Audio Engineering Society

Convention Paper

Presented at the International Conference on Audio for Virtual and Augmented Reality, 2018 August 20–22, Redmond, WA, USA

Individualized HRTF for playing VR videos with Ambisonics spatial audio on HMDs

Marco Binelli[1], Daniel Pinardi[1], Tiziano Nili[2] and Angelo Farina[1]

[1] *University of Parma, Dipartimento di Ingegneria e Architettura, Parco Area delle Scienze 181/A, 43124, Parma, ITALY*

[2] *ASK Industries, Reggio Emilia, ITALY*

Correspondence should be addressed to Angelo Farina (farina@unipr.it)

ABSTRACT

Current audio/video head-mounted rendering systems for virtual and augmented reality rely on a binaural approach combined with Ambisonics technology. These head-tracking systems employ generic HRTFs commonly measured with a dummy head in anechoic room. In this paper, we describe a new solution that has been designed to play 360 video files with spatial audio, developed for desktop and portable platforms and built from existing open source software. The HRTF sets can be loaded from a standard audio file chosen in an existing database or from an ad-hoc measurement. The capability to switch multiple HRTF sets while playing files has been added.

1 Introduction

The demand and interest for Augmented/Virtual Reality applications is growing quickly and worldwide: videogames is probably the most known, but also immersive 3D Audio Broadcast Streams of Live Performances [1], 3D-Audio in a Multi-Modal Teleoperation Platform for Remote Driving / Supervision [2] or Virtual Reality for Subjective Assessment of Sound Quality in Cars [3].

The number of enthusiasts and professional users is constantly increasing, thanks to the drop of prices and to the birth of simple, portable solutions. In this scenario, the key for success is to grant for everyone the same immersive and surprising experience, independently on the fact that the listener's head is more or less symmetrical or more or less similar to the dummy head employed for deriving the Spherical Harmonics to Binaural (SH2BIN) filter matrix employed for binaural rendering over headphones with head-tracking.

One of the most important aspects for attaining this target is to provide an experience as uniform as possible, both from visual and aural point of view. Two aspects in particular affect the quality of aural perception: localization capability and naturalness.

Already existing solutions are based on two sets of HRTFs, both measured with the dummy head Neumann KU100: one is the so-called "Thrive Audio" and the other is the "SADIE" [4], and both have been adopted by Google. The Thrive set was embedded in Jump Inspector, the first player capable of supporting Ambisonics 3rd order and, for this reason, adopted as target by other researches about Virtual Reality playback. Recently Google discontinued support for Jump Inspector and released, as open source, the new "Resonance Audio" library, in which binaural rendering of spatial audio is based on the SADIE HRTF set.

Nevertheless, other HRTF filter sets are available nowadays: human HRTF databases [5], fast personalized measurements in anechoic conditions, individual HRTF modelling [6] based on FEM/BEM simulations. Each of these methods have pros and cons, but in general they are all improvements over the current state of the art, as a single, standardized dummy head cannot grant to everyone the same localization capability, which is highly affected by head's dimensions and shape.

To perform a scientific investigation, a tool for testing and comparing those different approaches is required. A first attempt for testing different HRTF sets has been provided by [7], whereas the work presented here aims to build a ready-to-use application for comparing HRTF sets playing 360 degree audio/video files.

2 Development background

Rendering of spatial audio content in AR/VR/MR environments is possible with two different approaches. In the first one, a number of "sound objects" are placed in the 3D scene and the sound emitted by each object is binaurally rendered by convolving the mono signal with the HRIR (the time domain equivalent of an HRTF) of the corresponding position. This requires the availability of a huge set of HRIRs, measured with hundreds of source positions scattered uniformly on the spherical surface. If few measurements are available, a proper interpolation algorithm is needed for making the head-tracking system to be sensitive to small head or sound objects movements.

The second method is Ambisonics-based [8]. The whole acoustical scene is first encoded in a single multichannel soundtrack containing a number of channels dictated by the maximum order of the spherical harmonics expansion. The minimum requirement for spatial reproduction of a 3D sound field is 1st order Ambisonics, corresponding to a 4-channels soundtrack, but for more accurate spatial rendering typically an Ambisonics order equal to 3 is required, resulting in a 16-channels soundtrack. In this case the rendering is performed decoding the Ambisonics signal to a set of virtual speakers and from them to the ears using HRTFs. Thanks to the mathematical properties of spherical harmonics, it becomes computationally very efficient to perform 3-axes rotations on the Ambisonics soundtrack, before decoding, to achieve head-tracking. This means that binaural rendering is obtained simply convolving the 4 or 16 Ambisonics channels (Spherical Harmonics) by a static filter matrix of 4-by-2 or 16-by-2 FIR filters. Our work deals with this second approach, as in such method a reduced set of directions needs to be measured or simulated for getting this individualized SH2BIN filter matrix and the computational load is significantly smaller than with the sound objects method.

