F. Brinkmann, W. Kreuzer, J. Thomsen, S. Dombrovskis, K. Pollack, S. Weinzierl and P. Majdak, "Recent Advances in an Open Software for Numerical HRTF Calculation"
J. Audio Eng. Soc., vol. XX, no. X, pp. 1–13, (2023 X).
DOI: https://doi.org/10.17743/jaes.2022.0078

Recent Advances in an Open Software for Numerical HRTF Calculation

FABIAN BRINKMANN,[1] *AES Member*, **WOLFGANG KREUZER,**[2]
(fabian.brinkmann@tu-berlin.de) (wolfgang.kreuzer@oeaw.ac.at)
JEFFREY THOMSEN,[1] **SERGEJS DOMBROVSKIS,**[3] *AES Member*,
AQ1
KATHARINA POLLACK,[2] *AES Member*, **STEFAN WEINZIERL,**[1] *AES Member* **AND**
PIOTR MAJDAK,[2] *AES Member*

[1]*Audio Communication Group, Technical University of Berlin, Germany*
[2]*Acoustics Research Institute, Austrian Academy of Sciences, Vienna, Austria*
[3]*China Euro Vehicle Technology AB, Gothenburg, Sweden*

Mesh2HRTF 1.x is an open-source and fully scriptable end-to-end pipeline for the numerical calculation of head-related transfer functions (HRTFs). The calculations are based on 3D meshes of listener's body parts such as the head, pinna, and torso. The numerical core of Mesh2HRTF is written in C++ and employs the boundary-element method for solving the Helmholtz equation. It is accelerated by a multilevel fast multipole method and can easily be parallelized to further speed up the computations. The recently refactored framework of Mesh2HRTF 1.x contains tools for preparing the meshes as well as specific post-processing and inspection of the calculated HRTFs. The resulting HRTFs are saved in the spatially oriented format for acoustics being directly applicable in virtual and augmented reality applications and psychoacoustic research. The Mesh2HRTF 1.x code is automatically tested to assure high quality and reliability. A comprehensive online documentation enables easy access for users without in-depth knowledge of acoustic simulations.

0 INTRODUCTION

Head-related transfer functions (HRTFs) capture the sound transmission from a free-field point source to the left and right (blocked) ear canal entrances [1]. They contain interaural and monaural cues that are exploited by the human auditory system to derive an auditory representation of the surrounding 3D space [2, 3]. In virtual and mixed-reality scenarios, they form the basis for rendering 3D sound
AQ2
through a filtering of anechoic audio signals with HRTFs.

Because interaural and monaural cues arise from the individual shape of the listener's ears, head, and torso, HRTFs differ substantially between listeners. As a consequence, individual HRTFs are required for authentic 3D audio rendering, and using non-individual HRTFs potentially reduces sound-localization performance (especially for source elevation, [4–7]) and introduces an unnatural coloration among other artifacts [8]. These artifacts are at least partially mitigated in interactive and audio-visual virtual and mixed reality [9, Chapter 5.4.1]. Different individualization approaches are available and two promising possibilities are to calculate the HRTFs from 3D meshes of the head or to estimate the HRTF based pictures or anthropometric features of listeners ears and heads using deep learning [10, 11].

Various applications and numerical methods are available to calculate acoustical transfer functions by solving the 3D wave equation or the Helmholtz equation [12, 13]. Commercial solutions include COMSOL [14], FastBEMAcoustics [15], and Ansys [16], and open-source software packages include AcouSTO [17], FreeFEM [18], Bempp [19], OpenBEM [20, 21], and ParallelFDTD [22, 23] among others. Commercial solutions are not accessible to everyone, and open-source solutions often have restrictions regarding their performance, richness of features, or usability.

