Audio Engineering Society

Convention e-Brief 590

Presented at the 148th Convention
2020 June 2 – 5, Online

HOAST: A Higher-Order Ambisonics Streaming Platform

Thomas Deppisch[1], Nils Meyer-Kahlen[2], Benjamin Hofer[3], Tomasz Łatka[4], and Tomasz Żernicki[4]

[1]*Institute of Electronic Music and Acoustics, University of Music and Performing Arts, Graz, Austria*
[2]*Aalto Acoustics Lab, Department for Signal Processing and Acoustic, Aalto University, Espoo, Finland*
[3]*Hofer Web Solutions, Graz, Austria*
[4]*Zylia Sp. z o. o., Poznań, Poland*

Correspondence should be addressed to Thomas Deppisch (thomas.deppisch@student.kug.ac.at)

ABSTRACT

The availability of free, user-friendly software tools as well as affordable hardware is boosting interest in higher-order Ambisonics productions not only in research communities, but also in the fields of Pro Audio and Virtual Reality. However, there is no practical solution available for presenting such productions publicly in a web browser. The largest commercial platforms, for example, are limited to first- or second-order binaural playback. We introduce the higher-order Ambisonics streaming platform HOAST, a new 360° video-platform, which allows for up to fourth-order Ambisonics audio material. Apart from implementation details of state-of-the-art binaural decoding and acoustic zoom, this contribution describes the current state of multichannel web audio and related challenges.

1 Introduction

To the best knowledge of the authors, there are currently four 360° video web-players supporting first-order Ambisonics (YouTube, VeeR, SamsungVR and the open-source player Video.js with the plug-in videojs-vr) and one player supporting up to second-order Ambisonics (Facebook). We introduce the HOAST (Higher-Order Ambisonics STreaming) platform[1], a browser-based multimedia platform for 360° videos with up to fourth-order Ambisonics audio content, featuring the dedicated open-source HOAST360 audio/video web-player[2].

In section 2 we motivate key points of the implementation by reviewing in-browser audio processing via the Web Audio API and investigating multichannel support of audio codecs for four browsers on two different operating systems. In section 3 we discuss all parts of the implementation, focusing on audio processing and two key features, which are the acoustic zoom and the specific binaural decoding approach.

2 Current State of Web Audio

2.1 Web Audio API

Audio processing in web applications can be performed efficiently via the Web Audio API, which is implemented in all major browsers[3] and provides high-level access to browser-specific and highly-optimized processing routines such as convolutions. It establishes a modular audio routing graph inside an audio context, which defines parameters such as a global sample rate to ensure seamless interaction between the processing modules. The Web Audio API provides source

[1] https://hoast.iem.at
[2] https://github.com/thomasdeppisch/hoast360
[3] https://caniuse.com/#search=web%20audio%20api

	Windows 10			MacOS 10.14		
Codec	Firefox	Chrome	Edge	Firefox	Chrome	Safari
WAV	32	31	31	32	31	32
FLAC	8	8	8	8	8	8
AAC	6	8	8	8	8	8
VORBIS	8	31	31	8	31	-
OPUS	32	31	31	32	31	-

Table 1: Maximum supported number of channels for multichannel audio files using different audio codecs. Tested on Windows 10 and MacOS 10.14 with Firefox 74.0, Chrome 80.0.3987, Safari 13.1 and Edge 80.0.361.

nodes for audio synthesis and integration of external audio signals into the audio routing graph, processing nodes for various common audio processing tasks, and a destination node connecting the audio routing graph to an audio output device. Custom processing nodes running on the audio rendering thread can be added using AudioWorkletNodes in combination with AudioWorkletProcessors. The AudioWorklet interface supersedes the deprecated ScriptProcessorNode interface, which is likely to cause audio glitches as it is running on the main thread and might interfere with other tasks. AudioWorklets not only support custom processing in JavaScript but also routines written in (or compiled to) WebAssembly, which aims at running code in a browser at native speed. However, at the time of writing, AudioWorkletNodes and AudioWorkletProcessors are only supported in recent versions of Chrome and are not running on a realtime priority thread which can cause glitches[4]. In our tests, enabling the experimental Chrome flag *#audio-worklet-realtime-thread* suppresses the glitches. However, due to the limited support, we currently do not use AudioWorklets in our audio/video player.

