Virtual Acoustics for Immersive Audio Workshop
Week 2 Assignments

Orchisama Das and Gloria Dal Santo

This week's course topics will cover spatial audio fundamentals - binaural rendering with Head-Related Transfer Functions, analysis of first-order-ambisonics (FOA) encoded Spatial Room Impulse Responses (SRIRs) and binaural room impulse responses (BRIRs).

- Go to the Google Drive at `Material/Assignments/Data/Week 2`
- Download the content of the folder and place it in the `/data` folder of the workshop repository.
- **For all auralizations with the racquetball court RIR measurements, please lower the track volume to at least -24 dB and master volume to at least -18 dB before listening.**

1 Day 1 - Binaural Rendering with HRTFs

In this assignment, we will cover binaural rendering with HRIRs. For this, we have provided you with an HRIR dataset from the CIPIC library in SOFA format, which has been resampled to an equiangular grid. You will parse this dataset by using the functions in the class `HRIRReader` located in `src/spatial_audio/sofa_parser.py`.

1. Read the HRIR dataset with `HRIRReader` and plot the azimuth and elevation angles at which the HRIRs were measured on a spherical grid. Note the grid spacing between azimuth angles (θ_{grid}) and elevation angles (ϕ_{grid}). For plotting, use `spatial_audio.plot.plot_points_on_sphere` function.

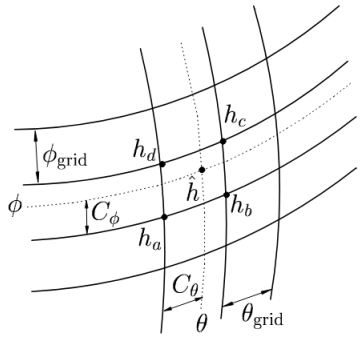


Figure 1: Bilinear interpolation of HRIRs

2. Create an `HRIRSet` dataclass from the dataset you have just read. See `src/spatial_audio/hrtf.py`.

3. Complete the function `bilinear_interpolation(new_az_res: float, new_el_res: float)` in the `HRIRInterpolator` class to interpolate the HRIRs onto a new grid. The new grid should have a spacing of 2° between azimuth angles in the range $(-180°, 180°)$ and 5° between elevation angles in the range $(-90°, 90°)$. This function should return an object of type `HRIRSet`.

   Recall the formula for the bilinear interpolation of HRIRs,

$$\begin{aligned} \hat{h}(k) &= (1 - c_\theta)\,(1 - c_\phi)\,h_a(k) + c_\theta\,(1 - c_\phi)\,h_b(k) + \\ &\quad + c_\theta c_\phi h_c(k) + (1 - c_\theta)\,c_\phi h_d(k), \\ c_\theta &= \frac{\theta \bmod \theta_{\text{grid}}}{\theta_{\text{grid}}}, \quad c_\phi = \frac{\phi \bmod \phi_{\text{grid}}}{\phi_{\text{grid}}} \end{aligned} \tag{1}$$

   where h_a, h_b, h_c, h_d are the adjacent HRIRs to a new angle-pair (θ, ϕ). Hint: the grid of HRIRs provided has uniform spacing between all azimuth and elevation angles.

4. Save the new densely sampled dataset to SOFA format using `HRIRWriter`.

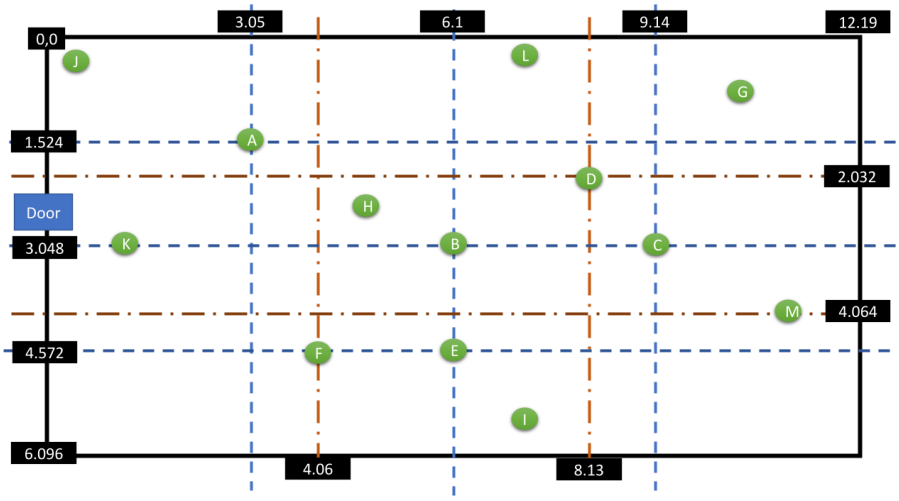
5. Dynamic rendering:

   - With head tracker :
     - Load the original and newly sampled HRTF dataset into the `sparta_binauraliser` plugin in your DAW.
     - Connect head tracker to your laptop. Open the `Bridgehead` app and make sure that the tracker outputs tracking data in the `SPARTA` format, and REAPER is set up to listen to OSC messages on port 9000.
   - Without head tracker :
     - Load the datasets to 3DTI's `Spatialisation` plugin. You can manually change the azimuth and elevation angles and listen to the rendered output.
   - Listen to a binauralized track with dynamic head rotation and comment on the differences you observe with the two datasets.

2 Day 2 - Spatial Room Impulse Responses

- For online attendees – first, download the `sparta_6DoFconv` VST file from here, and put it in your `/Library/Audio/Plug-Ins/VST/` folder. You may also need to run the following command if MacOS quarantines the binary: `sudo xattr -d com.apple.quarantine /Library/Audio/Plug-Ins/VST/sparta_6dofconv.vst`. Then, restart Reaper and look for `sparta_6DoFconv` plugin.

- For in-person attendees – you can do the same if you want to test things on your laptop, we recommend using Orchi's laptop in Studio E. Upload your SOFA files to



(a) Layout of racquetball court sine sweep measurements. I, B and C are the three source positions.

(b) Core Sound tetramic used to record the A-format RIRs in the racquetball court

Table 1: Positions in racquetball court

Position	X (m)	Y (m)	Z (m)
I	5.3	6.7	1.5
C	3.048	9.14	1.5
B	3.048	6.1	1.5
A	1.524	3.05	1.5
D	2.032	8.13	1.5
G	1	10.5	1.5
H	2.7	4.5	1.5
K	3.048	1	1.5

your personal submission folder, download it on Orchi's laptop in Studio E and load it into `sparta_6DoFconv`.

We will provide you with tetramic sine sweep recordings measured in a racquetball court [1]. The measurement layout is shown in Fig. 2a.

- Complete the multichannel deconvolution function, `rir_from_sweep()` in `utils.py`. Given the sine sweeps at receiver locations, A,D,G,H,J,K for source locations I,C and B and the dry sweep, deconvolve them to get A-format RIRs. The source and receiver locations are given in Table 1.

- Convert the A-format recordings to B-format. The normalization should be SN3D and we will follow the ACN ordering. Recall that we want to convert from A-format RIRs, $\mathbf{d}(t) = \begin{bmatrix} d_1(t) & d_2(t) & d_3(t) & d_4(t) \end{bmatrix}^\top$ to B-format RIRs, $\mathbf{h}(t) = \begin{bmatrix} W(t) & Y(t) & Z(t) & X(t) \end{bmatrix}^\top$, we have to use the spherical harmonics transformation matrix $\mathbf{Y}$,

$$
\begin{aligned}
\mathbf{h}(t) &= \mathbf{Y}^\dagger \mathbf{d}(t) \\
\mathbf{Y} &= \begin{bmatrix} Y_{0,0}(\boldsymbol{\theta}, \boldsymbol{\phi}) & Y_{1,-1}(\boldsymbol{\theta}, \boldsymbol{\phi}) & Y_{1,0}(\boldsymbol{\theta}, \boldsymbol{\phi}) & Y_{1,1}(\boldsymbol{\theta}, \boldsymbol{\phi}) \end{bmatrix}, \\
Y_{0,0}(\boldsymbol{\theta}, \boldsymbol{\phi}) &= \frac{1}{\sqrt{4\pi}} \\
Y_{1,-1}(\boldsymbol{\theta}, \boldsymbol{\phi}) &= \sqrt{\frac{3}{4\pi}} \sin\boldsymbol{\theta} \sin\boldsymbol{\phi} \\
Y_{1,0}(\boldsymbol{\theta}, \boldsymbol{\phi}) &= \sqrt{\frac{3}{4\pi}} \cos\boldsymbol{\theta} \\
Y_{1,1}(\boldsymbol{\theta}, \boldsymbol{\phi}) &= \sqrt{\frac{3}{4\pi}} \sin\boldsymbol{\theta} \cos\boldsymbol{\phi} \\
\boldsymbol{\theta}, \boldsymbol{\phi} &\in \mathbb{R}^4, \ \theta_i = \arccos(z_i), \ \phi_i = \arctan(y_i, x_i) \\
\mathbf{e}_i &= \frac{1}{\sqrt{3}} \begin{bmatrix} x_i & y_i & z_i \end{bmatrix}^\top.
\end{aligned}
\tag{2}
$$

  Here, $\mathbf{e}_i$ contains the direction vectors for each of the four capsules of the tetramic (this has been provided in the code). Hint: the calculations can be greatly simplified if you calculate θ_i, ϕ_i from $\mathbf{e}_i$ and plug it in $\mathbf{Y}$.