Many solutions are already available on the market, but for our purposes, we opted for open source software. Two solutions have been employed for this work: the first is a modified version of Vive Cinema, a 360 video player by HTC, working on a PC equipped with top-quality visors; the second is a brand new application for standalone systems (Samsung Gear VR, Oculus Go, etc.) using Resonance Audio, a library released by Google, available for many development platform, such as Unity or Unreal Engine.

Both solutions use a mid/side convolution scheme for reducing computational load. This means that a mid-signal is computed convolving left HRIRs with the symmetrical Spherical Harmonics (SH), while the side-signal is obtained convolving the asymmetrical SH. The left ear signal is than mid+side, while the right ear is mid-side (Figure 1). This kind of processing relies on the assumption that the HRTFs are perfectly symmetrical, that is plausible only for some people.

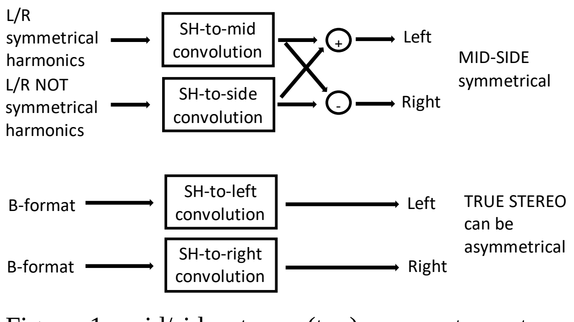


Figure 1: mid/side stereo (top) versus true stereo (bottom) Spherical Harmonics to Binaural rendering.

The main goal of this work is providing two software solutions for testing and comparing individualized HRTF sets obtained with different methods.

For testing our new apps, instead of relying solely on available sets of dummy or human heads, we preferred to also develop our own solution for fast measurements of SH2BIN filters, working both for dummy heads and humans, which is described in the following.

2.1 Fast HRTF measurement setup

The experimental measurements and processing techniques employed were carried out at LABEL – LABoratory of ELectroacoustic located at Casa della Musica, Parma (Italy), making use also of some existing software, in particular Plogue Bidule, Adobe Audition with Aurora Plugins and Matlab.

The recording equipment consists in a standard dummy head, a Neumann KU100, or alternatively, for human heads, a small couple of in-ear microphones provided by JVCKENWOOD Corporation, developed for the EXOFIELD® project (Figure 2) [9].

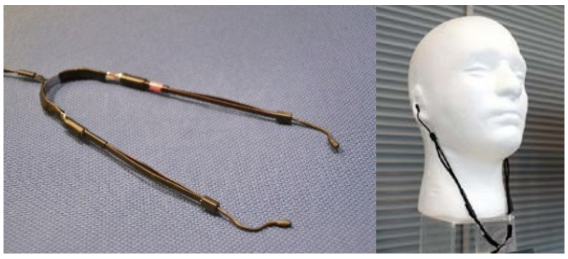


Figure 2. EXOFIELD® microphones.

The EXOFIELD® microphones are also employed for the headphones-equalization procedure, exactly as in the original EXOFIELD® method for stereo reproduction. In practice we extended the EXOFIELD® method to Ambisonics reproduction over headphones with head tracking.

The microphones were connected to a Roland Studio Capture UA-1610 Audio Interface and placed in the center of a three-dimensional rig of loudspeakers with a radius of 1.30 m. Employed loudspeakers are 16 Turbosound Impact 50, and 2 Genelec S30D in a 16.2 configuration (Figure 3).

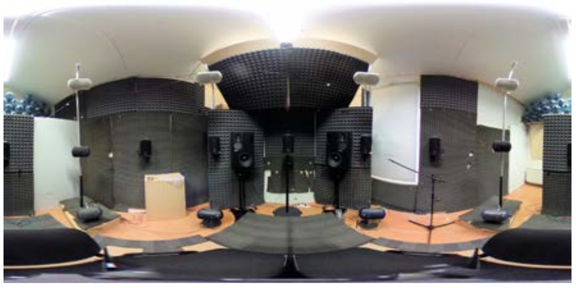


Figure 3. Measurement configuration

The 16 Turbosound speakers are arranged in a 4-8-4 configuration: eight placed in the vertexes of a cube (this geometry corresponds to the standard virtual loudspeakers configuration adopted by Google) and eight equally spaced along a ring at 45° steps, for improving performances on the horizontal plane.

Two Genelec S30D have been used as subwoofers (fed by the signals of the 16 main speakers through cross-over filters), for covering the lack in low frequencies of the small Turbosound units. Cut off frequencies have been properly chosen: 50 Hz for high pass filter, to not excite lower frequencies room modes and 120 Hz for low pass filter, in order to have a good crossover with Turbosound loudspeakers.

