ECS7012P Music and Audio Programming
Spring 2023
Final Project

Real-Time HRTF Spatial Interpolation and Binaural Rendering on Bela

Chin-Yun Yu

May, 2023

Abstract

The standard method for simulating virtual acoustic environments over headphones is by rendering binaural audio with Head-Related Transfer Functions (HRTFs). However, as available HRTFs were measured on discrete spatial points, interpolating HRTFs to directions outside the measurement points is necessary for simulating realistic virtual audio scenes. For real-time applications, such interpolation should not require intensive computation. In this report, I implemented two spatial interpolation methods, capable of running in real-time with low latency on Bela. In addition, I empirically evaluate the rendering quality and CPU usage of two different HRTF representations for interpolation and show that raw impulse responses provide a more natural simulation than a combination of minimum-phase responses and time delays for moving sound sources. At the same time, the latter is more computationally efficient.

1 Introduction

The HRTFs describe the frequency response of the travelling path from the sound source to human ears in every direction. Combining it with real-time filtering, it can render binaural sound to simulate virtual acoustic environments, which can be used in games, virtual reality (VR) applications, and the metaverse. To provide immersive experiences for the end users, filtering the HRTFs with low latency and high precision is required.

HRTFs are continuous in 3D space. Continuous representation is feasible on simple objects, e.g., a rigid sphere model, but not for human heads with complex shapes. The common practice is to measure HRTFs in an anechoic chamber (or simulate it using the boundary element method (Ziegelwanger, Kreuzer, and Majdak 2015)) on discrete spatial points and use interpolation methods to fill the gap between the discrete points, assuming the spatial resolution is dense enough. Methods like Spherical Harmonics (SHs) transform (Arbel et al. 2021; Pörschmann, Johannes M. Arend, and Brinkmann 2019; Ben-Hur et al. 2019; Ben-Hur et al. 2021) or Principle Component Analysis (Wang, Yin, and Chen 2009) achieve high quality for interpolation but are not feasible to run in real-time due to high computational complexities. Simpler techniques, such as nearest neighbour and linear interpolation, are often used instead, and the problem becomes retrieving suitable points for interpolation. The worst complexity of finding the nearest point $\mathbf{p}$ to the query $\mathbf{q}$ is $O(n)$ where n is the number of HRTFs, which can be a performance bottleneck when the HRTFs have a dense spatial grid. It is possible to reduce the complexity to $O(1)$ if the sampled points are located uniformly on a selected coordinate system, i.e., equal degree spacing in azimuth and colatitude. However, the measurement grid is varied from database to database, and a more general retrieval method is needed.

In addition, the representation used for interpolation is important. If the representation varies more slowly across the dimensions we want to interpolate, the interpolation introduces less aliasing. It has been shown that the required order of SHs to represent the magnitude and phase of HRTFs is significantly less than the raw value of Head Related Impulse Responses (HRIRs), the time-domain representation of HRTFs (Zaar 2011). In other words, the magnitude and phase of HRTFs vary more slowly than the raw HRIRs along the spatial coordinates. It is worth considering disentangling the magnitude and phase components of HRTFs for more accurate interpolation.

In this work, I implemented the nearest neighbour and linear interpolation methods (Gamper 2013) on Bela (McPherson 2017) to retrieve HRTFs based on the user-provided direction. A KD tree is built based on the Cartesian coordinates of the HRTFs normalised to a unit sphere S^2 during the initialisation of the application. Thus, the complexity of retrieving the nearest point is $O(\log n)$ on average. After interpolation, the HRIR is further upsampled to audio resolution and filters the incoming audio sample by sample. Moreover, I empirically evaluated the interpolation quality of two representations on two different HRTFs. One is the raw HRIRs; the other consists of the minimum-phase HRIRs and Interaural Time Delays (ITDs). The results show that interpolating on raw HRIRs introduces fewer perceptual artefacts than the other, while minimum-phase and ITD representation requires fewer computational resources than raw HRIRs.

2 Application Design

I made a simple GUI interface with three sliders, corresponding to the Elevation $\theta \in [-\frac{\pi}{3}, \frac{\pi}{2}]$ (a large portion of HRTFs databases do not provide measurements for θ around $-\frac{\pi}{2}$), Azimuth $\phi \in [-\pi, \pi]$, and radius $r \in [0.3, 2]$ (in meters) of the sound source with respect to the listener. The user plugs the output of their sound device into the audio input port on Bela, uses these parameters to control the position they want to place the audio source, and listens to the rendered audio over headphones.

