constant amplitude and energy gain for all source directions
- At low frequencies, reproduced wavefront direction and velocity are correct
- At high frequencies, maximum concentration of energy in the source direction
- Matching high- and low-frequency perceived directions

In the case of decoders for partial-coverage arrays, we relax these to apply only to source directions that are within the covered part of the sphere, but still require that the decoder be "well behaved" for sources from other directions.

Conventional techniques for periphonic decoder design work well when the speakers are distributed uniformly around the listening position. First-order Ambisonics can be accommodated in many listening rooms; however, when moving to higher-order reproduction the need arises to place more loudspeakers below the listener. This requires placing the listening position high in the room or on an acoustically transparent floor with a space below to install speakers. Neither of these are practical for most installations, so hemispherical dome configurations are a popular alternative. In addition, it may be impractical to install speakers directly overhead, resulting in a configuration of horizontal rings of speakers at multiple heights. These configurations leave gaps in coverage below, and possibly above, the listening position.

In a previous paper, we describe a Matlab/GNU Octave[1] toolbox for generating Ambisonic decoders that uses inversion or projection to generate an initial estimate and then non-linear optimization to simultaneously maximize r_E and minimize directional and loudness errors [2012]. While this works well for small arrays, we found that increasing the Ambisonic order and number of loudspeakers causes the optimizer to converge slowly and get stuck in local minima unless the starting solution is close to optimal.[2]

In the case of hemispherical domes and multilevel rings, neither inversion or projection provide a close starting point. Once the speaker array deviates from uniform geometry, an inversion decoder will trade uniform loudness for directional accuracy by putting more energy in directions where gaps between the loudspeakers are larger. A projection decoder does just the opposite, putting equal energy into all the speak-

[1] In this paper, we use "Matlab" to refer to both Matlab and GNU Octave. Care has been taken to make sure the code runs in both; however, not all of the graphics work well in Octave. Matlab is a registered trademark of The MathWorks, Inc.

[2] A recent paper by Arteaga [2013] takes advantage of symmetries in the loudspeaker array and a reformulation of the objective function to improve the convergence behavior of the optimization process.

ers regardless of spacing, hence they are louder in directions where there are more speakers. In practice, neither provides an adequate starting point for the optimization process.

The general problem is that it is difficult to pull the sound image beyond the space where there is dense coverage. For the case of hemispheres this not only means that performance will suffer below the horizon, but that it will be poor at the horizon. Because horizontal performance is uniquely important, it is necessary to make the decoder perform well there, despite the difficulties.

New design techniques have been proposed over the last few years to handle these sorts of arrays. We have implemented these in the toolbox to make them available to a wider user group. The toolbox has been extended beyond third-order decoding, and to support component order and normalization conventions other than Furse-Malham. We also wanted to support a variety of plug-in architectures. A new decoder engine was written in the FAUST (Functional Audio Stream) DSP Specification language [Orlarey, Fober, and Létz 2009; Smith 2013a], which includes facilities for dual-band decoding, and near-field, distance, and loudness compensation.

1.1 Auditory Localization

In this paper we utilize Gerzon's two main localization models to predict decoder performance: the velocity localization vector, $\mathbf{r_V}$, and the energy localization vector, $\mathbf{r_E}$. These are defined and discussed in our previous paper on the toolbox [Heller, Benjamin, and Lee 2012] (and many other places). Briefly, these models encapsulate the primary interaural time difference (ITD) and interaural level difference (ILD) theories of auditory localization. The direction of each indicates the direction of the localization perception, and the magnitude indicates the quality of the localization. In natural hearing from a single source, the magnitude of each is exactly 1 and the direction is the direction to the source.

1.2 Math Notation

We use lowercase bold roman type to denote vectors ($\mathbf{v}$), uppercase bold roman type to denote matrices ($\mathbf{M}$), italic type to denote scalars (s), and sans serif type to denote signals (W). A scalar with the same name as a vector denotes the magnitude of the vector. A vector with a circumflex ("hat") is a unit vector, so, for example, $\hat{\mathbf{r_E}} = \mathbf{r_E}/r_E$. "$\mathbf{A}^\dagger$" is the Moore-Penrose pseudoinverse of $\mathbf{A}$ (`pinv(A)` in MATLAB) and "$\mathbf{A}^\top$" is the transpose of $\mathbf{A}$ (`A.'` in MATLAB).

2 Decoder Design Techniques for Domes and Multilevel Rings

In Ambisonics, the standard technique for deriving the basic decoder matrix, $\mathbf{M}$, is to invert the matrix, $\mathbf{K}$, whose columns are composed of the spherical harmonics sampled at the speaker positions, such that $\mathbf{M}\,\mathbf{K} = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix [Gerzon 1980; Heller, Lee, and Benjamin 2008].[3]

Because $\mathbf{K}$ is "encoding" the speaker positions, some authors call it the *reencoding matrix* and refer to the inversion as *mode matching*. In the general case, $\mathbf{K}$ is rank deficient, so the inversion must be done by least-squares or by using singular-value decomposition (SVD) and the Moore-Penrose pseudoinverse.