The response of each loudspeaker has been corrected with a pre-equalization filter. In the center of the loudspeaker rig a measurement microphone (B&K 4189 with preamplifier 2671) has been placed and a test signal is played and recorded subsequently by each loudspeaker. The test signal played is an ESS – Exponential Sine Sweep from 20 Hz to 20 kHz, 10 seconds long plus 3 seconds of silence [10]. After getting the sequence of linear IRs for the 16 loudspeakers, they were inverted with the Kirkeby method [11]. This way, a pre-equalization filter has been obtained for each loudspeaker, and has been inserted in the reproduction system so that the signal played by the corresponding loudspeaker gives a flat spectrum in the measurement point and all the signals are perfectly matched in gain and time-of-arrival (phase).

The HRTFs measurement is also performed playing a sequence of 16 exponential sine sweeps, which are routed, in sequence, to one of the 16 Turbosound speakers, while the two subwoofers are always active.

For speeding up the measurements, the sweeps can be partially overlapped, resulting in a measurement time of approximately one minute (the silence can be shortened to twice the reverberation time of the room).

The binaural signal is recorded, coming by the Neumann dummy head or by the EXOFIELD® microphones being worn by the subject. At the end of the recording, the stereo file obtained is convolved with the inverse sweep to get a sequence of 16 impulse responses, which eventually are cut of proper length (typically 16384 samples at 48 kHz), keeping only the linear response (Figure 4).

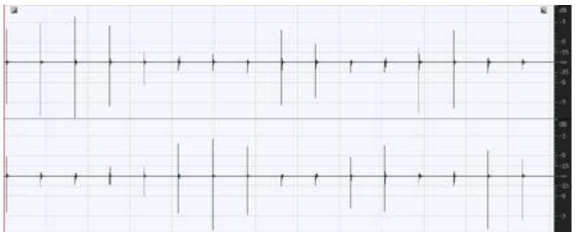


Figure 4. IRs of the Neumann KU100 dummy head

2.2 Signal processing

The signal processing is shown in Figure 5. N is the number of SH, equal to 4 in case of FOA – First Order Ambisonics or equal to 16 in case of TOA – Third Order Ambisonics. M is the number of loudspeaker measurement directions: 16 in our case, but could be even higher, for better spatial resolution, albeit this method, addressing 3rd order Ambisonics maximum, requires no more than 32 loudspeakers for optimal performances. K is the number of reproduction transducers that is 2 using headphones.

The measured HRIRs are disposed in an M-by-K (16-by-2) matrix, expressing the Speakers-to-Binaural filters (SP2BIN). But we need instead a Spherical Harmonics-to-Binaural (SH2BIN) filter matrix: for getting this, we need to compute another set of filters, converting the Spherical Harmonics signals to the Speaker feeds (SH2SP, Figure 6). This is called Ambisonics decoder: an existing VST plugin was employed, providing 3rd order Ambisonics decoding to an arbitrary set of loudspeakers [12]. It is now possible to get SH2BIN by convolving SH2SP by SP2BIN.

For avoiding coloration due to the transducers employed (microphones and headphones) an equalization procedure is mandatory. This is obtained placing the headphones over the head (either dummy or human, in this case wearing the EXOFIELD® microphones). The impulse responses are measured with an exponential sine sweep, an inverse filter for each ear is generated by using the Kirkeby method.

The SH2BIN is than convolved with the inverse filters, giving the final SH2BINeq filters. This is a filter matrix, containing a set of 4-by-2 FIR filters in case of FOA or 16-by-2 FIR filters in case of TOA.

The headphones signals are obtained by convolving the SHs (Ambisonics signals) with the SH2BINeq filters.

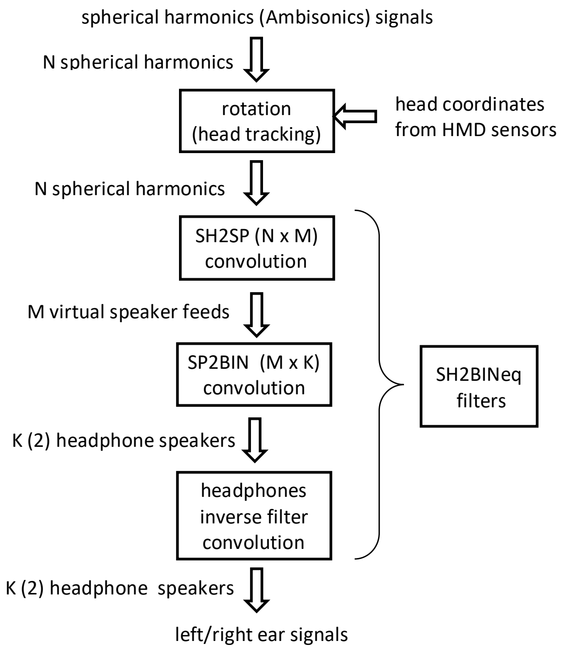


Figure 5. Signal processing flow

The computational load of these matrix convolutions is not a problem, because the SH2BINeq computation is done only once, offline and with a powerful system if needed.