In this article, Mesh2HRTF 1.x—an open-source pipeline for the numerical calculation of HRTFs—is introduced. The pipeline, with the overview shown in Fig. 1,
AQ3

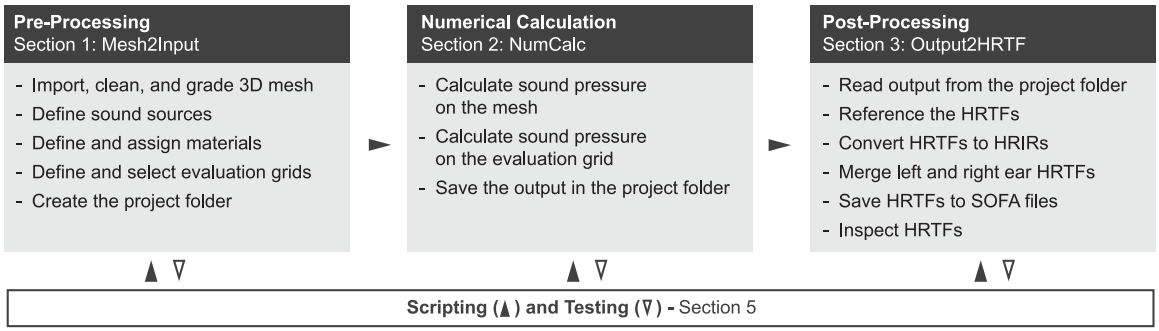

Fig. 1. Overview of the Mesh2HRTF pipeline for calculating HRTFs. Left: Mesh2Input, based on a mesh, it creates input for the numerical calculations. Center: NumCalc, processes the input to output. Right: Output2HRTF, processes the numerical output to HRTFs. Each section reflects a corresponding directory in the Mesh2HRTF package.

includes fully scriptable tools for preparing 3D meshes for the numerical calculations, performing the calculations, as well as processing and inspecting the results. The final HRTFs are provided in the widely supported SOFA files [24, 25], to be used in extended-reality and other binauralized audio applications.

Mesh2HRTF was originally developed as versions 0.x at the Acoustics Research Institute of the Austrian Academy of Sciences (ÖAW) with a focus on using existing open-source software to build a flexible yet convenient pipeline [26]. It has been recently refactored as Mesh2HRTF 1.x and is now maintained by the Audio Communication Group of the Technical University of Berlin and the ÖAW. In the following, the Mesh2HRTF 1.x pipeline is detailed using the example of separately calculating left and right ear HRTFs based on a 3D head mesh. The authors aim at providing a good understanding of Mesh2HRTF as a whole and provide a good basis for novice users before getting more involved with the online tutorials [27].

Note that Mesh2HRTF 1.x was developed for two environments, MATLAB and Python. Although the degree of functionality for these individual environments depends on the state of development, the names of the functions across these environments are the same. In this article, these functions are thus referred to without the environment-specific extension.

1 MESH2INPUT: MESH PRE-PROCESSING AND INPUT PREPARATION

Before HRTFs can be calculated, the input data for the numerical calculation needs to be prepared. This includes the 3D mesh of the listener, the evaluation grid, i.e., the positions at which the HRTF will later be calculated, and optional material files to specify acoustic boundary conditions that can be applied to parts of the 3D mesh. These preparations are supported by the Mesh2HRTF functionality stored in the directory `Mesh2Input`.

1.1 The Mesh

The starting point of the Mesh2HRTF chain is a 3D mesh of the scatterer's surface, i.e., a discrete description of the individuals head (and torso) geometry using triangles or plane quadrilaterals. The mesh must meet a number of requirements to be suitable for the numerical calculations. It must be water tight (i.e., a closed mesh without holes), have neither duplicate vertices, nor intersecting faces, nor faces with an area of zero, and have all normals facing outward, i.e., the vertices of each element must be listed in counter-clockwise order. Many free software tools such as Blender [28], MeshLab [29, 30], and Meshmixer [31] offer functionality to check and comply with these requirements. Violation of these requirements can cause severe errors in the calculated HRTFs and it is up to the user to check the results for plausibility.

For a proper numerical accuracy, the boundary-element method (BEM) requires as regular triangles or plane quadrilaterals as possible, as similar edge length as possible, and edge lengths to be at least six times smaller than the simulated wave length [32]. For example, maximum edge lengths of approximately 2.6 mm are required for simulating HRTFs up to 22 kHz (assuming a speed of sound of 343 m/s). However, the pinna geometry must be sampled with an error of less than 1 mm, which requires edge lengths in the range of $1-2$ mm depending on the local curvature of the mesh [33, 34, 26]. Numerical experiments have shown that the accuracy of the calculations may suffer if very fine meshes are used at low frequencies [35], but edge lengths in the range of $1-2$ mm work well across the frequency range from 100 Hz$-$20 kHz in the authors' experience.