Problems arise when a given loudspeaker array does a poor job of sampling some of the spherical harmonics, such as sampling at or near zero crossings or having more than one zero crossing between samples. In these cases, $\mathbf{K}$ will be ill-conditioned (difficult to invert without loss of precision) and the resulting decoder will have greater energy gain in certain directions, resulting in reduced r_E and greater loudness in those directions.

In the following subsections, we discuss three strategies implemented in the toolbox:

- Use an inversion technique suited to ill-conditioned problems
- Invert a well-behaved full-sphere coverage array, map to the real array
- Derive a new set of basis functions for which the inversion is well behaved

2.1 Modified Inversion

One proposed solution is to set all of the singular values to 1 when computing the pseudoinverse [Pomberger and Zotter 2012]. This has the effect of diminishing the use of the poorly sampled spherical harmonics. The resulting decoder has constant energy (hence, loudness) in all directions, at the expense of increased directional errors.

Another solution is to use a *truncated* SVD when computing the pseudoinverse. This simply discards the poorly sampled spherical harmonics. In the conventional pseudoinverse (e.g., as

[3] The term *sampling* is used here to mean evaluating the given spherical harmonic function at a particular azimuth and elevation.

implemented in MATLAB), normalized singular values[4] less than 10^{-15} are not inverted. In a truncated SVD, a much larger threshold is used. For example, setting the threshold to $\frac{1}{2}$ puts an upper limit of 3 dB on the loudness variations, again, at the expense of increased directional errors.

The toolbox also can produce decoders that are a linear combinations of conventional pseudoinverse and these alternatives, providing a single parameter to tradeoff uniform loudness and directional accuracy. Other approaches to inverting ill-conditioned matrices have been applied to this problem, such as Tikhonov regularization [Poletti 2005] and LASSO (least absolute shrinkage and selection operator) [Chen and Huang 2013]. Currently, we have not implemented these, although the linear combination approach described above provides a result similar to Tikhonov regularization.

2.2 Hybrid Ambisonic-VBAP Decoding

The hybrid Ambisonic-VBAP approach is called "All Round Ambisonic Decoding" (AllRAD) by Zotter and Frank [2012]. Briefly, one computes a decoder for a uniform array of virtual speakers and then maps the signals for the virtual array to the real loudspeaker array using Vector Base Amplitude Panning (VBAP) [Pulkki 1997].

VBAP always produces the smallest possible angular spread of energy for a given panning direction and speaker array, hence the perceived size of a virtual source changes depending on direction. This is directly at odds with the Ambisonic approach, which tries to keep the perceived size of a virtual source constant regardless of source direction. AllRAD uses two strategies to mitigate this:

1. The number of virtual speakers is made much larger than the number of real speakers.
2. *Imaginary speakers* are inserted to fill in large gaps in the real loudspeaker array in order to keep the triangular faces of the tessellation as regular as possible.

AllRAD places the virtual speakers according to a *spherical t-design* [Hardin and Sloane 2002]. A spherical t-design of degree t is a finite set of points on a sphere, such that the integral of any polynomial of degree t or less over the sphere is equal to the average value of the polynomial

[4] the set of singular values divided by the largest one

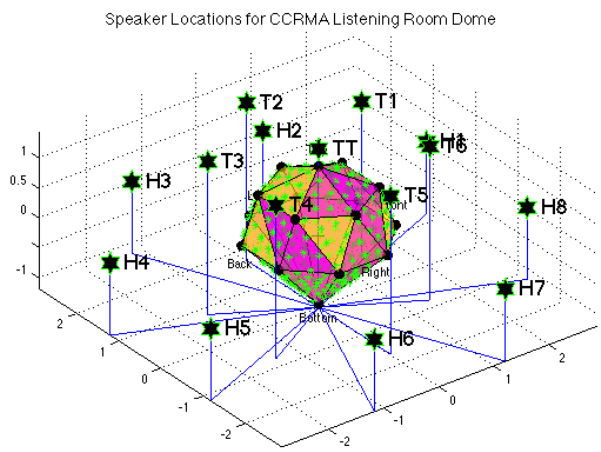
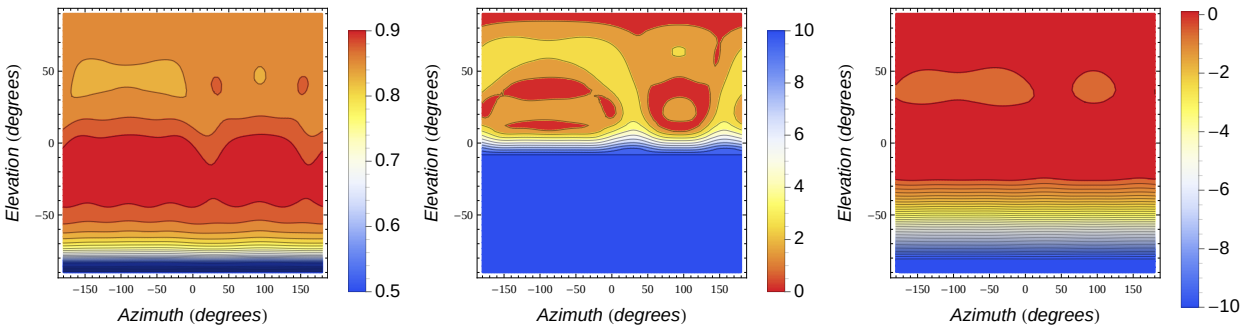