When dealing with spatial audio technologies, the ability to do simultaneous processing of a large number of audio channels is essential. Web Audio API specifications[5] demand a minimum number of 32 channels to be supported in each audio node by the individual browser implementations. To the best knowledge of the authors, all current Web Audio API implementations establish a 32-channel limit which in turn limits Ambisonics audio processing within the Web Audio API to fourth order.

2.2 Audio Codecs

For use within the HOAST platform, a suitable audio codec has to be chosen. The ideal codec for up-to-fourth-order Ambisonics streaming should support efficient (lossy) compression, up to 25 channels per file and should be playable in all common browsers. Unfortunately, no such codec is available at the moment. To investigate the supported number of channels, a small test application[6] is created. It allows to try the playback of different audio codecs in a browser. The results are displayed in table 1.

For audio files used with HOAST, the OPUS codec was chosen as it is the lossy codec supporting the highest number of channels per file across all tested browsers except Safari. While lossy audio coding generally deteriorates sound quality and spatial imaging, Narbutt et al. [1] showed that OPUS with 32 kbps per channel leads to good sound quality and excellent localization accuracy in context of third-order Ambisonics.

3 Implementation

3.1 Adaptive Streaming

For use within an audio/video streaming platform, adaptive streaming technologies such as MPEG-DASH or HLS are desirable as they ensure smooth playback by segmenting content and automatically adapting the bitrate to match the available bandwidth of the user. For HOAST, MPEG-DASH was chosen as streaming technology as it is codec-agnostic and available implementations support favorable video and audio codecs such as VP9 and OPUS delivered inside the WebM container. MPEG-DASH support is enabled in HOAST via integration of dash.js[7], the MPEG-DASH reference client implementation provided by the DASH industry forum.

[4] https://bugs.chromium.org/p/chromium/issues/detail?id=813825

[5] https://www.w3.org/TR/webaudio/

[6] https://thomasdeppisch.github.io/MultichannelAudioCodecBrowserTester/audioCodecTest.html

[7] https://github.com/Dash-Industry-Forum/dash.js

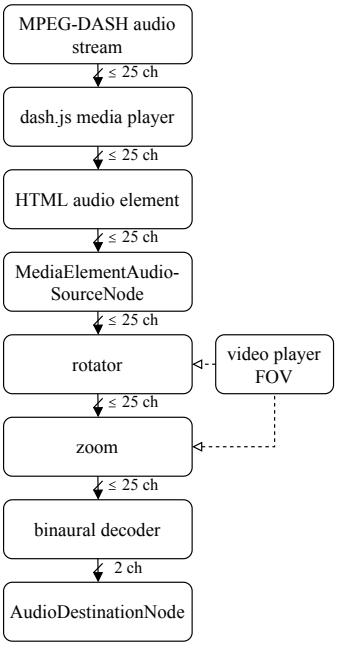


Fig. 1: Audio signal flow in HOAST, supporting up to fourth-order Ambisonics (25 audio channels).

3.2 Audio Signal Flow

To support up to fourth-order Ambisonics content in Firefox and Chromium-based browsers, the audio data is served in 25-channel OPUS files. For HOAST, we package the OPUS files in a WebM container for streaming with MPEG-DASH. On client side, the DASH stream is received via dash.js and sent to an HTML audio element, cf. figure 1. The decoded audio signals from the HTML audio element is then included into the Web Audio API audio context via a MediaElementAudioSourceNode. The Ambisonics audio stream is forwarded to custom processing nodes for Ambisonic rotation and acoustic zoom (cf. section 3.3). These are based on nodes provided by JSAmbisonics [2] and implement matrix products by utilizing GainNodes of the Web Audio API. Binaural decoding filters (cf. section 3.4) are convolved with the Ambisonics multichannel signal using Web Audio API convolver nodes, which provide a convolution routine at native performance. In the end, the two-channel binaural audio stream is sent to the client audio hardware via the AudioDestinationNode.