Additionally, thin structures pose a challenge for the numerical calculation and might require specialized methods [36, 37]. They occur at the folds of the outer ear, e.g., at the helix where there were observed thicknesses in the range of $2-3$ mm in 3D meshes, for which the HRTF could well be calculated with Mesh2HRTF. However, even thinner structures might occur due to creases of cloth that can cause numerical issues. Such cases should be resolved prior to the HRTF calculation, for instance by manually deleting the creases and filling the resulting holes, or by applying the mesh grading described below.

The errors induced by violations of these requirements depend on the position of the problematic faces as well as on the severity and number of violations. The closer the faces violating the requirements are located to the ear for which the HRTF is calculated, the more they will affect the

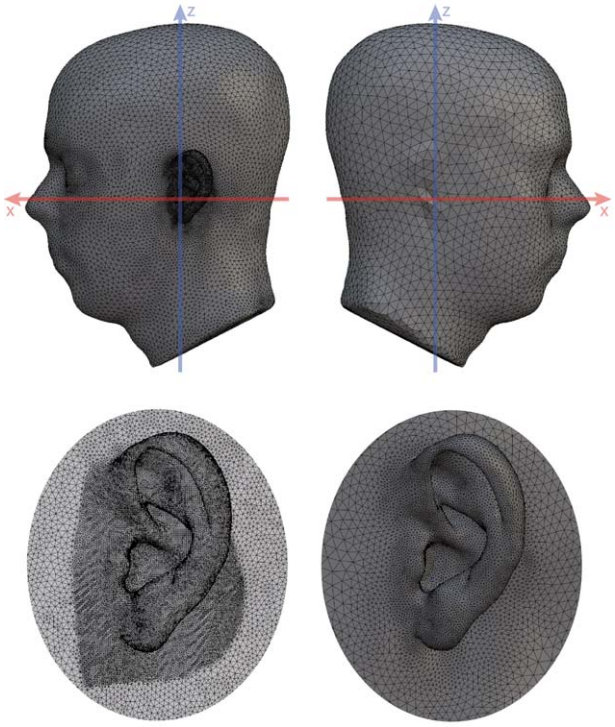

Fig. 2. Illustration of meshgrading. Top: Example of a graded mesh showing a high-resolution left ear (left panel) and a low-resolution right ear (right panel), used to calculate HRTFs of the left ear. Note that the mesh for calculations of the right-ear HRTFs (not shown here) has a high-resolution right ear and a low-resolution left ear. Bottom: Left ear before (left) and after (right) mesh grading. The original mesh is from the HUTUBS HRTF database [42, 43, subject 22].

results. Unfortunately, meshes resulting from various scanning methods often violate some of these requirements if not post-processed properly. Specifically, scanned meshes may consist of too small and irregular triangles with edge lengths well below 2 mm, which slow down the numerical calculation and increase the memory requirements significantly. Additionally, holes may be present in the mesh, especially at the pinna regions, in which the geometry is not easy to be captured because of its crevices and folds. One established solution to meet the BEM specific requirements is to apply mesh-filling methods and to remesh the scanned geometry targeting the required edge length.

To further reduce the computation time and required memory, the requirement on the maximum edge lengths can be relaxed by increasing the size of elements that are far away from the ear for which the HRTF is simulated—a process termed *mesh grading*. Two types of mesh grading algorithms are provided as part of Mesh2HRTF, a distance-based grading [38] using the remeshing algorithm from OpenFlipper [39] as a basis, and a hybrid grading [40] based on the remeshing algorithm from the Polygon Mesh Processing library [41]. The grading algorithms are stored in "`Meshes/GradingDistanceBased`" and in "`Meshes/GradingHybrid`," respectively.

An example for a mesh processed with the hybrid grading algorithm [40] is shown in Fig. 2. The grading reduced the faces in the mesh from approximately 190,000 to 32,000. In comparison to using a high-resolution reference mesh, the grading speeds up the HRTF calculation by about 60% on average while keeping the average induced error below 6% (0.5 dB). Mesh grading is an example of violating the above mentioned requirements on purpose, on elements far from the ear for which the HRTF is calculated, for which the effect of this violation is negligible [38, 40]. Additionally, mesh grading often resolves problems with irregular faces.