Figure 1: Plot of real speaker locations for the upper hemisphere in CCRMA's Listening Room (black hexagrams), unit sphere tessellation, and intersection points of 240 virtual speaker directions (green plus sign). The speaker at the bottom is an imaginary speaker added to keep the facets of the tessellation as regular as possible. The location of the intersection points are used to calculate the VBAP gains to the real speakers.

sampled at the points in the set. The present implementation uses the 240-point spherical t-design for the virtual array, which is the largest currently-known t-design.

There are three steps to the design of an AllRAD decoder:

1. Select a spherical t-design for the array of virtual speakers and compute a decoder for it. Because the virtual speakers are distributed uniformly on the sphere the inversion is well behaved.
   (a) Compose the matrix $\mathbf{K_V}$ whose columns are the spherical harmonics sampled at the directions of the virtual speakers.
   (b) Compute the decoder matrix for the virtual array, $\mathbf{M_V} = \mathbf{K_V}^{\dagger}$.
2. Compute the matrix of VBAP gains for each virtual speaker.
   (a) Project the positions of the real speakers onto the unit sphere.
   (b) Add imaginary speakers to the array to fill in any gaps larger than 90°. For a dome this will be one at the bottom. For a multilevel ring, one at the top and one at the bottom. The distance from the center determines how quickly



(a) r_E vs. Test Direction (b) r_E Direction Error (degrees) (c) Energy Gain (dB)

Figure 2: The AllRAD decoder's performance for the upper hemisphere of CCRMA's Listening Room. These show the (a) energy concentration, (b) directional accuracy, and (c) loudness of sources from various directions. Directional errors are clipped at 10° so that smaller errors can be seen. The plots have been quantized to make the structure clearer. Note that the Mercator projection used overemphasizes the poles.

sources fade as they move outside the region of the sphere covered by the real speaker array.

(c) Compute the triangular tessellation of the convex hull of the projected speaker positions.

(d) Determine the intersection point of the vector to each virtual speaker with the faces of the convex hull.

(e) Calculate the barycentric coordinates of each intersection point. These are the VBAP gains from that virtual speaker to the three real speakers at the vertices of the face.

(f) Assemble the matrix of the VBAP gains, $\mathbf{G}_{\mathbf{V}\to\mathbf{R}}$. This matrix has one column for each virtual speaker and one row for each real speaker. Each column will have up to three gains for that virtual speaker from the previous step. Gains to imaginary speakers are omitted.

3. The basic decoder matrix is

$$\mathbf{M} = \mathbf{G}_{\mathbf{V}\to\mathbf{R}}\,\mathbf{M}_{\mathbf{V}}.$$

Figure 1 shows the real and imaginary speaker positions, the tessellation of the speaker directions, and the intersection points of the vectors to each virtual speaker with the faces of the tessellation. The example shown is for the upper hemisphere of loudspeakers in CCRMA's Listening Room. Figure 2 shows the performance of the AllRAD decoder used in the listening tests.

2.3 Spherical Slepian Function Decoding

Spherical Slepian functions (SSF) are linear combinations of spherical harmonics that produce new basis functions that are approximately zero outside the chosen region of the sphere, but also remain orthogonal within the region of interest. This makes them suitable for decomposing spherical-harmonic models into portions that have significant energy only in selected areas [Beggan et al. 2013; Simons, Dahlen, and Wieczorek 2006]. They have been used in satellite geodesy to model the magnetic and gravitational fields of the earth from satellite data that does not cover the whole earth. In designing Ambisonic decoders, they allow us to specify a region of interest on the sphere and derive a new set of basis functions that is well conditioned within that region. Zotter et al. call this "Energy-Preserving Ambisonic Decoding" (EPAD) [2012]. The procedure implemented in the toolbox is described here.

1. Define the subset of the surface of the sphere for the decoder, $\mathcal{R} \subset S^2$, where S^2 denotes the surface of the unit sphere in $\mathbb{R}^3$. To assure good performance at the boundary, select it to be a bit larger than the area covered by the loudspeakers; for the decoder tested, we used $-30°$ to $90°$ elevation.