3.3 Zoom

Ambisonic zoom is a combination of spatial windowing and warping. These spatial transformations are based on decoding the scene to a dense set of points, which are weighted and changed in angle before re-encoding to Ambisonics [3]. Decoding the scene to the grid of t-design points with coordinates $\boldsymbol{\Theta}_P$, spatial windowing using the diagonal matrix $\boldsymbol{G}$ and re-encoding to the new points $\hat{\boldsymbol{\Theta}}_P$ is expressed as

$$\boldsymbol{T}(\alpha) = \boldsymbol{Y}(\hat{\boldsymbol{\Theta}}_P(\alpha))\, \boldsymbol{G}(\alpha)\, \boldsymbol{Y}(\boldsymbol{\Theta}_P)^{\mathrm{T}}. \tag{1}$$

As these steps do not depend on the signal, they can be computed offline, resulting in one $(N+1)^2 \times (N+1)^2$ zoom matrix $\boldsymbol{T}$ for every step in a set of zoom factors $\alpha \in [1, 2.5]$. During playback, the correct matrix is loaded depending on the current zoom factor and multiplied with the Ambisonics signal $\boldsymbol{\chi}(t)$,

$$\boldsymbol{\chi}_{\mathrm{zoomed}}(t) = \boldsymbol{T}(\alpha)\boldsymbol{\chi}(t). \tag{2}$$

When no zoom is applied, a range of $\pm 60°$ of a spherical video is visible. A zoom factor of $\alpha = 2$ indicates that the range has reduced to $\pm 30°$. Figure 2 shows t-design grid and field-of-view (FOV) before applying the spatial transformations (a), after reducing the FOV (b) and after warping (c) for a zoom factor of $\alpha = 1.8$.

The warping function (cf. figure 3) describes the change in angle of a t-design point before re-encoding and is chosen in a such a way that the points in the new, reduced field of view are warped away from the view direction, only so much as that they resemble the original FOV. Since zooming is always done with respect to the view direction, the function can be described in terms of the angle between the point and the view axis. If all points are on the surface of a unit sphere and have Cartesian coordinates (x_p, y_p, z_p), the angle to the view-axis is computed as $\theta_p = \arccos(x_p)$. After the warping function is applied to obtain the new angle $\hat{\theta}_p$, the warped points will be on a a greater circle of unit radius, which intersects both the x-axis and the point before warping, i.e. they will keep their angle $\nu = \mathrm{atan2}(z_p, y_p)$ to the x-y-plane. The new Cartesian coordinates are then obtained by

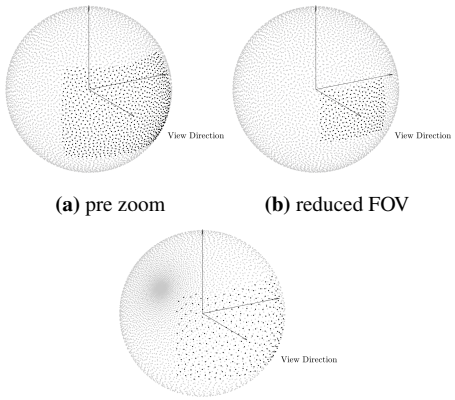
$$\hat{x}_p = \cos(\hat{\theta}_p), \tag{3}$$

$$\hat{y}_p = \sqrt{1 - \hat{x}_p^2} \,/\, (1 + \tan(\nu)^2) \cdot \mathrm{sign}(y_p), \tag{4}$$

$$\hat{z}_p = \hat{y}_p \tan(\nu). \tag{5}$$

3.4 Binaural Decoding

The Ambisonics binaural decoder creates a two-channel binaural audio stream from the multichannel Ambisonics stream. Binaural audio uses HRTF-based (head-related transfer function) localization cues to allow



(a) pre zoom **(b)** reduced FOV

(c) post zoom

Fig. 2: Distortion of FOV due to acoustic zoom, zoom factor $\alpha = 1.8$.

