vdt Verband Deutscher Tonmeister e.V.

Ambisonic Decoder Description (.ADD)

Presented * by VDT.

Developing a new file format for Ambisonics decoding matrices

G. Arlauskas, J. Ohland, H. Schaar
University of Applied Sciences Darmstadt, Germany,
Email: {jonas.ohland, gabriel.arlauskas, henning.schaar}@stud.h-da.de

Abstract

Different software solutions have been developed for the calculation and implementation of Ambisonics decoding matrices. The present paper presents and describes a new data file format which can be used as an intermediate between solutions.

Currently available software solutions use particular data conventions causing difficult compatibility and exchangeability. In the present work an open-source toolkit is developed for storing, handling and using Ambisonics decoding matrices. The toolkit includes tools for conversion from common matrix data conventions to the ADD-format and back, calculating decoding matrices, decoding Ambisonics signals and extracting existing matrices from external decoding tools.

The new ADD-format and toolkit enables increased flexibility in production workflows and eliminates the drawbacks and limitations regarding compatibility between software solutions.

1. Introduction

Spatial audio can be considered as an extension of already established surround sound, with the difference of covering a full-sphere instead of just a horizontal plane. Ambisonics has been established as a reliable mathematical way represent the sound field components [citation needed]. Those components are obtained by encoding a sound source with spherical harmonics.

Spherical harmonics are an infinite set of harmonic functions defined over the surface of a sphere and can be defined as

$$Y_\ell^m(\boldsymbol{\vartheta}) = N_{\ell,|m|}^X P_\ell^{|m|}(\cos\theta) \begin{cases} \cos(m\phi), & \text{for } m \geq 0 \\ \sin(|m|\phi), & \text{for } m < 0 \end{cases} \tag{1}$$

where ϑ is the angular direction, P is the associated Legendre polynominal of order ℓ and index m and N is a normalisation factor defined by a method X, either

$$N_{\ell,m}^{\mathrm{SN3D}} = \sqrt{(2-\delta_m)\,\frac{(\ell-|m|)!}{(\ell+|m|)!}}, \delta_m \begin{cases} 1 & \text{if } m = 0 \\ 0 & \text{if } m \neq 0 \end{cases} \tag{2}$$

or

$$N_{\ell,m}^{\mathrm{N3D}} = N_{\ell,m}^{\mathrm{SN3D}} \sqrt{2\ell+1} \tag{3}$$

And the encoding of a set of k input signals g_l can be expressed as

$$\hat{\boldsymbol{\phi}} = \sum_{k=1}^{\mathrm{K}} \boldsymbol{y}\left(\boldsymbol{\vartheta}_k\right) g_k \tag{4}$$

where

$$\boldsymbol{y}(\boldsymbol{\vartheta}) := \left[Y_0^0(\boldsymbol{\vartheta}), \cdots, Y_\ell^m(\boldsymbol{\theta}), \cdots\right]^{\mathrm{T}}$$

* Please note that the papers at ICSA can be published by VDT, in print, online and as PDF download.

or in a simplified form as

$$\hat{\phi} = \boldsymbol{\Upsilon} \boldsymbol{g} \tag{5}$$

where

$$\begin{aligned} \boldsymbol{g} &:= [g_1, \dots g_{\mathrm{K}}] \\ \boldsymbol{\Upsilon} &:= [y(\vartheta_1), \dots, y(\vartheta_{\mathrm{K}})] \end{aligned}$$

In most cases the signals $\boldsymbol{s}$ decoded for various loudspeaker setups can be obtained by applying a decoding matrix D

$$\boldsymbol{s} = \boldsymbol{D}\boldsymbol{\phi} \tag{6}$$

Thus decoding an Ambisonics signal, as long as the position of the speakers is constant, is a static operation with low complexity once the matrix has been calculated.

In recent years, numerous approaches have been presented to calculate these matrices. Above all, the approaches differ in that they are more or less suitable for certain speaker systems and playback situations and contents. A good overview of existing approaches, their advantages and disadvantages can be found in [1] [2] [3] [4] [5].

Often these methods are difficult to use in practice. Many of the methods described exist as implementations only for applications specialized in the field of research.

For example, decoding matrices well suited for irregular loudspeaker layouts can be calculated with implementations created for Matlab, like the EPAD [5] or CSAD [4] method, but using them for example in Max/MSP with the well known plugin *ambidecode~* [6] proves to be difficult. Although both solutions support importing and exporting of decoding matrices as files, the formats are not compatible with each other, even though they both contain the same data.

To overcome compatibility problems like these, we propose the .add format. It should serve as a bridge between different solutions while still providing enough flexibility to incorporate all common features, aswell as future possibilities.

2. Method

In order to design such a format, firstly it is recommendable to get an overview of the requirements that meet existing formats. Obviously, it is necessary to describe at least one decoding matrix D which transfers an incoming set of SH ϕ to the reproduction channels s. Especially interesting though, is the additional data produced by the applications we analyzed.