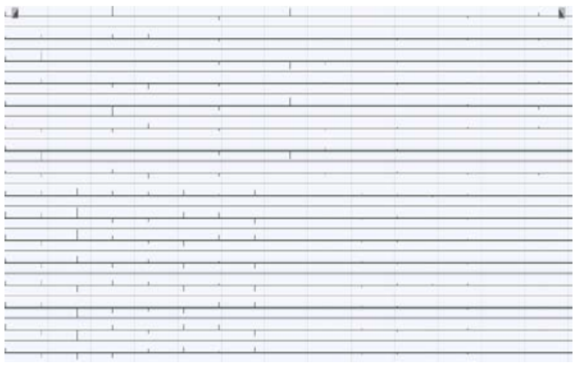


Figure 6. IEM Allrad decoding matrix (SH2SP) for 3rd order Ambisonics

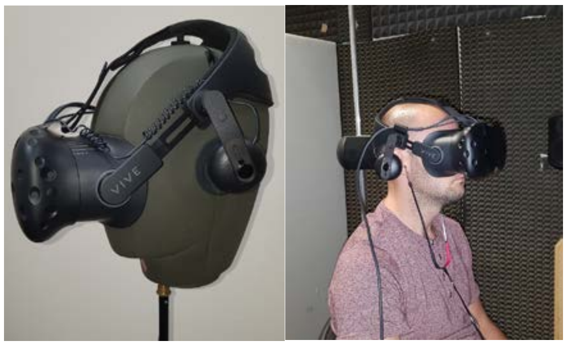


Figure 7. HMD headphones measurements set up.

Eventually, the result of convolution is trimmed, transposed and saved in a multichannel wav file. The transposition means that, instead of using a standard stereo file (for binaural rendering) containing in sequence the filters for each Spherical Harmonic signal, as common with matrix convolution software such as X-volver or Kronlachner's MCFX Convolver, the WAV file must have a number of channels equal to the number of SH signals, containing in sequence the filters for the Left and Right ears. This transposed format is required by Google Resonance, whilst Vive Cinema also recognizes the standard format.

The length of filters is kept equal to 256 samples (at 48 kHz) for each ear. In the following, these SH2BINeq filter sets are shown in case of FOA (Figure 8) and TOA (Figure 9).

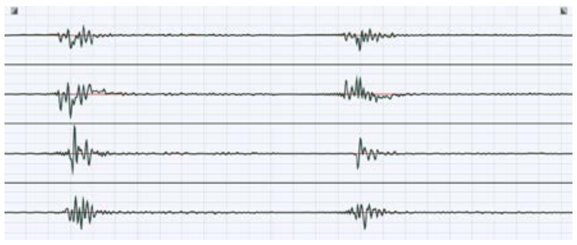


Figure 8. First order SH2BINeq decoding matrix, transposed.

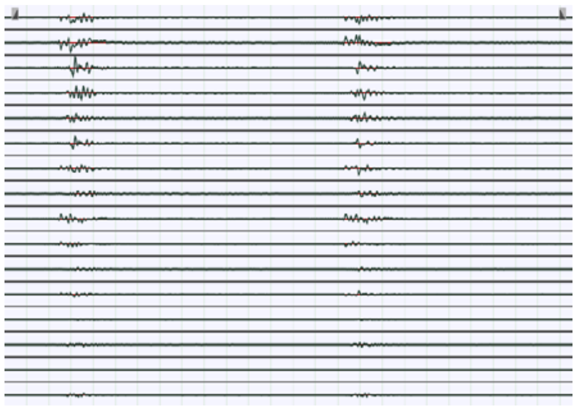


Figure 9. Third order SH2BINeq decoding matrix, transposed.

3 PC based solution

Vive Cinema has been chosen to be the software running on Windows computers to be modified because it's a GPLv3 open source project compatible with many HMD systems, e.g. HTC Vive, Oculus Rift and Samsung Odyssey (Figure 10) and also supports many audio/video file types.

Figure 10. HMDs used during development: HTC Vive (left) and Samsung Odyssey (right).

It can play high resolution 360° video files with a refresh rate of 90 Hz and spatial audio formats (1st, 2nd, 3rd order Ambix/FuMa and Facebook TBE, Mono, Stereo, ITU 5.1 or ITU 7.1). Vive Cinema is a C++ project that relies on some open source third party libraries like FFmpeg for file decoding and Kiss FFT to implement the audio convolver.

About the development environment, Microsoft Visual Studio Community 2017 on a Windows 10 operating system has been used. On the hardware side, an Alienware Aurora desktop equipped with an NVDIA GeForce 1080 TI video card was employed together with the 2 HMD systems in Figure 10.

The working base project (ver. 0.9.742, debug mode build) makes use of SH2BIN filters coming from Google Omnitone and Songbird projects and SADIE ITU 7.1. Those filters are 48 kHz, 256 samples, mono (left ear), so the audio rendering is using the mid/side symmetrical-stereo convolution (Figure 1, top).

Modification of the open-source software was relatively easy, so that it was possible to remove the nasty and pointless symmetricity hypothesis, replacing the mid/side convolution scheme with a full matrix convolution scheme, which allows for employing a complete (not-symmetrical) SH2BINeq TOA (16 channels x 2) filter set.