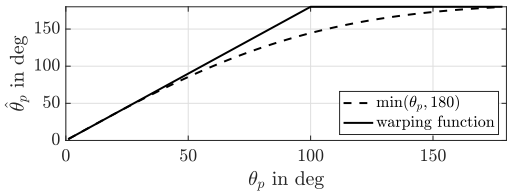


Fig. 3: Nonlinear warping function in comparison to $\min(\theta_p, 180)$, zoom factor $\alpha = 1.8$.

the perception of audio sources from arbitrary directions using regular headphones. A common approach consists of decoding the Ambisonics scene to a set of virtual loudspeakers and convolving each virtual loudspeaker signal with an HRIR (head-related impulse response) corresponding to the direction of the virtual loudspeaker, before summing up all the signals for the left and the right ear, respectively [4]. More recent approaches [5][6] avoid this intermediate step and instead solve a least squares (LS) optimization problem to create decoding filters $\boldsymbol{\omega}_{\mathrm{LS}}$ applied directly in the Ambisonics domain

$$\boldsymbol{\omega}_{\mathrm{LS}} = \arg\min_{\boldsymbol{\omega}} ||\boldsymbol{Y}_{\Upsilon}\boldsymbol{\omega} - \boldsymbol{h}_{\Upsilon}||_2^2 . \tag{6}$$

Here, $\boldsymbol{\omega}_{LS}$ holds the $(N+1)^2$ decoding filters, minimizing the least squares error between the HRTF measurements in $\boldsymbol{h}_{\Upsilon}$ and the spherical harmonics in $\boldsymbol{Y}_{\Upsilon}$ evaluated at all HRTF directions of the measurement grid Υ. For HOAST, we used a set of $P = 2702$ HRTFs, measured with the KU 100 dummy head in directions arranged according to a Lebedev grid [7]. As the simple approach described by equation 6 only gives good solutions for unpractically high Ambisonic orders (e.g. $N > 20$), we implemented the magnitude least squares (MagLS) approach proposed in [5]. In a nutshell, this solution reduces the large required spatial resolution for representing HRTFs by finding an optimal phase yielding an improved magnitude match for high frequencies. For low frequencies, the least squares solution is applied.

It can be observed that decoding using an anechoic decoder often times leads to internalized sound events, when compared to listening to an Ambisonics production in a room. Rudrich and Frank [8] showed that additional room reflections can increase externalization and demonstrated that even a simple room model decoded in first-order Ambisonics can create this effect. Taking into account results about externalization [9] and the effects of BRIR modifications with respect to externalization and timbre [10], we generated a first-order Ambisonics impulse response using the image source model. It is applied to the first-order decoding filters via convolution, yielding a total length of 2048 samples. The decoding filters for the higher-order components remain at a length of 256 taps.

3.5 Video Rendering

For video rendering, the open-source HTML5 video player framework Video.js[8] is used. In combination with the videojs-contrib-dash plug-in[9] it allows to receive and play back MPEG-DASH audio/video streams. For support of 360° equirectangular video content and dual-mono video content for head-mounted devices (HMDs), the custom videojs plug-in videojs-xr[10] is developed. It is based on the videojs-vr[11] plug-in but uses the new WebXR API, replacing the WebVR API used in videojs-vr. A polyfill makes HMDs usable in recent Firefox and Chromium-based browsers although not all of them support WebXR natively yet. Note that currently WebXR needs to be enabled explicitly in Chromium-based browsers via enabling flag *#webxr*, setting the WebXR runtime via *#webxr-runtime* and disabling *#xr-sandbox*. Internally, the video is rendered via the JavaScript 3D library three.js which itself uses a WebGL renderer. Support of further 360° and HMD formats in videojs-xr is planned for the future.