2. Compose the Gramian matrix, $\mathbf{G}$, of the inner products of the real spherical harmonics, $Y_{lm}(\hat{\boldsymbol{\theta}})$, over the region $\mathcal{R}$. Each element, $g_{lm,l'm'}$, of $\mathbf{G}$ is given by

$$\begin{aligned} g_{lm,l'm'} &= \langle Y_{lm}, Y_{l'm'}\rangle_{\mathcal{R}} \\ &= \int_{\mathcal{R}} Y_{lm}(\hat{\boldsymbol{\theta}})\; Y_{l'm'}(\hat{\boldsymbol{\theta}})\; d\boldsymbol{\theta} \end{aligned}$$

where lm is a single-index designator for the real spherical harmonic of degree l and order m, $\hat{\boldsymbol{\theta}} = \begin{bmatrix}\cos\theta\cos\phi & \sin\theta\cos\phi & \sin\phi\end{bmatrix}^\top$, and θ and ϕ are azimuth and elevation.

3. Compute the eigen decomposition of $\mathbf{G} \rightarrow \mathbf{U}\,\boldsymbol{\Lambda}\,\mathbf{U}^{-1}$. $\mathbf{U}$ is a unitary matrix whose columns are the eigenvectors of $\mathbf{G}$. The diagonal elements of $\boldsymbol{\Lambda}$ are the corresponding eigenvalues.

4. Compose a new matrix, $\mathbf{U}_{\text{SSF}}$, by selecting the columns of $\mathbf{U}$ with eigenvalues above some threshold, α. α should be approximately the fraction of the sphere covered by the region of interest. For a hemispherical dome, we use $\alpha = \frac{1}{2}$. This matrix transforms points in the spherical harmonic basis to points in the new SSF basis.

5. Compose the speaker reencoding matrix, $\mathbf{K}$, where the columns are the spherical harmonics sampled at each speaker direction. Transform it to the new basis, $\mathbf{K}_{\text{SSF}} = \mathbf{U}_{\text{SSF}}{}^\top\,\mathbf{K}$

6. Compute the basic decoder matrix, $\mathbf{M} = \mathbf{K}_{\text{SSF}}{}^\dagger\,\mathbf{U}_{\text{SSF}}{}^\top$.

Figure 3 shows balloon plots of the all 16 spherical Slepian basis functions for the region $-30°$ to $90°$ elevation on the sphere. Note that the first eight are concentrated in the upper hemisphere, the next two in the middle, and the last six in the lower hemisphere. The first 13 (those with $\lambda > \frac{1}{2}$) were used for the third-order decoder we tested. One observation is that this method creates basis functions that have a clearer relationship with source directions, which is not possible for the spherical harmonics above first order. Figure 4 shows the performance of the SSF decoder used in the listening tests.

2.4 Max-r_E Decoders

The basic decoder matricies, $\mathbf{M}$, calculated in the preceding sections, are transformed into max-r_E decoders by multiplying by a matrix, $\boldsymbol{\Gamma}$, whose diagonal entries are the per-order gains that maximize r_E over the sphere. $\mathbf{M}_{\text{max-}r_E} = \mathbf{M}\,\boldsymbol{\Gamma}$. The calculation of these gains is discussed in the appendix of [Heller, Benjamin, and Lee 2012].

3 In-situ Performance Measurements

The Ambisonic decoder design philosophies discussed above are generally intended to optimize the psychoacoustically based parameters of the Gerzon Energy Vector theory. It is expected that those parameters generally predict the subjective performance of the system but, they are not the same as the parameters that directly predict what is heard by the listeners. We use measurements of the ITD and ILD to gauge the localization performance in actual systems. ITDs are known to predict localization of low-frequency sounds and ILDs are known to predict the localization of high-frequency sounds.

A group of measurements were performed in CCRMA's Listening Room at Stanford University.[5] That room is equipped with 22 loudspeakers arranged as a horizontal ring of eight loudspeakers, rings of six loudspeakers at $+40°$ and $-50°$ elevation, and one loudspeaker each at the zenith and nadir. This allowed the option of either using the full spherical array or decoders designed specifically to drive the upper 15 loudspeakers as a hemisphere. One decoder was derived by using the AllRAD method and the other by using a SSF basis set.

The ITDs and ILDs created by real systems were measured by using a dummy head to record test signals reproduced from a variety of directions. The test signals are ambisonically panned exponential sine sweeps from which the impulse response is computed from each direction. Those impulse responses are binaural impulse responses, from which the ITDs and ILDs can be derived.

The ITDs were calculated by band-pass filtering the impulse responses to the bandwidth of interest and comparing the time of arrival at the two ears of the dummy head. Performing the calculation at 192 kHz sample rate gives a time resolution of 5 µs. The measurement was repeated in each of the 37 directions at 10° intervals around the horizon, and for each of the three decoders being evaluated. The result is shown in Figure 5a. All three decoders provide a plausible ITD result. The significant differences occur at the sides.