2.1 Simulate the distance with r

The energy of the sound is proportional to the squared inverse of its distance. Given the measurement radius r_m of HRTFs, we can simulate the distance effect by multiplying the signal with scaler $\frac{r_m}{r}$. The approximation is close if r is not significantly different from r_m. Better quality can be achieved if the HRTFs provided measurements in multiple radii, but these databases are rare.

2.2 Simulate the direction with θ and ϕ

This section explains how to retrieve the HRTF at the query position $\mathbf{q}$, converted from the slider values as $[\sin\phi\cos\theta, \cos\phi\cos\theta, \sin\theta] \in S^2$, given a discrete set of HRTFs. The HRIRs is denoted as $\mathbf{H} = [\mathbf{h}_1, \ldots, \mathbf{h}_n] \in \mathbb{R}^{m\times n}$, where m is the length of the impulse responses. I omitted the notation of left and right ears for simplicity. The coordinates of the HRTFs sampled points are denoted as $\mathbf{P} = [\mathbf{p}_1, \ldots, \mathbf{p}_n] \in \mathbb{R}^{3\times n}, \mathbf{p}_k \in S^2$. I used an open-source implementation of the KD tree [1].

2.2.1 Nearest Neighbour Method

This is done by building a KD tree with $\mathbf{P}$ and running the built-in nearest neighbour function provided by the open-source implementation with the query $\mathbf{q}$ during test time. This operation has a $O(\log n)$ complexity. I denote the resulting HRIR as $\mathbf{h}_{nn}$, the closest HRIR to the query point.