A second specific requirement is the position and orientation of the mesh within the right-handed Cartesian coordinate system used by Mesh2HRTF (cf. Fig. 2). The head must be placed on the interaural axis, i.e., the line connecting the centers of the ear canal entrances must coincide with the y axis. Further, the head must be viewing in the positive x-direction. Consequently, the y-coordinate of the left ear-canal entrance y_l must be positive and the y coordinate of the right ear-canal entrance y_r must be approximately $-y_\mathrm{l}$. The interaural center, i.e. the midpoint between the left and right ear canal entrances thus must coincide with the origin of coordinates. The positioning and rotation of the mesh is supported by the functionality stored in "`Meshes/CenterHead`."

1.2 Evaluation Grids

The positions in space at which the HRTFs are calculated are defined in evaluation grids, which conceptually equal the source positions in acoustic HRTF measurements. Mesh2HRTF defines evaluation grids in the directory "`EvaluationGrids/Data`" by means of text files containing the nodes and elements of the grid. Whereas some predefined evaluation grids are provided with the package, other grids can easily be read and written using the methods "`EvaluationGrids/read_evaluation_grid`" and "`EvaluationGrids/write_evaluation_grid`." An easy possibility to generate a large variety of evaluation grids is to use pyfar[1], which contains methods for generating spherical equi-angular, Gauss, Lebedev, and Fliege grids among others.

1.3 Materials

Materials are used to define acoustic boundary conditions that enforce a specific acoustic property on one or more elements of the mesh. Mesh2HRTF stores the materials in the directory "`Materials/Data`" and can handle three types of boundary conditions:

Admittance: Sets the specific acoustic admittance $\frac{1}{\rho c}\frac{v}{p}$, in which ρ denotes the density of air in kg/m^3 and c the speed of sound in meters per second. The normalization by ρc makes it possible to use a single boundary definition for simulations with different speeds of sound and densities of air. Admittance boundary conditions can be used to define the frequency-dependent

[1] https://pyfar.org.

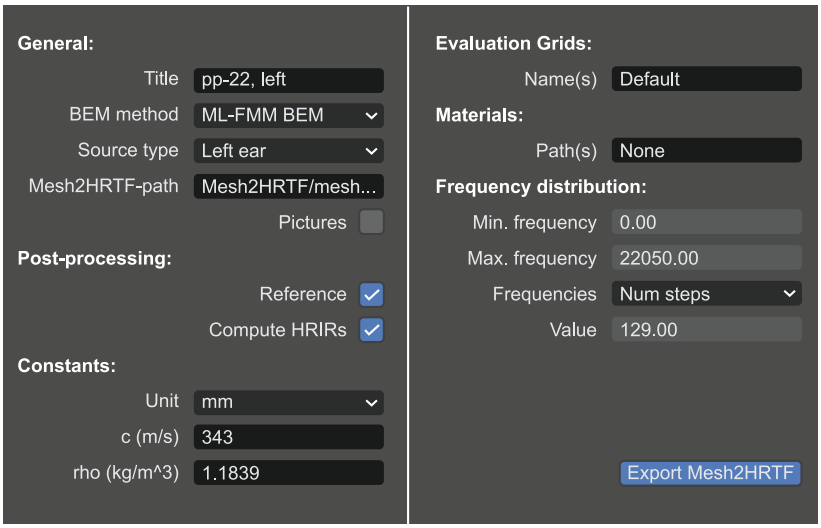

Fig. 3. Example of settings of the Blender export add-on for preparing a Mesh2HRTF project for the left ear using a volume velocity source. The settings for the right ear are identical except for the parameters *Source type* and *Title*.

sound-absorbing material properties, for example, to account for the effect of clothes or hair.

Velocity: Enforces the Neumann boundary condition, i.e., sets the particle velocity v in meters per second in the direction of the normal vector to the surface on an element. A velocity of $v = 0$ is the default setting in Mesh2HRTF and specifies a rigid, i.e., sound hard surface. Note that non-zero v values can be used to define a volume velocity source.