ILDs are considerably more complex than ITDs, with the major differences between the two ears occurring at frequencies above 1 kHz. As a simplification to make comparison easier, the ILD was calculated as an average level between 1 to 4 kHz. As for the ITDs, ILD was calculated at 10° intervals around the horizon. The results are shown in Figure 5b.

The three decoders produce substantially dif-

[5] https://ccrma.stanford.edu/room-guides/listening-room

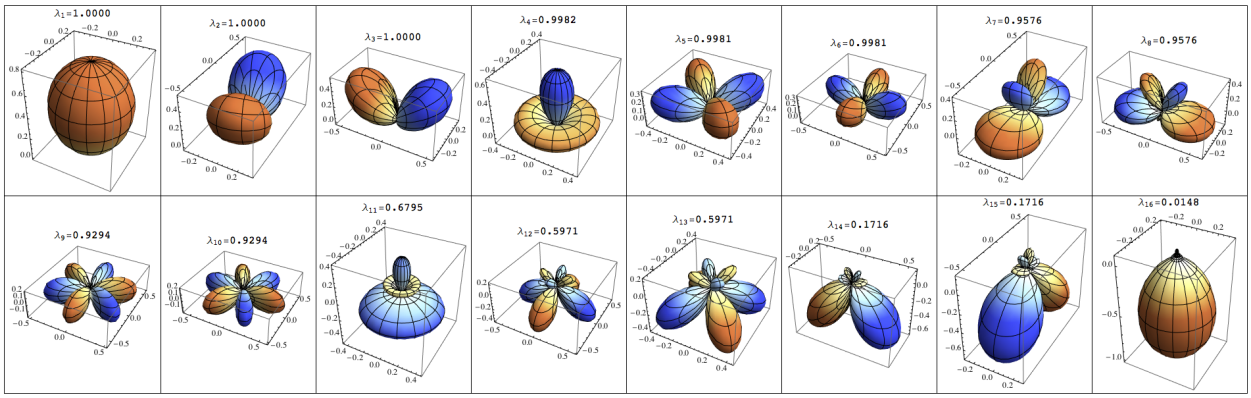
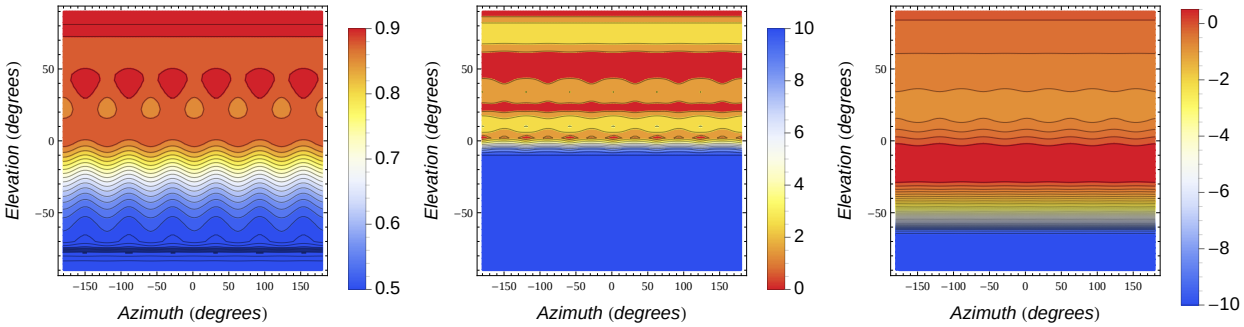


Figure 3: Balloon plots of all 16 spherical Slepian basis functions for the region $-30°$ to $90°$ elevation on the sphere. Lobes with reversed polarity are shown in blue. Note that the first eight are concentrated in the upper hemisphere, the next two in the middle, and the last six in the lower hemisphere. The first 13 ($\lambda > \frac{1}{2}$) were used for the third-order decoder we tested.



(a) r_E vs. Test Direction (b) r_E Direction Error (degrees) (c) Energy Gain (dB)

Figure 4: The Spherical Slepian function decoder's performance. These show the (a) energy concentration, (b) directional accuracy, and (c) loudness of sources from various directions. Directional errors are clipped at $10°$.

ferent values of ILD for sounds coming from the sides. It should be noted that the high values of ILD come from cancellation of signals on the opposite side of the head from the sound source by diffraction of sound traveling around the head.

Because the results of the ITD, and particularly the ILD measurements, are so complex the analysis of their effect is quite difficult and beyond the scope of the present paper. That analysis will be published in a subsequent paper.