[1] https://github.com/crvs/KDTree

2.2.2 Triangular Linear Interpolation Method

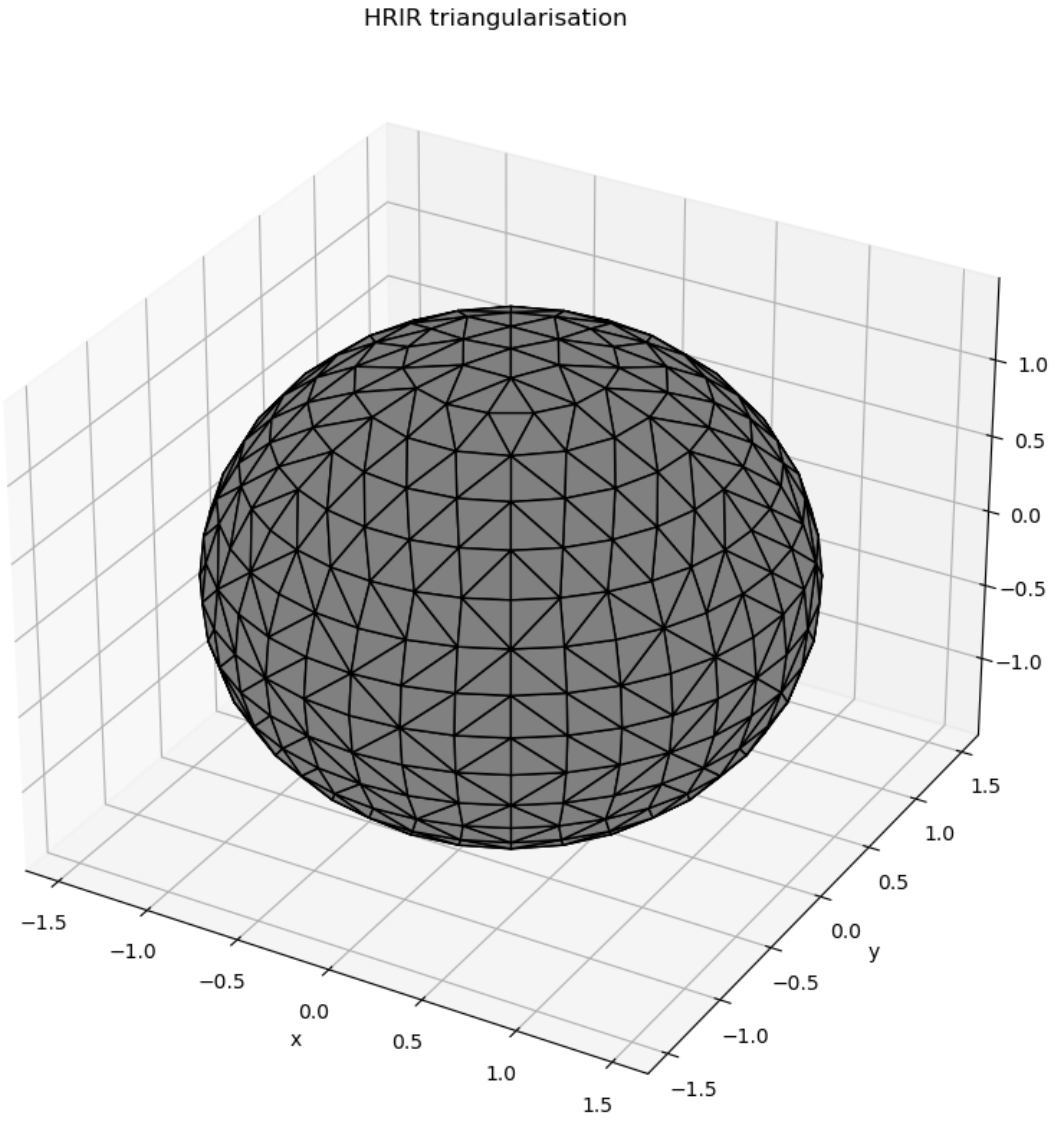

Figure 1: The triangularisation surfaces on the HUTUBS HRTF measurement points.

I follow the method proposed in (Gamper 2013) except for the HRTF subset selection part. A convex hull algorithm is used (specifically, `scipy.spatial.ConvexHull`) to generate a set of simplices $\mathbf{S} \in \mathbb{Z}^{+3\times L}$ where L is the number of simplices. Each simplex consists of indexes of three points in $\mathbf{P}$, representing one of the faces of the volume enclosed by the discrete points (see Fig. 1). The query $\mathbf{q}$ lies on the curved triangle of the simplex $[s_{1,q}, s_{2,q}, s_{3,q}]$ is a weighted sum of the three points:

$$\mathbf{q} = g_1\mathbf{p}_{s_{1,q}} + g_3\mathbf{p}_{s_{2,q}} + g_3\mathbf{p}_{s_{3,q}}. \tag{1}$$

The weights $\mathbf{g} = [g_1, g_2, g_3]$ can also be used as interpolation weights, resulting in

$$\mathbf{h}_{tri} = g_1\mathbf{h}_{s_{1,q}} + g_3\mathbf{h}_{s_{2,q}} + g_3\mathbf{h}_{s_{3,q}}. \tag{2}$$

The weights can be obtained via

$$\mathbf{g} = \mathbf{W}^{-1}\mathbf{q}, \tag{3}$$

where $\mathbf{W} = [\mathbf{p}_{s_{1,q}}, \mathbf{p}_{s_{2,q}}, \mathbf{p}_{s_{3,q}}]$.

The simplex that can be used for interpolation has all its weights positive $\mathbf{g} \in \mathbb{R}^+$. To find the simplex that contains $\mathbf{q}$, instead of directly iterating through the simplices

and checking the above condition, Gamper et al. (Gamper 2013) proposed to first sort the simplices based on their distance to $\mathbf{q}$ and then checking the condition starting from the closest simplex. The sorting algorithm complexity is usually $O(n \log n)$. I propose leveraging the KD tree again and using the k-nearest neighbours (KNN) algorithm to select a subset of simplices. The KNN algorithm can reduce the complexity further to $O(k \log n)$, where k is the number of neighbours and $k \leq n$.

The KD tree is built based on the centres of the simplices. Firstly, for each simplex $\mathbf{s}_k$, I calculated the centroid of the triangle:

$$\hat{\mathbf{c}}_k = \frac{\mathbf{p}_{s_{k,1}} + \mathbf{p}_{s_{k,2}} + \mathbf{p}_{s_{k,3}}}{3}. \tag{4}$$

Then, I normalised the centroids so they lie on the unit sphere S^2 and used it as the centre vectors for simplices:

$$\mathbf{c}_k = \frac{\hat{\mathbf{c}}_k}{||\hat{\mathbf{c}}_k||}. \tag{5}$$

I implemented KNN by extending the open-source implementation because the package doesn't provide KNN but only the nearest neighbour search and neighbours inside a given distance. After finding k nearest simplices, the algorithm iterates through the simplices and stops after finding a simplex with positive weights; if no simplex matches the condition, an error is printed to the Bela console.