In detail, the features added to the project are:

- Loading coefficients of SH2BINeq filter from a WAV file. When the program starts, all WAV files that are present in a default program subfolder are loaded. It is possible to reload by pressing F6 or load a file browsing the file system by pressing F7. If the file is stereo (standard, not transposed filter matrix), each block of 256 samples is considered as a FIR filter related to a channel (Ambix-ordered filter sequence). Otherwise, if the file has 4 or more channels, each file channel is considered as a SH channel (transposed filter matrix), the first 256 samples are for the left ear and the following 256 for the right ear (like in Figure 8 and Figure 9).
- True stereo (not symmetrical) convolution. This feature allows to compare true stereo and mid/side stereo convolutions and so to test the acoustic symmetry hypothesis. The mid/side method is working with the preexisting Sadie filter set (displayed as Default) and for mono (symmetrical) SH2BINeq loaded files.
- Real-time filters switching. It provides an instant switch allowing listening comparison among all loaded filters sets plus the default one. It is possible to switch by pressing the small touchpads and triggers included in the controllers of the HMD systems or Page Up/Down keys on the keyboard.
- Current filter file name is displayed on HMD and PC monitor. On the HMD, the current filter file name is displayed in the mid-low part of the scene in the 4 cardinal directions (Figure 11). On the top-right corner of the PC screen, a list of all available SH2BIN filters is displayed, the current one is highlighted in red (Figure 12).

The whole modified project (source code and pre-compiled executable) is available for download at the following link:
http://www.angelofarina.it/Public/ViveCinema

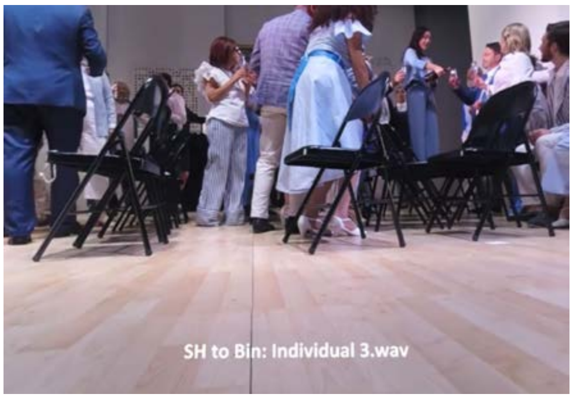
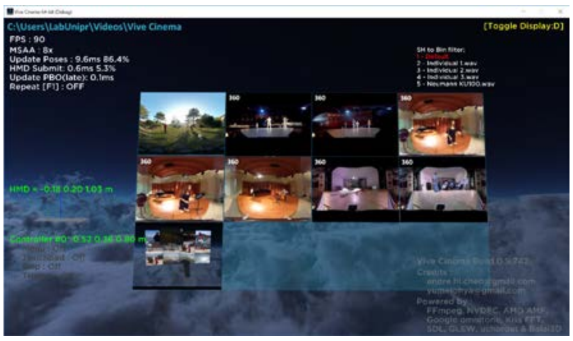


Figure 11. HMD screenshot during playback. In the lower side, the current filter name is displayed.



```
C:\Users\LabUnipr\Videos\Vive Cinema                [Toggle Display:D]
FPS : 90
MSAA : 8x
Update Poses : 9.6ms 86.4%
HMD Submit: 0.6ms 5.3%
Update PBO(late): 0.1ms
Repeat [F1] : OFF
                                        SH to Bin filter:
                                        1 - Default
                                        2 - Individual 1.wav
                                        3 - Individual 2.wav
                                        4 - Individual 3.wav
                                        5 - Neumann KU100.wav
HMD = -0.18 0.20 1.03 m
```

Figure 12. Vive Cinema start screen. In the top-right corner the list of available filters is displayed. The current filter is highlighted in red.

4 Standalone solution

Two main contenders are now dividing the market of portable VR visors: Google, with Cardboard and Daydream, and Oculus (Facebook), with Samsung Gear VR and Oculus Go.

Cardboard runs also on low cost smartphones, but with bad performances, whilst Daydream only runs on very expensive top-grade Android smartphones or on expensive stand-alone VR visors (i.e., Lenovo Mirage Solo). Instead the Oculus solution runs on not-so-expensive Samsung smartphones (S6 and above), or on the very cheap standalone Oculus GO device.

For these reasons, it has been opted (for now) for the Facebook powered solutions (Figure 13).

Figure 13. Oculus Go and Samsung Gear VR.

At the current stage of development, the SH2BINeq filter matrix cannot be loaded at the run time, it must be embedded during compilation of the code. This means that it is not possible to install a generic app and then loading separately the SH2BIN filters, it is necessary to compile a specific app for each filter set, which will be embedded. Such a limitation will be possibly removed in the next future, allowing for getting a general-purpose app which installs “as is” and lets the user to load and select a SH2BINeq filter set.