4 Listening tests

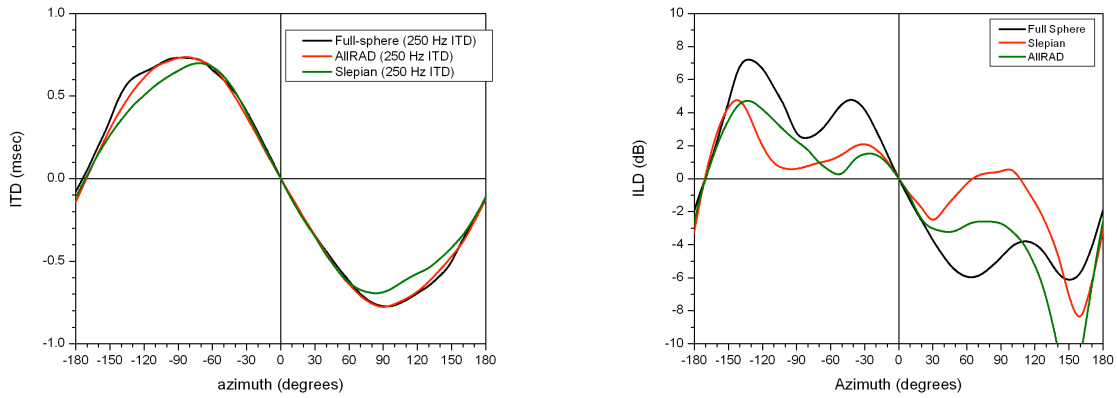
We conducted informal (non-blind) listening tests of third-order, single-band max-r_E AllRAD and SSF-based decoders using the 15 loudspeakers comprising the upper hemispherical dome in the Listening Room at Stanford's CCRMA. The decoders computed by the toolbox were saved as AmbDec configuration files and loaded into multiple instances of AmbDec so that rapid comparisons could be made.

As a reference, we also listened to full-sphere playback of the test material over all 22 loudspeakers in the Listening Room using the third-order, two-band, decoder described in the previous paper [Heller, Benjamin, and Lee 2012]. Playback levels of all three decoders were matched by ear.

The test material comprised two third-order recordings, a full-sphere mix by Jay Kadis, CCRMA's audio engineer, of "Babel" by Allette Brooks[6] and Jörn Nettingmeier's recording of *Chroma XII* by Rebecca Sanders [Nettingsmeier 2012]. Playback was directly from the Ardour sessions for each piece, which gave us the capability to move individual elements of the mix spatially to test performance from a wider variety of directions, as well as solo individual tracks.

In general, both decoders sounded quite good, providing compact and directionally accurate

[6] http://www.cdbaby.com/cd/allette4



(a) 250 Hz ITD

(b) 1 to 4 kHz ILD

Figure 5: Interaural time difference (ITD) and interaural level difference (ILD) as a function of azimuth for full-sphere, AllRAD, and SSF-based decoders. Source elevation is 0°.

imaging down to the horizontal limit of the playback array. Sources below the horizon were reproduced at the horizon, fading out as they were panned towards the nadir. The SSF-based decoder sounded brighter and more detailed than the AllRAD decoder, despite the fact that neither decoder used frequency-dependent decoding. It was also noted that with the AllRAD decoder as the listener leaned to the left and right, central sources moved in the opposite direction, whereas with the SSF-based decoder central sources remained in place.

Neither of the test decoders sounded as good as the reference dual-band, full-sphere decoder, especially in the reproduction of lower frequency percussion, which lost some of its impact. This may be attributable to the use of correct low-frequency velocity decoding ($r_V = 1$) in the reference decoder vs. wideband max-r_E decoding in the test decoders.

At the end of the listening session, we used a first-order SSF-based decoder to briefly audition a first-order Soundfield microphone recording of an orchestra made by one of the authors.[7] In this case, the instrumental balance of the orchestra was incorrect; notably, the woodwinds were almost inaudible. After the listening session, we recalled that in this recording, the microphone was hung vertically, approximately 3 meters behind and 1.5 meters above the conductor's head, placing the entire orchestra in the lower hemisphere of the recording. The first-order SSF-based decoder starts fading sources at approximately 20° *above* the horizon, which caused the instruments at the front of the orchestra to be attenuated significantly. At this point, we cannot recommend this configuration for first-order program material with significant sources in the lower-hemisphere. Possible workarounds we intend to try include inverting the vertical signal, Z, to mirror the soundfield across the $Z = 0$ plane or rotating the soundfield about the Y-axis ("tilt") in order to move important sources to the upper hemisphere.

AllRAD decoders generated by toolbox have been used for performances at Stanford's Bing Concert Hall and Studio employing CCRMA's 24-speaker, hemispherical dome, loudspeaker array. At the dress rehearsal for a performance in the Concert Hall, we were able to compare the new AllRAD decoder to the projection decoder that had been used for previous concerts. The improvement was clearly audible to all present, with increased clarity and directional focus, especially for sources behind and above the audience.