- Using `spatial_audio.plot.plot_spherical_harmonics`, visualise spherical harmonic functions upto order N.

- Dynamic rendering:

  - Save the B-format SRIRs in wav files with the convention `Bformat_Speaker_src=<src_name>_rec=<rec_name>.wav`. **Remember to normalise the RIRs wrt the loudest RIR**.
  - Auralize the B-format SRIR with SPARTA's `sparta_matrixconv`. Try SRIRs from a few different positions.

  - Save the B-format normalized SRIRs for each source in a separate SOFA file of convention `SingleRoomSRIR` file using the `SRIRWriter` class in `spatial_audio.sofa_parser`.
  - Auralize the SOFA files with SPARTA's `sparta_6DoFconv`.

- Binaural / out-loud playback
  - Binauralize the output of `sparta_6DoFconv` / `sparta_matrixconv` with SPARTA's `sparta_ambiBIN`. Use the HRTF dataset from the last assignment, OR
  - Listen to it out loud in Studio E with the `sparta_ambiDEC` plugin. Try rotating the soundfield.

3 Day 3 - Binaural Room Impulse Responses from SRIRs and HRIRs

In this exercise, we will convert the FOA SRIRs we saved in the SOFA format to binaural RIRs for headphone playback of spatial reverberation. Recall that there are multiple ways of doing this. The direct way is to convolve each channel of the SH-encoded SRIRs with SH-encoded HRIRs and sum them. For ambisonics order, N, the number of channels is $(N+1)^2$.

The BRIRs, $b_{L,R}(\mathbf{x}, \Omega_0, t)$ (or, BRTFs, $B_{\mathrm{L,\,R}}(\mathbf{x}, \Omega_0, f)$) at positions $\mathbf{x}$ and orientation $\Omega_0 = [\theta_0, \phi_0]$ are given by,

$$
\begin{aligned}
\mathbf{h}_{\mathrm{sh_{rot}}}(\mathbf{x}, t) &= \mathbf{D}(-\Omega_0)\mathbf{h}_{\mathrm{sh}}(\mathbf{x}, t), \\
b_{\mathrm{L,\,R}}(\mathbf{x}, \Omega_0, t) &= \left(\sum_{n=0}^{(N+1)^2-1} \mathbf{g}^{(n)}_{\mathrm{sh}_{L,R}}(-t) * \mathbf{h}^{(n)}_{\mathrm{sh_{rot}}}(\mathbf{x}, t) \right), \\
B_{\mathrm{L,\,R}}(\mathbf{x}, \Omega_0, f) &= \left(\sum_{n=0}^{(N+1)^2-1} \mathbf{G}^{(n)*}_{\mathrm{sh}_{L,R}}(f) \mathbf{H}^{(n)}_{\mathrm{sh_{rot}}}(\mathbf{x}, f) \right).
\end{aligned}
\tag{3}
$$

Here, $\mathbf{g}_{\mathrm{sh}_{L,R}}(-t) \in \mathbb{R}^{Q \times T}$ are the time-reversed HRIRs in the spherical harmonics domain ($\mathbf{G}^{(n)*}_{\mathrm{sh}_{L,R}}(f)$ are the corresponding conjugated HRTFs), $\mathbf{h}_{\mathrm{sh}}(\mathbf{x}, t) \in \mathbb{R}^{Q \times T'}$ are the SH-encoded (B-format) SRIRs at position $\mathbf{x}$ and $\mathbf{h}_{\mathrm{sh_{rot}}}(\mathbf{x}, t)$ are SRIRs rotated in the direction $-\Omega_0$ ($\mathbf{H}^{(n)}_{\mathrm{sh_{rot}}}(\mathbf{x}, f)$) are the corresponding HRTFs). $\mathbf{h}_{\mathrm{sh_{rot}}}(\mathbf{x}, t)$ is obtained from the SH-encoded SRIRs, $\mathbf{h}_{\mathrm{sh}}(\mathbf{x}, t)$ using a rotation matrix, $\mathbf{D}(-\Omega_0)$.