3 Alternative Representation: Minimum-Phase and ITD

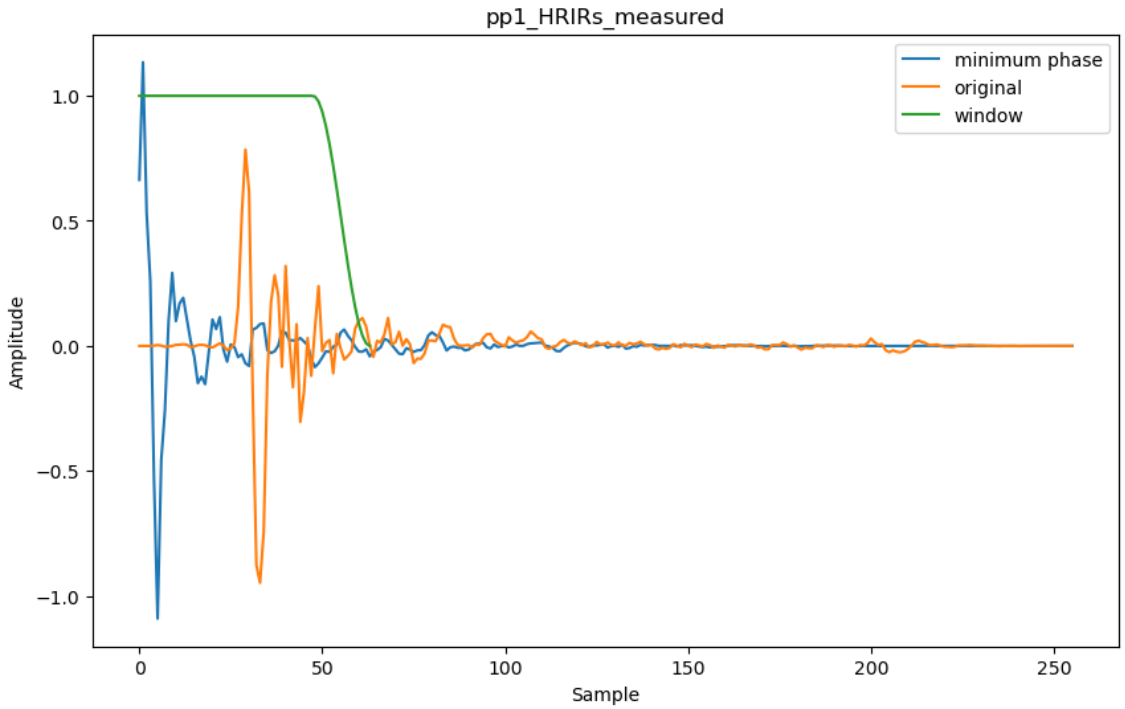

Figure 2: One of the left-ear impulse response from subject `pp1` in HUTUBS, its minimum-phase response, and the truncation window I used in the experiment.

In addition to using HRIRs directly for interpolation, I propose an alternative representation that could possibly reduce interpolation error: minimum-phase HRIRs and ITDs. It is known that HRIR can be roughly decomposed into its minimum-phase response with a time delay τ (Kulkarni, Isabelle, and Colburn 1999):

$$h[t] \approx h_{min}[t] * \delta[t - \tau]. \tag{6}$$

This representation is smoother in the spatial dimension because, in the minimum-phase HRIRs, the waveforms are peak-aligned, and the time delays mostly reflect the distance from the source to the ears, which vary very slowly. The other benefit of minimum-phase HRIRs is the energies are located more at the beginning of the impulse, so we can truncate the HRIRs to a shorter length to save computation resources on convolution.

The minimum-phase HRTFs were calculated using Hilbert transform. After truncating the minimum-phase HRIRs to length M, a half-size Hanning window is applied to the last $\frac{M}{4}$ samples to smooth the response. The window is depicted in Fig. 2. For the time delays, I used the ITD, which is the difference between the delays of two ears. The ITDs were calculated using inter-aural cross-correlation (Andreopoulou and Katz 2017). The same interpolation method in section 2 is used to retrieve minimum-phase HRIR and ITD in samples. If ITD > 0, the delay is added to the right ear HRIR by zero-padding, and vice versa. Linear interpolation is applied on the delayed minimum-phase HRIR to account for fractional delays.