In this context, we investigated the following solutions:

ambidecode~ ambidecode~ from [7] and described in [6] allows to export and import the internal matrix as an xml file. In addition, the expected normalization of the incoming Ambisonics signal as well as the layout of the speaker system, a gain factor for each output and a set of decoding weights for the ambisonic components can be exported to a separate file.

Ambix decoder In addition to the matrix, the configuration files for the ambix decoder also contain information about the expected order of spherical harmonics and a gain factor for the entire matrix.[8]

Compact higher-order Ambisonic Library This is a collection of Matlab function for use with higher order ambisonics. None of these functions was explicitly written to generate files, but this can be done easily. [9]

IEM AllRAD decoder The AllRAD decoder by Daniel Rudrich exports not only the matrix and some metadata such as a name and a description, but also information about the expected normalization of the ambisonics signal, a desired weighting of Ambisonics components per order, and the layout of the target rendering system. [10]

IEM Simple Decoder The Simple Decoder is unable to produce a matrix itself, but can read files produced by AllRAD Decoder and use the matrices it contains for ambisonic decoding. In addition, one can specify a subwoofer channel, which will be taken from the matrix output and passed through a high pass filter after decoding. [10]

Ambilibrium Ambilibrium [11] exports configuration files for the Ambix decoder and the format used by the IEM.

AmbDec AmbDec enables the seperation of the Ambisonics signal into two frequency bands and to then decode it with different decoding matrices. Thus, the format produced by the AmbDec offers the possibility to store two matrices, a crossover frequency and a relative gain factor for both frequency bands. Information about the expected normalization, the speaker layout and whether the decoder should make a latency compensation when all speakers are not on the surface of a sphere can also be represented.[12]

ambisonics decoder toolbox The ambisonics decoder toolbox exports ambdec files and configuration files for the ambix decoder. [13]

3. Design

For the design of the .add format we decided to orient ourselves very close to the design of the configuration files for the IEM plugins. However, with the addition of an optional filter stage, optional additional matrices and extended metadata. All filters and all outputs can be named. An .add file contains a creation date, author information, details about the software it was created with and a version number. To avoid compatibility problems, a format revision is saved in each file.

[14] and [15] describe the advantages of multiband decoding. Therefore, the .add format supports the basic description of filters applied to the Ambisonics signal before the decoding step with corresponding matrices.

In the description of the filters, we have decided to restrict ourselves to specifying the cut-off frequencies and to leave the filter design to the implementation, but we encourage the use

of phase-matched IIR Filters to preserve uniform frequency response over all directions.

Channel ordering in .add files is done according to the ACN standard, where the channel number can be detemined algorithmically:

$$ACN = \ell^2 + \ell + m \tag{7}$$

The expected type of normalization is specified with each file and may be either SN3D or N3D as in (2) and (3) respectively.

As a container format, we chose JSON for a variety of reasons. JSON is widely used and supported by many programming languages. The structure of a JSON object can be represented by native constructs in memory which can be operated intuitively on. Furthermore, it is human-readable and can be edited with a simple text editor. Compared to XML, the memory requirements of JSON strings are much lower.

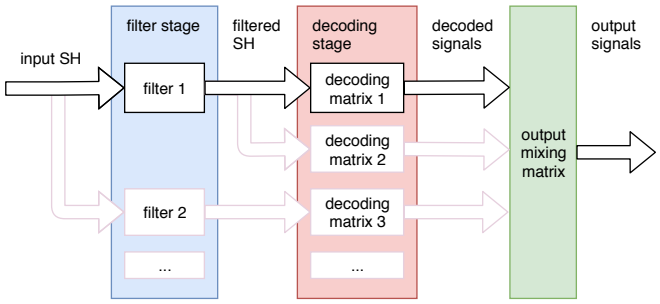

Fig. 1: Schematic representation of an ambisonic decoder that can be described by the .add format.

4. Implementation

2 shows a complete example of creating a .add file describing an ambisonic decoder of order decoder with a single output.

```
const ADD = require("dotadd.js");
const fs = require("fs");

let add_file = new ADD();

add_file.setAuthor("My Name")
    .setName("Example Decoder");

add_file.addMatrix(new ADD.Matrix(
        [[1., 0., 0., 0.]]));

fs.writeFileSync("/path/to/output/file.add",
        add_file.export().serialize());
```

Fig. 2: Creation and export of an .add file in JavaScript

Software libraries In order to garantee the uniformity of .add files and to facilitate the adaptation, software libraries for the programming languages C ++, Python, JavaScript and Matlab are developed. These allow fast adaptation of existing program infrastructure to the .add format.

dotaddtool The dotaddtool converts the various formats for storing decoder matrices into .add files and back. This tool enables end users to work with multiple softwares, which haven't implemented the .add file format themselves. As such it also serves as an intermediate solution, before .add files have seen widespread adoption.