Good results have also been reported using modified inversion for a second-order decoder for a 12-speaker trirectangle array that is limited by the ceiling height of the room, leaving a large gap in coverage at the top and bottom of the array.

5 Decoding Engine

To support operation beyond third-order, a variety of plug-in architectures, and use with third-party SDKs, a new Ambisonic decoder engine

[7] Beethoven: Sym. No. 4 in B-flat Major, Op. 60, 4th Mvt. Available at http://www.ambisonia.com/Members/ajh/ambisonicfile.2008-10-30.6980317146

was implemented in FAUST. FAUST is a DSP specification language, which can target a variety of plug-in formats and operating systems.

The new implementation comprises about 250 lines of FAUST. It has no inherent limits on the Ambisonic order at which it operates and supports three modes of decoding: one decoding matrix with per-order gains ($\mathbf{\Gamma}$), one decoding matrix with phase-matched shelf filters, and dual-band, with phased-matched bandsplitting filters and two decoding matrices. The outputs can be delay and level compensated for speakers at different distances from the center of the array.

Nearfield compensation is supplied by digital state-variable realizations of Bessel filters [Smith 2013b] and can be applied at the input or output of the decoder, or turned off completely. The current implementation provides filters for operation up to fifth-order, although the toolbox includes facilities for automatically generating filters up to approximately 25th order.[8]

User adjustments are supplied for overall gain and muting, as well as crossover frequency and relative levels of high and low frequencies. All realtime controls are "dezippered" and can be accessed directly through GUI elements or via Open Sound Control.

In practice, the toolbox writes out the configuration section of the decoder and appends the implementation section, producing a single FAUST "dsp" file, containing the full decoder. The FAUST compiler (either online or local) is used to produce a highly optimized C++ class that implements the decoder, which is then wrapped in a plug-in-specific architecture file that provides the interface to the various SDKs. This is compiled to produce the plug-in file. At the time of this writing VST, AU, MaxMSP, Pd, LADSPA, LV2, Supercollider, and many others are supported on Windows, MacOSX, and Linux. In addition, an online compiler is available.

The decoder engine implementation can be used apart from the toolbox by editing the configuration options and inserting the per-order gains and matrix coefficients manually. Facilities are provided to generate configuration sections directly from existing AmbDec configuration files.

6 Channel-Order, Normalization, and Mixed-Order Conventions

At present, there are a number of channel-order and normalization conventions in use by the Ambisonics community. The toolbox implements all conventions known to the authors, including variants that adjust the gain of the omnidirectional component (W) to be compatible with B format. Internally, each channel is annotated with its degree, order, gain relative to full orthonormalization (N3D), and Condon-Shortly phase, so additional conventions can be added easily, if needed.

Two mixed-order conventions are supported by the toolbox: the scheme used in the AMB Ambisonic File Format (`#H#P`) [Dobson 2012] and one proposed by Travis [2009], which gives resolution-versus-elevation curves that are flatter in and near the horizontal plane (`#H#V`).

7 Conclusions and Future Work

We have reported on extensions to the Ambisonic Decoder Toolbox to handle popular loudspeaker configurations that do not cover the full sphere, such as hemispherical domes and multilevel rings. It also has been extended to operate at higher Ambisonic orders and with alternate channel order and normalization conventions. To support that, and multiple plug-in architectures, we have written a new, full-featured decoder in FAUST.

In general, the ability to generate decoders quickly has proven valuable in performance settings where one has to set up quickly and the speakers are not necessarily installed in the planned locations. The other effect is that it places less emphasis on performance prediction in that a number of decoders can be generated with different methods and parameter settings, and then auditioned to determine the best one for a particular set of playback conditions.

Generating dual-band decoders from these alternate methods is an obvious extension for the toolbox, as is using the decoders as initial estimates for the optimizer. Users have requested adding bass management to the decoder implementation. We have also investigated hosting the toolbox on a server and linking directly to the online FAUST compiler, so that a user does not need to install any software to use it.

As highlighted at the end of our listening session, a significant open question with partial-coverage decoders is what should happen if a source moves into a "poor" area, for example, the zenith or nadir directions. The effect of a Spitfire flying low overhead is probably not compromised if it appears too loud or doesn't have exact localization. Conversely, a source moving

[8] The limit is imposed by MATLAB's `roots()` function.

underground may be allowed to fade.[9]

The current implementations simply discard these sources, fading out as they are panned beyond the coverage region. In the case of the AllRAD decoders, they can be brought out for further processing by simply making the imaginary speakers into real speakers in the configuration file; however, these signals cannot be simply mixed into existing speaker feeds as the coherent combination of the signals will distort the directional fidelity of the decoder, especially for sources near the horizon. One proposal is to decorrelate them using a broadband 90° phase shift and sum into the speaker feeds. Other suggestions are welcome.