4 Implementation Details

The HRIR retrieval task is running in a separate thread. Each time a number of `hop_size` samples has passed, one such task is created, and the program collects the result in the next cycle of `hop_size`, which is illustrated in Figure 3. The sample-level interpolation is achieved by linearly interpolating the current HRIR feature and the previous one using the position in the current cycle of `hop_size`, so the total latency is > 2 `hop_size`. A circular buffer is used to store the past m samples for convolution. Some data preprocessing, including converting `.sofa` files to `.wav` format, calculating the convex hull and inverse matrices, were done on a jupyter notebook running on a separate machine.

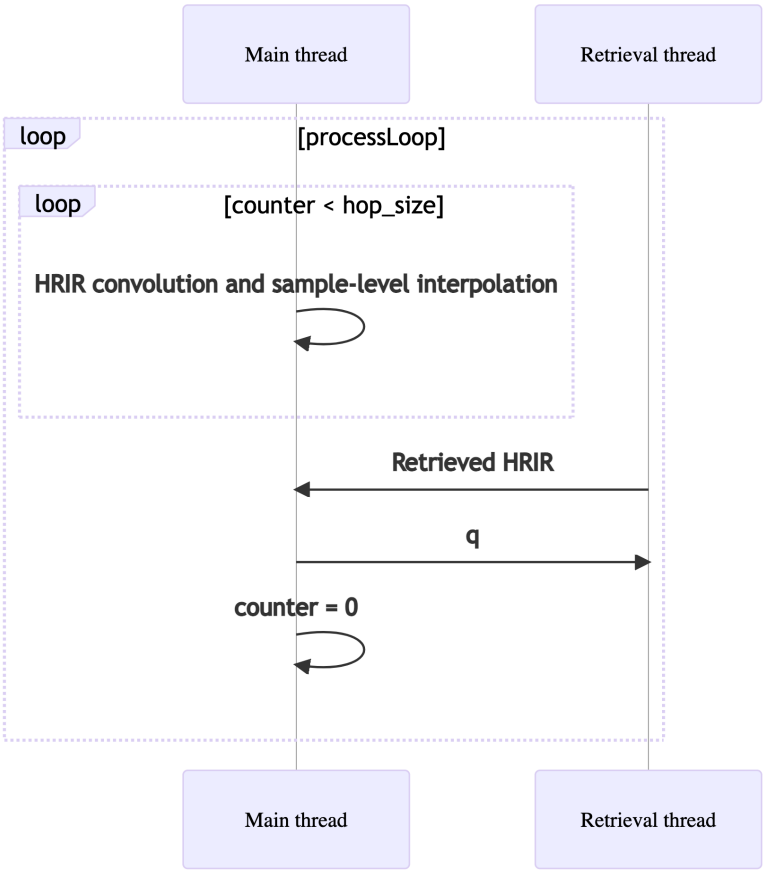

Figure 3: The sequence diagram between the two threads.

HRTF	n	L	Representation	m	Interpolation	CPU (%)
KU100	2702	5400	Raw HRIRs	117	Nearest	≈ 69
					Linear	≈ 69
			Min. + ITDs	64	Nearest	≈ 51
					Linear	≈ 52
HUTUBS	440	876	Raw HRIRs	256	Nearest	N.A.
					Linear	N.A.
			Min. + ITDs	64	Nearest	≈ 52
					Linear	≈ 51

Table 1: CPU usage comparison using the selected HRTF databases with different configurations. The numbers were directly copied from the Bela IDE. N.A. means Bela hits its computational limit and cannot report the number.

5 Evaluations and Discussions

The evaluations aim to test the efficiency of the proposed application and the smoothness of rendering moving sound sources. I tested the program on two HRTFs. One is the KU100 dummy head measured by Johannes M Arend, Neidhardt, and Pörschmann 2016, and the other is human subject `pp1` from the HUTUBS database (Brinkmann et al. 2019). All HRTFs were re-sampled to 44.1 kHz. Other details of the HRTFs are listed in Table 1. I set `hop_size` to 32 and the number of neighbours k to 24, the minimum value that the program can always find the target simplex on the two HRTFs. I truncated all the minimum-phase HRIRs to 64 samples. All experiments were run with a block size of 16 on Bela.