3.6 Website Implementation

In order to manage the media files and embed the custom-made media player, a web application was

[8] https://videojs.com/
[9] https://github.com/videojs/videojs-contrib-dash
[10] https://github.com/thomasdeppisch/videojs-xr
[11] https://github.com/videojs/videojs-vr

developed. On the backend side it allows basic CRUD (Create, Read, Update, Delete) tasks of the data in the underlying database. This database holds two main models: one for the meta data of the media files and another for the so called "channel", i.e. the owner of the media files. For the implementation the python-based web framework Django was chosen, since it provides an automatic and easy-to-configure admin interface. The frontend is based on the popular, responsive Bootstrap toolkit. It provides the user an intuitive interface to sort the available media by Ambisonics order or channels. Upon media selection, the associated meta data is retrieved from the backend and displayed together with the embedded HOAST360 media player. Currently, only the application maintainers are allowed to manage media files. However, the platform can easily be extended with a user management system and a media submission process for other community members at a later point.

4 Conclusion and Outlook

We showed how dynamic higher-order Ambisonics processing and binaural rendering can be done in a web browser. With the HOAST platform, we provide a platform for sharing dynamically-rendered higher-order Ambisonics audio content. The audio content can be accompanied by 360° video to be played back in the browser, where the FOV is controlled via the mouse, or via HMDs. In case of audio-only content, a 360° heatmap video is added. The simultaneous audio and video rendering is computationally demanding, which may lead to poor performance on weak computers. This problem will be tackled in the future by implementing custom processing in WebAssembly, which is unfortunately not yet well supported in browsers. As first successful experiments were already carried out, we are confident to be able to offer higher-order Ambisonics live streaming with HOAST360 soon.

Acknowledgment

This work was partly funded by the vice rectorate for research of the University of Music and Performing Arts, Graz, within the framework of a knowledge transfer project.

References

[1] Narbutt, M., O'Leary, S., Allen, A., Skoglund, J., and Hines, A., "Streaming VR for immersion: Quality aspects of compressed spatial audio," in *23rd International Conference on Virtual System & Multimedia*, pp. 1–6, IEEE, Dublin, 2017.

[2] Politis, A. and Poirier-Quinot, D., "JSAmbisonics : A Web Audio library for interactive spatial sound processing on the web," in *Proceedings of the Interactive Audio Systems Symposium*, September, 2016.

[3] Kronlachner, M. and Zotter, F., *Spatial transformations for the enhancement of Ambisonic recordings*, Master's thesis, University of Music and Performing Arts, Graz, 2014.

[4] Noisternig, M., Sontacchi, A., Musil, T., and Holdrich, R., "A 3D Ambisonic Based Binaural Sound Reproduction System," in *Audio Engineering Society Conference: 24th International Conference: Multichannel Audio, The New Reality*, 2003.

[5] Schörkhuber, C., Zaunschirm, M., and Höldrich, R., "Binaural Rendering of Ambisonic Signals via Magnitude Least Squares," *Proceedings of the DAGA*, pp. 339–342, 2018.

[6] Zaunschirm, M., Schörkhuber, C., and Höldrich, R., "Binaural rendering of Ambisonic signals by head-related impulse response time alignment and a diffuseness constraint," *J. Acoust. Soc. Am.*, 143(6), pp. 3616–3627, 2018.

[7] Bernschütz, B., "A Spherical Far Field HRIR/HRTF Compilation of the Neumann KU 100," *Proceedings of the DAGA (German Annual Conference on Acoustics)*, pp. 592—-595, 2013.

[8] Rudrich, D. and Frank, M., "Improving Externalization in Ambisonic Binaural Decoding," in *Proceedings of the DAGA German Annual Conference on Acoustics*, pp. 1466–1469, 2019.

[9] Catic, J., Santurette, S., and Dau, T., "The role of reverberation-related binaural cues in the externalization of speech," *J. Acoust. Soc. Am.*, 138(2), pp. 1154–1167, 2015.

[10] Giller, P. M., Wendt, F., and Höldrich, R., "The influence of different BRIR modification techniques on externalization and sound quality," *Proceedings on the Spatial Audio Signal Processing Symposium*, 2019.