The same existing application is already running on both Gear VR and Oculus Go without the need of targeting a specific one, but the devices must be put in developer mode. For security reasons, to make the application working on a Samsung Gear VR also a protection file named Oculus Signature must be obtained for the specific smartphone employed and hardcoded in the application. These procedures will no longer be necessary after having the application certified by Google and Oculus and made it available on relative App Stores.

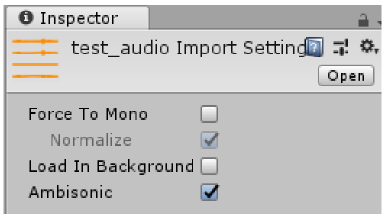
4.1 Unity3D

There are three main possibilities for creating Android applications: Android Studio, Unity3D and Unreal Engine. The last two are graphic environments that help the developer to create his own applications, with a user-friendly interface, many pre-compiled tools and the possibility to target a great number of devices, such as Microsoft PC, Xbox, OSX, iOS, Linux, Android or Sony PS. The drawback is that some compromises must be accepted

The chosen platform is Unity3D, version 2018.1f1. It is not the most powerful: rendering quality provided by Unreal is much better for videogames. Android Studio instead offers much less limitations, being the native developing platform. Unity3D otherwise is the easiest of the three and the most diffused, well supported by the community: many example projects, forums and tutorials are available.

Unity carries two principal limitations with some consequences: the maximum number of channels for a multi-channels wave file is still stuck to eight (7.1 format) and Ambisonics soundtracks must be pre-decoded by Unity itself. The first condition limits de facto to 1st order Ambisonics, being a 2nd order made of 9-channels, even if the audio engine is ready for 3rd order. The second condition instead obliges to employ separate audio and video files for being reproduced by the application.

Therefore, it is not yet possible to open directly from a folder a standard 360° equirectangular video with embedded Ambisonics soundtrack, as it happens on the Vive Cinema app. Instead, it is necessary to provide a separate file containing the soundtrack. The last must be loaded in Unity, pre-decoded as Ambisonics (Figure 14) and then saved as AssetBundles (name is “audio” and versions goes from 01 to 16). This is an owner format containing pre-processed contents that can be used by applications without the need to hardcode them: they can be downloaded or copied in memory and loaded when required. To keep the correct matching between audio and video tracks, a proper name code must be employed: for video files video01, video02 and so on, while for audio files audio.01, audio.02 and so on.



Inspector
test_audio Import Settings
Open
Force To Mono ☐
Normalize ✓
Load In Background ☐
Ambisonic ✓

Figure 14. First order Ambisonics decoding setting inside Unity3D.

Two packages of plugins are needed for making the application to work on Oculus platforms: Oculus SDK and OVR inspector. They can be both downloaded from the Oculus developer support webpage.

The interface of the application consists in a main panoramic 360° skybox where it is possible to choose the video scene to load or to quit the application and go back to the main VR screen (Figure 15, left). The choice can be done looking to the desired label and confirming with a click. Each label is connected to a scene where corresponding audio/video tracks are loaded. For these functions, two dedicated scripts have been written, and two dedicated asset packages have been downloaded: a skybox and a Unity UI.

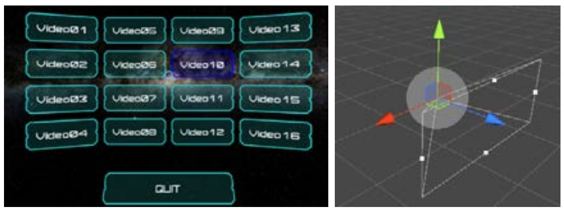
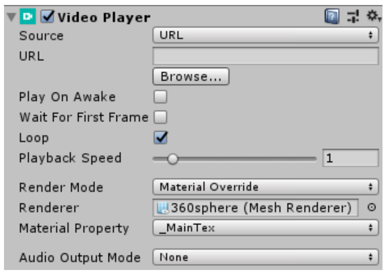


Figure 15. Application main screen (left) and rendering scene (right).

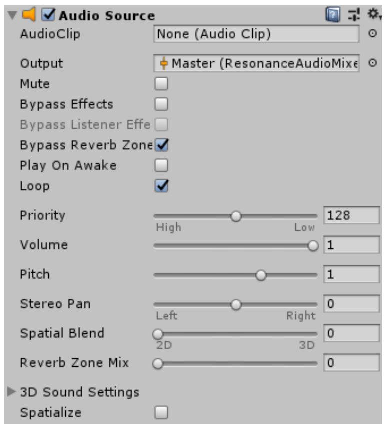
The reproduction scene is made of a main camera, placed in the middle of a 360° video projection sphere (Figure 15, right). A personalized shader has been used: the native 3D sphere object in fact has a uniform and too low polygonal density, resulting in visible artefacts at sphere's poles. The personalized material created for the sphere has an inside-out property to correct the orientation of the image, which is seen by the camera from inside the sphere and not outside.