The CPU usage numbers reported in Table 1 tells us a few things. The usage difference is not noticeable between dense and sparse HRTFs, implying that the proposed HRIR retrieval methods' computational load is low. It is so low that I could run the retrieval task inside each processing block directly instead of putting it in a separate thread. The `hop_size` I chose (32) is also close to the block size on Bela (16). The usage is hugely dominant by the computation of convolution. Convoluting HRIRs with 256 samples exceeds the Bela computational limit. There is no noticeable difference between the nearest neighbour and linear interpolation regarding the computational load. Minimum-phase and ITD representation reduces usage by around 17% to 18% usage, proving its efficiency.

I tested the smoothness by listening to various audio samples rendered by the proposed program on AKG K271 MKII headphones. When using the raw HRIR representation, I could barely notice any difference between the two interpolation methods. The nearest neighbour method is smooth enough not to introduce any artefacts, probably due to the high spatial resolution of KU100 HRTFs. For the minimum-phase and ITDs representation, I can hear glitches when sweeping the sliders quickly, regardless of the interpolation methods. I found the problem is due to the interpolation of fractional delays. The artefacts still exist after I replaced the linear interpolation with parabolic interpolation, which has a smoother interpolation curve than the former. It is possible this issue can be solved by a more advanced interpolation method, but it requires more computational resources.

6 Conclusions

In this report, I implemented a real-time HRTF-based renderer that can transform any sound input to the Bela board to any direction with a simple web GUI interface. The application is able to run HRTFs with up to 2702 points and 117 samples. The empirical results from the experiment show that optimising convolution is more important than optimising the HRIR retrieval method when efficiency is the priority. Although the proposed minimum-phase and ITD representation successfully reduce the load, it is prone to introduce artefacts for fast-moving sound sources.

References

Andreopoulou, Areti and Brian F. G. Katz (Aug. 2017). "Identification of perceptually relevant methods of inter-aural time difference estimation". In: *The Journal of the Acoustical Society of America* 142.2, pp. 588–598. (Visited on 05/02/2023).

Arbel, Lior et al. (June 2021). "Applied Methods for Sparse Sampling of Head-Related Transfer Functions". In: *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 446–450.

Arend, Johannes M, Annika Neidhardt, and Christoph Pörschmann (2016). "Measurement and Perceptual Evaluation of a Spherical Near-Field HRTF Set". In: *29th Tonmeistertagung*.

Ben-Hur, Zamir et al. (Dec. 2019). "Efficient Representation and Sparse Sampling of Head-Related Transfer Functions Using Phase-Correction Based on Ear Alignment". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 27.12, pp. 2249–2262. (Visited on 10/17/2021).

— (2021). "Binaural Reproduction Based on Bilateral Ambisonics and Ear-Aligned HRTFs". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 29, pp. 901–913.

Brinkmann, Fabian et al. (2019). "The HUTUBS head-related transfer function (HRTF) database". In: URL: `https://depositonce.tu-berlin.de/handle/11303/9429` (visited on 05/07/2023).

Gamper, Hannes (2013). "Selection and interpolation of head-related transfer functions for rendering moving virtual sound sources". In: *Proceeding of the 16th International Conference on Digital Audio Effects*.

Kulkarni, Abhijit, S. K. Isabelle, and H. S. Colburn (May 1999). "Sensitivity of human subjects to head-related transfer-function phase spectra". In: *The Journal of the Acoustical Society of America* 105.5, pp. 2821–2840.

McPherson, Andrew (May 2017). "Bela: An embedded platform for low-latency feedback control of sound". In: *The Journal of the Acoustical Society of America* 141.5_Supplement, p. 3618.

Pörschmann, Christoph, Johannes M. Arend, and Fabian Brinkmann (2019). "Directional Equalization of Sparse Head-Related Transfer Function Sets for Spatial Upsampling". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 27.6, pp. 1060–1071.

Wang, Lin, Fuliang Yin, and Zhe Chen (2009). "Head-related transfer function interpolation through multivariate polynomial fitting of principal component weights". In: *Acoustical Science and Technology* 30.6, pp. 395–403. (Visited on 05/08/2023).

Zaar, Johannes (2011). "Phase unwrapping for spherical interpolation of headrelated transfer functions". In: *M. thesis, IEM, Univ. of Music and Performing Arts Graz*.

Ziegelwanger, Harald, Wolfgang Kreuzer, and Piotr Majdak (2015). "Mesh2hrtf: Open-source software package for the numerical calculation of head-related transfer functions". In: *22nd International Congress on Sound and Vibration*.