Pressure: Enforces the Dirichlet boundary condition, i.e., sets the pressure p in Pa on an element of the boundary. For example, a pressure of $p = 0$ can be specified to model a sound-soft scattering element. Although this condition is not relevant for HRTF simulations, it can be useful in the evaluation of the performance of Mesh2HRTF when compared against the analytical solution of a sound-soft sphere.

For more detailed information on the definition of the boundary conditions, see [35]. Additional boundary conditions are defined by creating material files that can be generated with the function "`write_material`."

1.4 Preparation in Blender

Before the HRTFs can be calculated, parameters of the Mesh2HRTF project need to be defined. This can be done in the open-source 3D computer-graphics software Blender [28], which can import meshes in most common file formats. Mesh2HRTF provides a Blender export add-on, which is contained in "`mesh2input.py`." With that add-on, the parameters of the calculations can be set up and the Blender scene can be exported as a Mesh2HRTF project, ready for the numerical calculations of the HRTFs that are done by NumCalc (the numerical kernel of Mesh2HRTF performing the BEM calculations, cf. SEC. 2).

The available parameters are shown in Fig. 3. Although all parameters are fully described in the online documentation [27], the key parameters are the selection of the acoustic source, the boundary conditions, the post-processing steps (cf. SEC. 3), the names of the evaluation grids and material files, as well as the definition of the frequencies at which the HRTFs will be calculated.

The boundary conditions are defined by assigning Blender materials to selected elements of the mesh. For each assigned material, a material file (cf. SEC. 1.3) with the same name must be available (with the exception of the materials *Skin*, *Left ear*, and *Right ear*), and the folder containing that file must be given in the export interface shown in Fig. 3.

The acoustic source can be set up by selecting one of the available types to calculate HRTFs: Point sources and volume velocity sources. HRTF calculations usually exploit the concept of acoustic reciprocity to speed up calculations by swapping the position of receivers and sources. This is done by placing a point source or volume velocity source at the (blocked) ear canal entrance (where the microphone would be during the acoustic measurements) and receivers in the surrounding space (where the sources would be in acoustic measurements). Volume velocity sources are defined by creating and assigning the materials *Left ear* and/or *Right ear* to one or more mesh elements at the corresponding ear canal entrance [44]. This will assign a velocity boundary condition on the marked elements. Point sources are defined by placing a point light object in the Blender scene close to but not on the surface of the blocked ear canal. The authors recommend a distance in the range of $1-1.5$ times the average edge length of the element closest to the point source. Note that placing the source directly on the face surface may cause numerical issues originating from the singularities of the Green's functions in the calculations.

If a point source is too close to the mesh surface, there was too much high-frequency energy experienced at the contralateral ear. As a consequence, this causes a decrease in interaural level difference (ILD) for lateral source positions. Although point sources can be useful to compare

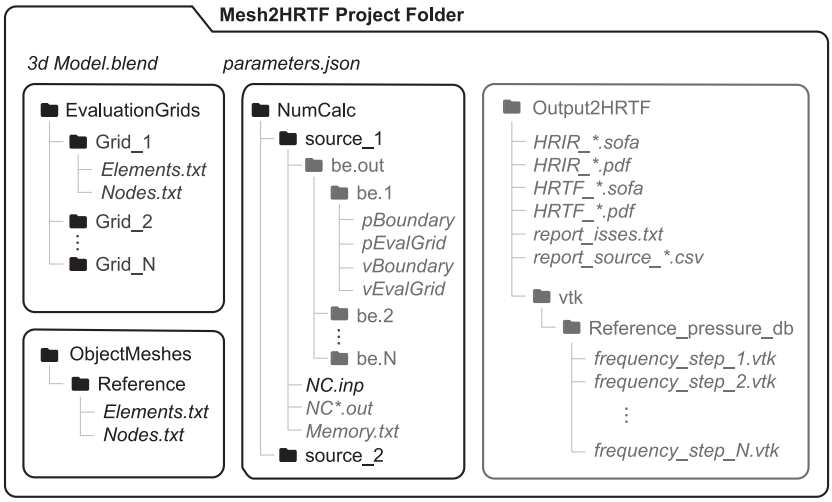

Fig. 4. Example of a Mesh2HRTF project folder. Folders are given in normal font and files in italic. Data generated by the Blender export add-on are indicated by black symbols and font. Data generated during the numerical calculations and post-processing is denoted by gray symbols and font. The folder source_2 only exists, if option *Both ears* is selected in the field *Source type* (cf. Fig. 3).

results against analytical references or across different numerical solvers, the authors recommend the volume velocity sources placed at the blocked ear channel as they do not have such issues [44, 45].