The toolbox is open source and available under the GNU Affero General Public License, version 3. The FAUST code generated by the toolbox is covered by the BSD 3-Clause License, so that it may be combined with other code without restriction. Contact the authors to obtain a copy of the toolbox.

8 Acknowledgements

The authors thank Fernando Lopez-Lezcano, who encouraged us to address this topic and helped carry out the measurements and listening tests. We also thank Andrew Kimpel, Marc Lavallée, and Paul Power who have been using the toolbox and have provided helpful feedback and discussion, and Richard Lee, Jörn Nettingsmeier, Bob Oldendorf, and the anonymous referees who made several suggestions that improved the paper.

References

Arteaga, Daniel (May 2013). "An Ambisonics Decoder for Irregular 3-D Loudspeaker Arrays," in: *134th Audio Engineering Society Convention*. Rome.

Beggan, Ciarán D. et al. (Mar. 2013). "Spectral and spatial decomposition of lithospheric magnetic field models using spherical Slepian functions," in: *Geophysical Journal International* 193.1, pp. 136–148.

Chen, Fei and Qinghua Huang (2013). "Sparsity-based higher order ambisonics reproduction via LASSO," in: *Signal and Information Processing (ChinaSIP), 2013 IEEE China Summit & International Conference on*, pp. 151–154.

[9]Since the treatment of these sources depends to some degree on the producer's intent, we suggest that new full-sphere sound transmission standards, such as MPEG-H 3D Audio, should include provisions for "rendering hints", along the lines of the downmix metadata in Dolby Digital. [Dolby Laboratories, Inc 2005]

Dobson, Richard (2012). *The AMB Ambisonic File Format*. Accessed 1 Feb 2014. URL: http://people.bath.ac.uk/masrwd/bformat.html.

Dolby Laboratories, Inc (Oct. 2005). *Dolby Metadata Guide*. Tech. rep. S05/14660/16797.

Gerzon, Michael A. (Feb. 1980). "Practical Periphony: The Reproduction of Full-Sphere Sound," in: *65th Audio Engineering Society Convention Preprints*. 1571. London.

Hardin, R. H. and N. J. A. Sloane (2002). *Spherical Designs*. Accessed 1 Feb 2014. URL: http://neilsloane.com/sphdesigns/.

Heller, Aaron J., Eric M. Benjamin, and Richard Lee (Mar. 2012). "A Toolkit for the Design of Ambisonic Decoders," in: *Linux Audio Conference 2012*, pp. 1–12.

Heller, Aaron J., Richard Lee, and Eric M. Benjamin (Dec. 2008). "Is My Decoder Ambisonic?" In: *125th Audio Engineering Society Convention, San Francisco*, pp. 1–21.

Nettingsmeier, Jörn (Mar. 2012). "Field Report II – Capturing *Chroma XII* by Rebecca Saunders," in: *Linux Audio Conference 2012*.

Orlarey, Yann, Dominique Fober, and Stephane Létz (2009). "Faust: an Efficient Functional Approach to DSP Programming," in: *New Computational Paradigms for Computer Music*. Ed. by Gérard Assayag and Andrew Gerzso. Delatour.

Poletti, Mark (Nov. 2005). "Three-Dimensional Surround Sound Systems Based on Spherical Harmonics," in: *Journal Of The Audio Engineering Society* 53.11, p. 1004.

Pomberger, Hannes and Franz Zotter (Mar. 2012). "Ambisonic panning with constant energy constraint," in: *DAGA 2012, 38th German Annual Conference on Acoustics*.

Pulkki, Ville (June 1997). "Virtual Sound Source Positioning Using Vector Base Amplitude Panning," in: *Journal Of The Audio Engineering Society* 45.6, pp. 456–466.

Simons, Frederik J., F. A. Dahlen, and Mark A. Wieczorek (Sept. 2006). "Spatiospectral Concentration on a Sphere," in: *SIAM review* 48.3, pp. 504–536.

Smith, Julius O (June 2013a). *Audio Signal Processing in FAUST*. Accessed 1 Feb 2014. URL: https://ccrma.stanford.edu/~jos/aspf/.

Smith, Julius O. (2013b). *Digital State-Variable Filters*. Accessed 1 Feb 2014. URL: https://ccrma.stanford.edu/~jos/svf.

Travis, Chris (June 2009). "A New Mixed-Order Scheme for Ambisonic Signals," in: *Proc. 1st Ambisonics Symposium*, pp. 1–6.

Zotter, Franz and Matthias Frank (Nov. 2012). "All-Round Ambisonic Panning and Decoding," in: *Journal Of The Audio Engineering Society* 60.10, pp. 807–820.

Zotter, Franz, Hannes Pomberger, and Markus Noisternig (Jan. 2012). "Energy-Preserving Ambisonic Decoding," in: *Acta Acustica united with Acustica* 98.1, pp. 37–47.