A Toolkit for Customizing the ambiX Ambisonics-to-Binaural Renderer

143rd AES Convention
Engineering Brief 403
Session EB06 - Spatial Audio
October 21st, 2017

Joseph G. Tylka (presenter)
and Edgar Y. Choueiri

3D Audio and Applied Acoustics (3D3A) Laboratory
Princeton University
www.princeton.edu/3D3A
Ambisonics-to-Binaural Rendering

HOA Signals → Render as discrete sources → Convolve with HRTFs → Binaural signals

- Sennheiser AMBEO VR Mic
- SoundField Microphone
- mh acoustics Eigenmike

Xie (2013), Head-Related Transfer Function and Virtual Auditory Display, Fig. 2.5.

https://developers.google.com/vr/concepts/spatial-audio

http://novaspire.ca/3-dimensional-binaural-audio/
Outline

• Existing tools
  • Structure of binaural rendering configuration
• The SOFA/AmbiX Binaural Rendering (SABRE) Toolkit
  • Decoding ambisonics
  • HRTF processing
  • Using the toolkit
• Summary
# Existing Tools

<table>
<thead>
<tr>
<th>Tool</th>
<th>Author</th>
<th>Platform</th>
<th>Maximum Order</th>
<th>Custom HRTF Support</th>
<th>Free</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ambi Head</td>
<td>Noise Makers</td>
<td>VST</td>
<td>1</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Harpex-X</td>
<td>Harpex Ltd.</td>
<td>VST</td>
<td>1</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>O3A Decoding VST</td>
<td>Blue Ripple Sound</td>
<td>VST</td>
<td>3</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>SPAT</td>
<td>IRCAM</td>
<td>Max, C++ Library</td>
<td>No limit</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>JSAmbisonics Library</td>
<td>A. Politis and D. Poirier-Quinot</td>
<td>JS Library</td>
<td>No limit</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ambiX Plug-ins</td>
<td>M. Kronlachner</td>
<td>VST</td>
<td>7+</td>
<td>!</td>
<td>✓</td>
</tr>
</tbody>
</table>

https://en.wikipedia.org/wiki/List_of_Ambisonic_software
Binaural Rendering Configuration

**Decoder Matrix**

- Renders HOA to discrete “speakers” signals

\[
\begin{bmatrix}
    x_1(t) \\
    x_2(t) \\
    \vdots \\
    x_Q(t)
\end{bmatrix} = D \cdot \begin{bmatrix}
    a_0(t) \\
    a_1(t) \\
    \vdots \\
    a_{N-1}(t)
\end{bmatrix}
\]

- Generally frequency independent
- Requires a known speaker grid
- Various decoder types [4,5,6]

**HRTFs**

- Spatialize discrete signals to binaural
- Stored as impulse responses with corresponding measurement grid
- Various storage formats; standardized one is SOFA [2]
- May require interpolation to decoder’s speaker grid
- May require equalization for playback
The **SOFA/AmbiX Binaural Rendering (SABRE) Toolkit**
Toolkit Architecture

SOFA HRTFs

Interpolation Settings

Speaker Grid

Decoder Type

Interpolate HRTFs

Design EQ Filters

EQ Type

Equalize HRTFs

Load/Design Decoder Matrix

ambiX Binaural Decoder
Decoding Ambisonics
Decoder Matrix

- Load an existing decoder
- Built-in ambiX presets
- Ambisonics Decoder Toolbox (ADT) [4]
- Design basic decoder
  - Pseudoinverse [5]
  - Quadrature [6]
  - “Compact” decoder
    - Pre-multiply HRTFs with decoder matrix

Pseudoinverse decoder

\[ \mathbf{D} = \left[ \begin{array}{cccc} Y_0(\hat{v}_1) & Y_0(\hat{v}_2) & \cdots & Y_0(\hat{v}_Q) \\ Y_1(\hat{v}_1) & Y_1(\hat{v}_2) & \cdots & Y_1(\hat{v}_Q) \\ \vdots & \vdots & \ddots & \vdots \\ Y_{N-1}(\hat{v}_1) & Y_{N-1}(\hat{v}_2) & \cdots & Y_{N-1}(\hat{v}_Q) \end{array} \right]^{+} \]

Compact decoder

\[
\left[ \begin{array}{ccc} h_{1}^{L,R}(t) & h_{2}^{L,R}(t) & \cdots & h_{Q}^{L,R}(t) \end{array} \right] \cdot \mathbf{D} = \tilde{\mathbf{h}}^{L,R} \cdot \mathbf{I}_{(N \times N)}
\]

Effective HRTFs

Effective decoder matrix
HRTF Processing
HRTF Interpolation

- Nearest neighbor
- Time domain:
  - Time-align onsets
  - Average responses & delays
  - Reintroduce average delay
- Alternatively, frequency domain
- Weighting schemes:
  - Linear, natural neighbor, spherical harmonic
- Interpolation threshold:
  - Nearest neighbor if a “close enough” measurement exists
  - Otherwise, use specified method

**Time domain interpolation**

1. Time-align onsets
2. Remove delay
3. Introduce delay
4. Interpolate
HRTF Equalization

- Equalization type:
  - None, frontal (free-field), diffuse, horizontal
- Regularized inverse filter

\[
Z(f) = \frac{H^*(f)}{H^*(f)H(f) + \beta(f)} \quad [9, \text{Eq. (4)}]
\]

\[
\beta_0 = 10^{-4}, \quad f_{L0} = 50 \text{ Hz}, \quad f_{H0} = 21 \text{ kHz},
\]

\[
\beta_1 = 10^{-2}, \quad f_{L1} = 20 \text{ Hz}, \quad f_{H1} = 22 \text{ kHz}.
\]
Using the Toolkit
Running in MATLAB

MATLAB code from “examples.m” distributed with the toolkit

1. Initialize the toolkit
2. Specify ambisonics order
3. Specify SOFA file and location
4. Specify output ambiX filename
5. Generate the renderer!

```
% Note: Make sure you've added the SOFA API to your MATLAB search path!
SABRE_Start;
maxOrder = 1;

% Example 1) Native HRTF grid with compacted (by default) basic decoder
sofaFile = fullfile('hrtfs','Subject2.sofa');
configFile = fullfile(SABRE_AmbiXPath,'3D3A-SABRE','ex1-Subject2.config');
[config, flags] = SABRE_BinauralRenderer(configFile, maxOrder, sofaFile);
```

Can also specify optional settings — see user manual for details.
The ambiX Binaural Plug-in

Click “open” to select binaural decoder
Summary

- Presented a MATLAB toolkit for customizing the ambiX binaural decoder
- Converts any SOFA-formatted HRTFs into ambiX rendering configuration
- Implemented basic ambisonic decoder functionality; compatible with established ambisonic decoder toolbox
- Implemented several methods of HRTF interpolation and equalization
- Source code is freely-available on GitHub: https://github.com/PrincetonUniversity/3D3A-SABRE-Toolkit

Acknowledgements

- Requires SOFA API: https://github.com/sofacoustics/sofa
References