One or more evaluation grids can be defined by entering their names in the *Evaluation grids* setting. Predefined evaluation grids can be selected by their names, custom evaluation grids must be provided by the full paths to the corresponding files. Custom material files can be included by providing the paths of the containing folder. The frequencies for the calculations are defined by the minimum and maximum frequency and either the number of frequency steps or the spacing between successive frequencies. The example shown in Fig. 3 will calculate HRTFs for 129 frequencies between 0 and 22,050 Hz.[2]

Once the parameters are set up, a click on ”Export Mesh2HRTF” creates a Mesh2HRTF project folder containing the data for the numerical calculation. An example of such a project folder is shown in Fig. 4. The file “`3d Model.blend`” is a copy of the Blender project that is stored for convenience and “`parameters.json`” contains all project settings for documentation and post-processing of the results. The folders “`ObjectMeshes`” and “`EvaluationGrids`” contain the mesh and receiver positions stored as text files. The folder `NumCalc` contains files for the numerical calculation (cf. SEC. 2).

2 NUMCALC: THE NUMERICAL CALCULATION

HRTFs calculations require the computation of the the sound field on the surface of the head and at positions around the head (called evaluation grid in Mesh2HRTF). To that end the Helmoltz equation describing the linear wave propagation in space in the frequency domain is solved numerically using “NumCalc,” the BEM kernel of Mesh2HRTF (see Appendix A and Kreuzer et al. [12, 35] for details).

The BEM has the advantage that only the surface of the head needs to be represented by a mesh—and not the entire space containing the 3D mesh and evaluation grid, as would be required by finite difference time domain (FDTD) approaches and the finite element method (FEM) [46, Parts I and II]. Also, with the BEM, it is guaranteed that the acoustic field decays toward infinity, i.e., that the Sommerfeld radiation condition is fulfilled (cf. [47, SEC. 3.1]). On the other hand, the BEM formulation requires to numerically integrate the singular Green’s function and its derivatives, which may require special methods. Also, in the BEM, the matrix representing the final system of equations is densely populated; thus, a feasible computation is only possible using iterative solvers in combination with the fast multipole method.

In the first step of the BEM calculation, the unknown sound pressure on the head is calculated on the mid points of each element in the mesh, defined as the arithmetic mean of the vertex positions (see Fig. 5 for an example). The pressure is assumed to be constant for each element in the mesh (collocation with constant elements). The fact that constant elements are used is one of the reasons that the sizes of the elements need to reflect the maximum frequency for which the HRTF is simulated.

To speed up the computation and to reduce memory consumption, the multilevel fast multipole method (FMM), [48, 49], can be selected upon project export (cf. Fig. 3). In a nutshell, the FMM is based on a man-in-the-middle principle. The elements of the mesh are grouped into different clusters, and element-to-element interactions are partly replaced by element-to-cluster and cluster-to-cluster interactions, that can be calculated independently. This way, the

[2]In fact, calculations for $f = 0$ Hz are not performed, and the corresponding data is added during the post-processing described in SEC. 3. Thus, only 128 frequencies are calculated beginning with 172.27 Hz.

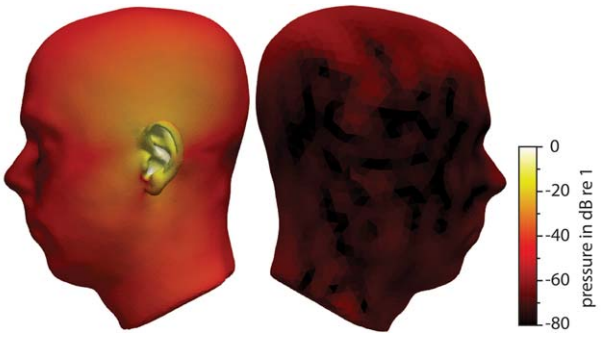

Fig. 5. Example of a left-ear mesh showing the color-coded sound pressure in decibels at 7 kHz. The sound pressure was calculated with NumCalc. The visualization was done with "export_vtk" and "ParaView" (https://www.paraview.org/). The pressure was normalized to 0 dB.

computational complexity can be reduced from $O(N^2)$ to $O(N \log N)$ operations. For more details on applying FMM to calculate HRTFs, see [48, 49, 12].