The native Unity video player (Figure 16) and audio source (Figure 17) are attached to the sphere. In addition, the audio source is completed by the Resonance plugin script (Figure 18), thanks to which the Ambisonics soundtrack is binaurally decoded.



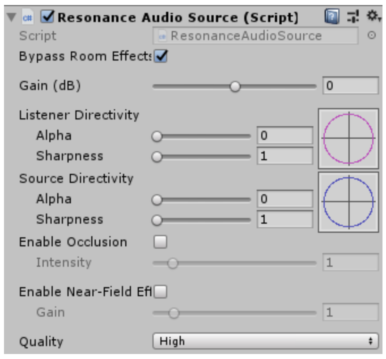
Video Player
Source URL
URL
Browse...
Play On Awake ☐
Wait For First Frame ☐
Loop ✓
Playback Speed 1
Render Mode Material Override
Renderer 360sphere (Mesh Renderer)
Material Property _MainTex
Audio Output Mode None

Figure 16. Unity Video Player.



Audio Source
AudioClip None (Audio Clip)
Output Master (ResonanceAudioMixe
Mute ☐
Bypass Effects ☐
Bypass Listener Effe ☐
Bypass Reverb Zone ✓
Play On Awake ☐
Loop ✓
Priority 128 (High – Low)
Volume 1
Pitch 1
Stereo Pan 0 (Left – Right)
Spatial Blend 0 (2D – 3D)
Reverb Zone Mix 0
3D Sound Settings
Spatialize ☐

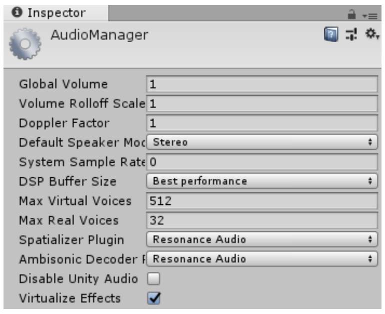
Figure 17. Unity Audio Source.



Resonance Audio Source (Script)
Script ResonanceAudioSource
Bypass Room Effects ✓
Gain (dB) 0
Listener Directivity
Alpha 0
Sharpness 1
Source Directivity
Alpha 0
Sharpness 1
Enable Occlusion ☐
Intensity 1
Enable Near-Field Eff ☐
Gain 1
Quality High

Figure 18. Resonance plugin.

The usage of Resonance Audio as Ambisonics decoder must be done inside a dedicated control panel for Unity audio settings (Figure 19).

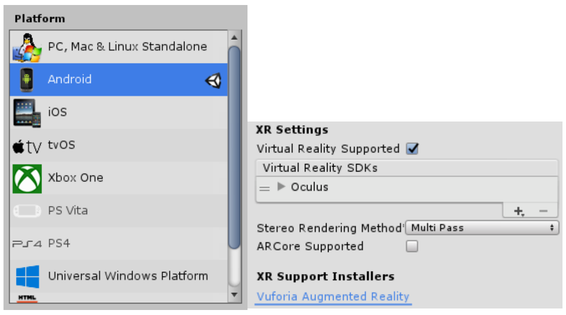


Global Volume 1
Volume Rolloff Scale 1
Doppler Factor 1
Default Speaker Moc Stereo
System Sample Rate 0
DSP Buffer Size Best performance
Max Virtual Voices 512
Max Real Voices 32
Spatializer Plugin Resonance Audio
Ambisonic Decoder F Resonance Audio
Disable Unity Audio ☐
Virtualize Effects ✓

Figure 19. Unity audio settings.

Two scripts made from scratch complete the scene: the first loads audio and video tracks when the scene is selected, the second one permits to go back to selection menu clicking the back button of input controller.

The application must be compiled selecting Android as target and Oculus as Virtual Reality SDK (Figure 20). The last comes with Unity installation, the first instead becomes available only after having downloaded and installed Android Studio with the desired APIs.



XR Settings
Virtual Reality Supported ✓
Virtual Reality SDKs
Oculus
Stereo Rendering Method Multi Pass
ARCore Supported ☐
XR Support Installers
Vuforia Augmented Reality

Figure 20. Unity target (left) and Virtual Reality SDK choice panel (right).

All the scripts are coded in C# using Visual Studio 2017.

4.2 Google Resonance Audio

Resonance Audio is a multi-platform spatial audio SDK, released open source by Google on GitHub. The following platforms are supported: Unity, Unreal, Fmod, Wwise, Daw, Web, Android and iOS. The project can be downloaded both as source code and in a precompiled version for each supported platform. Resonance Audio comes with a set of SADIE HRTFs, the mid/side convolution and an Ambisonics Decoder up to the 3rd order.

We did not manage (yet) to inject the SH2BINeq filters in the Unity's source code, which would make it possible to change them at run time.