Decoder Catcher Another part of the toolkit is the "DotAdd Decoder Catcher" that aims for the edge case when a plugin offers no way to export decoder matrices or does not do so as intended. The tool consists of two VSTs using a peer-to-peer inter-process connection which are placed up- and downstream right next to a target decoder plugin. Since a matrix is applied passively in the processing algorithm it is not crucial which type of signal is being processed. That means sending a impulse with a value of 1 through the processor for every channel and index results in the output being solely the internal decoding matrix used in the algorithm. Because matrix data can vary depending on e.g. Ambisonics order or channel configuration, it is possible to configure the output impulse of the emitter VST to produce fitting and functional matrix data. The values of the matrix are cached on run-time, recalculated according to configurations specified by the user and lastly exported as an .add-file.

The Software libraries and tools will be released soon and can be found under [16].

5. Discussion

Along the history of Ambisonics developers have contributed to the technology in a gradual manner, making the theory practically accessible. Provision of full-fledged toolkits such as SPARTA or IEM PluginSuite has pushed the limits for widespread usage of Ambisonics technology. However, compatibility issues are still common and prevent industrial use of multi-software solutions. As for decoders it still lacks a common ground to improve further according to agreed upon standards. Our proposal is another step towards a universal workflow with this type of technology. We are optimistic about the outcome and are curious about improvements and open for discourse.

6. References

[1] Franz Zotter, Matthias Frank, and Hannes Pomberger. 2013. Comparison of energy-preserving and all-round ambisonic decoders. *Journal of the Audio Engineering Society*, 60, (January 2013), 807–820.

[2] Franz Zotter, Hannes Pomberger, and Markus Noisternig. 2010. Ambisonic decoding with and without mode-matching: a case study using the hemisphere. In *Proc. of the 2nd International Symposium on Ambisonics and Spherical Acoustics* (Paris, France). (April 2010).

[3] Franz Zotter and Matthias Frank. 2012. All-round ambisonic panning and decoding. *Journal of the Audio Engineering Society*, 60, (October 2012), 807–820.

[4] Nicolas Epain, Craig T. Jin, and Franz Zotter. 2014. Ambisonic decoding with constant angular spread. In.

[5] F. Zotter, H. Pomberger, and M. Noisternig. 2012. Energy-preserving ambisonic decoding. *Acta Acustica United with Acustica, 98(1)*, 37–47.

[6] Jan C. Schacher and Philippe Kocher. 2006. Ambisonics spatialization tools for max/msp. In *Proc. of the 2006 International Computer Music Conference*, 274–277.

[7] Zurich University of Arts. 2019. Ambisonics externals for max/msp. Retrieved 08/24/2019 from https://www.zhdk.ch/forschung/icst/software-downloads-5379/downloads-ambisonics-externals-for-maxmsp-5381.

[8] M. Kronlachner. 2014. ambiX v0.2.8 - ambisonic plug-in suite. Retrieved 08/24/2019 from http://www.matthiaskronlachner.com/?p=2015.

[9] A. Politis. 2015. Compact higher-order ambisonic library. Retrieved 08/30/2019 from http://research.spa.aalto.fi/projects/ambi-lib/ambi.html.

[10] IEM - Institute of Electronic Music and Acoustics. 2019. IEM Plugins configuration files manual. Retrieved 08/30/2019 from https://plugins.iem.at/docs/configurationfiles/.

[11] M. Romanov. 2018. Ambilibrium - a user-friendly ambisonics encoder/decoder-matrix designer tool. In *Audio Engineering Society Convention 144*. (May 2018). http://www.aes.org/e-lib/browse.cfm?elib=19541.

[12] F Adriaensen. 2019. Ambdec user manual - 0.4.2. Retrieved 08/30/2019 from https://kokkinizita.linuxaudio.org/linuxaudio/downloads/ambdec-manual.pdf.

[13] Aaron Heller. 2014. The ambisonic decoder toolbox: extensions for partial-coverage loudspeaker arrays. In *Proc. of Linux Audio Conference 2014*.

[14] Aaron Heller, Richard Lee, and Eric Benjamin. 2008. Is my decoder ambisonic? In *Audio Engineering Society Convention 125*. (October 2008). http://www.aes.org/e-lib/browse.cfm?elib=14705.

[15] J. Heller A. and E. M. Benjamin. 2018. Design and implementation of filters for ambisonic decoders. In *Proc. of the 1st International Faust Conference (IFC-18)* (Mainz, Germany). (July 2018).

[16] G. Arlauskas, J. Ohland, and H. Schaar. 2019. The .add format repository. Retrieved 08/31/2019 from https://github.com/smp-3d/dotadd.