This is a faster way of computing BRIRs from SRIRs and HRIRs (fewer convolutions),

than the one discussed in the lecture, which is,

$$\begin{aligned}
\mathbf{h}_{\text{rot}}(\mathbf{x}, t) &= \mathbf{D}(\Omega_0)\mathbf{h}_{\text{sh}}(t), \\
h_{\text{rot}}(\mathbf{x}, t; \theta_k, \phi_k) &= \mathbf{w}^T(\theta_k, \phi_k)\mathbf{h}_{\text{sh}_{\text{rot}}}(\mathbf{x}, t), \\
&= \sum_{n=1}^{(N+1)^2-1} Y_n(\theta_k, \phi_k)\mathbf{h}^{(n)}_{\text{sh}_{\text{rot}}}(\mathbf{x}, t) \\
b_{L,R}(\mathbf{x}, \Omega_0, t) &= \sum_{k=1}^{K} h_{\text{rot}}(\mathbf{x}, t; \theta_k, \phi_k) * g_{L,R}(t; \theta_k, \phi_k), \\
B_{\text{L, R}}(\mathbf{x}, \Omega_0, f) &= \sum_{k=1}^{K} H_{\text{rot}}(\mathbf{x}, f; \theta_k, \phi_k) G_{L,R}(f; \theta_k, \phi_k)
\end{aligned} \quad . \tag{4}$$

Here, $g_{L,R}(t; \theta_k, \phi_k)$ are the HRIRs corresponding to the direction (θ_k, ϕ_k) ($G_{L,R}(f; \theta_k, \phi_k)$ are the corresponding HRTFs), and $\mathbf{w}(\theta_k, \phi_k)$ are the beamforming weights required to convert B-format SRIRs into Directional RIRs for the direction (θ_k, ϕ_k). In (4), we first convert from SRIRs to DRIRs and then convolve with HRIRs of the respective directions, whereas in (3), we convert the HRIRs into the SH domain and then convolve with the SRIRs directly. The two are mathematically equivalent (try to prove this!). **We will use the `spaudiopy` library to compute rotation matrices and SH transforms.**

- First we will convert HRIRs in the CIPIC dataset to the SH-domain for order = 1. Do this by completing the function `get_spherical_harmonic_representation(ambi_order: int)` in the dataclass `HRIRSet` in `spatial_audio.hrtf.py`. Recall that to convert HRIRs into SHD for order N, the formula is,

$$\begin{aligned}
\mathbf{g}^{(n)}_{\text{sh}_{L,R}}(t) &= \sum_{k=1}^{K} w_k Y_n(\theta_k, \phi_k) g_{L,R}(t; \theta_k, \phi_k) \\
\mathbf{G}^{(n)}_{\text{sh}_{L,R}}(f) &= \sum_{k=1}^{K} w_k Y_n(\theta_k, \phi_k) G_{L,R}(f; \theta_k, \phi_k).
\end{aligned} \tag{5}$$

  where w_k are the quadrature weights, which are used to account for the fact that measured HRIRs are not uniformly distributed over the sphere. In matrix form, this is written as,

$$\begin{aligned}
\mathbf{G}_{\text{sh}_{L,R}} &= (\mathbf{WY})^{\dagger}\mathbf{W}\mathbf{G}_{L,R}, \quad \mathbf{W} = \text{diag}(w_1, \ldots, w_K), \\
\mathbf{g}_{\text{sh}_{L,R}} &= \text{IFFT}(\mathbf{G}_{\text{sh}_{L,R}}), \\
\mathbf{Y} \text{ (ACN ordering)} &= \begin{bmatrix} Y_{0,0}(\theta_1, \phi_1) & Y_{1,-1}(\theta_1, \phi_1) & Y_{1,0}(\theta_1, \phi_1) & Y_{1,1}(\theta_1, \phi_1) \\ Y_{0,0}(\theta_2, \phi_2) & Y_{1,-1}(\theta_2, \phi_2) & Y_{1,0}(\theta_2, \phi_2) & Y_{1,1}(\theta_2, \phi_2) \\ \vdots & \vdots & \vdots & \vdots \\ Y_{0,0}(\theta_K, \phi_K) & Y_{1,-1}(\theta_K, \phi_K) & Y_{1,0}(\theta_K, \phi_K) & Y_{1,1}(\theta_K, \phi_K) \end{bmatrix}.
\end{aligned} \tag{6}$$

  We recommend using `spaudiopy.grids.calculate_grid_weights` to get the quadrature weights and `spaudiopy.sph.sh_matrix` to get the SH coefficient matrix.