So we employed the precompiled version for Unity and injected the filters in the binary file named libaudiopluginresonanceaudio.so, that is the Android plugin used for Ambisonics decoding. This is a compiled shared object, therefore it has been necessary to examine it in a hexadecimal editor. With proper research keywords ("RIFF..WAVE") the header of a WAV file has been found three times. In this way, it has been possible to locate the data of the wav files corresponding to SH2BIN SADIE filters for 1st, 2nd and 3rd order. A double check has been done by copying the data in three empty binary files, saved with .wav extension and opened in Adobe Audition, compared against the original .wav files downloaded from the GitHub repository of Resonance Audio project. They were actually the same: 48 kHz, 16 bits, 256 samples, mono (mid/side convolution).

Using low-level functions, a Matlab script has been coded to automatically generate a version of the plugin with the personalized Ambisonics-to-Binaural filter data. The modified plugin is employed in the player app project, so that a new personalized APK is built, containing the new SH2BINeq filters.

Of course this procedure impedes to load the SH2BIN filter at runtime, and will be replaced by a user-controllable procedure in the next future, allowing also for the user of the mobile version of our player to switch the SH2BINeq filter set at runtime.

The whole modified project (source code and pre-compiled APK) is available for download at the following link: http://www.angelofarina.it/Public/UniPrVR360app

5 Conclusions

Two solutions, one for standalone visors and one for desktop top-level visors, have been successfully coded making possible to test different sets of HRTFs.

The standalone solution still presents some limitations, connected to the development platform and the intrinsic difficulty of Android systems. It is reasonable to think that in a further version of Unity it will be possible to use 3rd order Ambisonics soundtracks, without having to pre-decode them. More inconvenient, one APK for each set of filters must be built: due to security reasons, the shared object containing Resonance Audio Android plugin cannot be loaded at runtime but must be included in the building. A solution is still under investigation.

Nevertheless, currently this is the only one existing solution to use individualized HRTFs for decoding Ambisonics soundtracks on standalone VR devices.

Acknowledgements

This work was supported by the Italian Ministry of Economic Development (MISE), FUND FOR THE SUSTAINABLE GROWTH (F.C.S) under grant agreement (CUP) B48I15000130008, project VASM ("Vehicle Active Sound Management").

The authors want to express their profound gratitude to JVCKENWOOD Corporation for making available the EXOFIELD® technology, which allowed to measure quickly and reliably human HRTFs and to perform accurate, individualized headphones equalization.

Special thanks to Hisako Shinbara, JVCKENWOOD Corporation, Tokyo, Japan

References

[1] Jacuzzi, Giordano; Brazzola, Sofia; Kares, Johannes, "Approaching Immersive 3D Audio Broadcast Streams of Live Performances" *AES Convention 142*, paper 9696 (2017).

[2] Bourdot, Patrick; Katz, Brian F.G.; Tarault, Antoine; Vézien, Jean-Marc, "The Use of 3D-Audio in a Multi-Modal Teleoperation Platform for Remote Driving/Supervision" *30th International Conference: Intelligent Audio Environments*, paper 23 (2007).

[3] Farina, Angelo; Pinardi, Daniel; Binelli, Marco; Ebri, Michele; Ebri, Lorenzo, "Virtual Reality for Subjective Assessment of Sound Quality in Cars" *AES Convention 144*, paper 10003 (2018).

[4] Gavin Kearney and Tony Doyle, "A HRTF Database for Virtual Loudspeaker Rendering", *AES Convention 139*, New York, NY, USA (2015)

[5] V. R. Algazi, R. O. Duda, D. M. Thompson and C. Avendano – "The CIPIC hrtf database", *IEEE Workshop on Application of Signal Processing to Audio and Acoustics* (2001).

[6] Chan Jun Chun, Jung Min Moon, Geon Woo Lee, Nam Kyun Kim, and Hong Kook Kim – "Deep Neural Network Based HRTF Personalization Using Anthropometric Measurements", *AES Convention 143*, New York, NY, USA (2017).

[7] Jenny, Claudia; Majdak, Piotr; Reuter, Christoph, "SOFA Native Spatializer Plugin for Unity – Exchangeable HRTFs in Virtual Reality" *AES Convention 144*, eBrief 406 (2018).

[8] Noistering et al., "A 3D Ambisonic based binaural sound reproduction system", *AES 24th International Conference on Multichannel Audio*, Banff, Alberta, Canada (2003)

[9] "JVCKENWOOD Corporation "EXOFIELD® Headphone Technology Replicates the Acoustic Space of a Room", *News release*, Las Vegas, USA, January 5, 2018 http://pro.jvc.com/pro/pr/2018/ces/JVC_Exofield.html

[10] A. Farina, "Simultaneous measurement of impulse response and distortion with a swept-sine technique" *Audio Engineering Society Convention 108*, paper 5093 (2000)

[11] O.Kirkeby, P.A. Nelson, P. Rubak, A. Farina, "Design of Cross-talk Cancellation Networks by using Fast Deconvolution", *106th AES Convention*, Munich, 8-11 may 1999.

[12] https://plugins.iem.at/