- Generating BRIRs:
  - Complete the function `convert_srir_to_brir(srirs: NDArray, hrtf_set: HRIRSet, head_orientations: ArrayLike)` in `spatial_audio.spatial.py`
  - Read the SRIRs you have saved in the last assignment using the `sofar` library along with the listener positions. We will use a single source position in this case (say, I, corresponding to the first 5 SRIRs).
  - Use the SH-encoded HRIRs and the SRIRs you have read to obtain BRIRs for head orientations spanning an azimuth of $-180°$ to $+180°$ and elevation of $-90°$ to $+90°$ in $15°$ steps.
- Save the obtained BRIRs for a SINGLE source-receiver position in SOFA format using the `HRIRWriter` class. The different head orientations will be saved as source positions. For example, if your source positions is C and receiver position is A, then save the BRIRs as `racquetball_src=C_rec=A.sofa`.
- Auralize the saved SOFA file in your DAW using `3DTI Spatialisation` plugin. Listen to BRIRs for a few different source-receiver positions.

4 Day 4 - Spatial Impulse Response Rendering

We will explore one of the parametric spatial RIR rendering methods - SIRR (Spatial Impulse Response Rendering) [2] to render the racquetball court SRIR for one source-receiver location, instead of using the `sparta_ambiDEC` plugin to decode the FOA SRIRs to Studio E's setup directly, like we did in Tuesday's assignment. The AllRAD decoding method [3] that `sparta_ambiDEC` implements can adapt to any loudspeaker layout, but SIRR offers specific advantages such as better directional reproduction of early reflections and better diffuseness of late reverberation.

In this assignment, we will take any of the first-order B-format SRIRs we generated in Lecture 7's assignment, and carry out the SIRR analysis-synthesis pipeline to generate loudspeaker signals for the 22.1 setup in CCRMA's studio E.

- The following functions in the class `SIRR` in `spatial_audio/sirr.py` are described below.
  - `calculate_parameters` - This function should calculate the SIRR parameters for each time-frame – intensity vector, direction of arrival and diffuseness metric and return an object of type `SIRRParameters`. For details on how to calculate these parameters, see lecture slides for Lecture 8, or the original paper.
  - `smooth_parameters` - This function should smooth all the parameters in the dataclass `SIRRParameters` from one time-frame to another to avoid artifacts in rendering. For smoothing a parameter, $y(t)$, you can use the formula:

$$y(t) = \alpha y(t-1) + (1-\alpha) y_{\text{cur}}(t), \tag{7}$$

    where $y_{\text{cur}}(t)$ is the parameter estimate for the current frame returned by `calculate_parameters` and α is the smoothing factor.

- `process_frame` should process each STFT frame by calling `process_directional_part` and `process_diffuse_part` functions and return the output signal for the current frame.
- `process_directional_part` should use a VBAP implementation for panning the directional signal that has been provided in `spatial_audio/spatial.py`. `process_diffuse_part` uses the phase randomization method for decorrelation. This is done by imposing the magnitude spectrum of the current STFT frame onto the phase obtained from decorrelated noise signals. The noise signal has been initialised in `_init_decorrelation`. Re-initialize the noise sequence in each frame to obtain maximum decorrelation.
- `process` - This function should calculate the SIRR parameters by calling `calculate_parameters`, then smooth them by calling `smooth_parameters` and finally, process each STFT time frame by calling the `process_frame` function. Once all frames have been processed, take an inverse STFT of the 22-channel output signal to get the time-domain signals for each loudspeaker.

- Read an SRIR you have synthesized in Lecture 7's assignment and generate the loudspeaker signals obtained from SIRR for playback in CCRMA's studio E. The loudspeaker positions are specified in this JSON file. Save the first 1.5 s of the 22-channel output file.

- Log the DoAs of each time-frame and for each frequency bin by making use of `DataLogger` in `utils.py` . Save the DoAs in a mat file with the `save_history` function. Load the mat file and plot the DoAs for the 1 kHz frequency bin for each time-frame. You can use `spaudiopy`'s `plot.doa` function.

- Load the saved and truncated 22-channel output file into `sparta_multiconv`, and auralize a dry signal. Listen to the multichannel output in Studio E.

- What are some obvious artifacts you observe with the rendered output? (Eg: transient smearing, phasiness, lack of spatial clarity). What do you think causes these artifacts?

References

[1] Lloyd May, Nima Farzaneh, Orchisama Das, and Jonathan S. Abel, "Comparison of impulse response generation methods for a simple shoebox room," *Acoustics*, To appear.

[2] Juha Merimaa and Ville Pulkki, "Spatial impulse response rendering i: Analysis and synthesis," *Journal of the Audio Engineering Society*, vol. 53, no. 12, pp. 1115–1127, 2005.

[3] Franz Zotter and Matthias Frank, "All-round ambisonic panning and decoding," *Journal of the Audio Engineering Society*, vol. 60, no. 10, pp. 807–820, 2012.