In the second step of the BEM calculation, the sound pressure is propagated from the mesh to the evaluation grid. Whereas the first step is computationally expensive and time consuming, the second step is relatively cheap and fast for most evaluation grids. This is the reason for applying the reciprocal calculation (cf. SEC. 1.4). Even if HRTFs are calculated for thousands of different directions, i.e., source positions defined by the evaluation grid, the computationally expensive first step has to be done only twice per frequency—once for the left and once for the right ear. The following sections briefly describe the output generated by NumCalc and how NumCalc can be used to calculate the sound pressure on the mesh and at the evaluation grid.

2.1 Starting the Numerical Calculations

The numerical calculations are started by calling NumCalc from the command line. NumCalc requires the "NC.inp" file that is written during the project export and is contained in the folders "source_1" and "source_2" (cf. Fig. 4). NumCalc must thus be executed with these folders being the working directory. The NC.inp file defines all parameters that are required for the BEM calculation. This includes the paths to the text files that define the mesh and evaluation grids, the values of constants such as the speed of sound and the density of the medium, the frequencies for which the HRTFs are calculated, and the definition of the boundary conditions. For a detailed description of the NC.inp file, the interested reader is referred to [35] and the Mesh2HRTF documentation [27].

A simple call of NumCalc calculates the sound pressure and particle velocity on the mesh and the evaluation grid for all frequencies in a single thread. Depending on the available resources (threads, memory, and CPU load), the calculation can be sped up by manually running multiple NumCalc instances in parallel. For example, "NumCalc -istart 1 -iend 16" calculates frequencies 1−16 and each call of NumCalc with different parameters starts an additional, parallel thread. As the results of each frequency step are stored in a separate folder, it is easy to combine results from all threads once they are finished. Note that on its start, NumCalc deletes existing results for the range of frequency steps it is called to calculate (without command line arguments it deletes all previous results).

Additionally, NumCalc can estimate the required memory per frequency step if using the FMM-BEM by calling "NumCalc -estimate_ram." The estimation is based on the non-zero entries in the FMM matrices and is written to the file "Memory.txt" that contains the frequency step number, the frequency in Hertz, and the estimated memory consumption in gigabytes (cf. Fig. 4). Note that the actual memory consumption can vary around ±5% around the estimation. The estimation can be used for an automatic parallelization of NumCalc based on the available resources, significantly accelerating the calculations as described in SEC. 4. For the example shown in Fig. 2, the estimated memory consumption was large for the lowest frequencies (8.6 GB below 1 kHz), intermediate for the highest frequencies (approximately 6 GB for around 20 kHz), and small for the mid-frequencies (2.2 GB for between 1 and 6 kHz).

2.2 Results

The results are stored in the folder "be.out" (cf. Fig. 4). It contains the sub-folders "be.1" to "be.N," with N being the number of frequencies for which the Helmholtz equation was solved (128 in the example used here). In these folders, the pressure and velocity on the mesh and at the nodes of the evaluation grids are stored in the following text files:

- "pEvalgrid" contains the sound pressure at the nodes of the evaluation grids.
- "pBoundary" contains the sound pressure at the midpoints of the mesh elements.
- "vEvalgrid" contains the particle velocity in the x, y, and z direction at the nodes of the evaluation grid.
- "vBoundary" contains the particle velocity at the midpoints of the mesh elements in the direction of the normal vector.

In all cases, the first column in these files always contains a unique number identifying the node of the evaluation grid or the element of the mesh. The following columns contain the real and imaginary parts of the complex valued results in separate columns.

Furthermore, a detailed summary of the calculations is written to the file(s) "NC*.out," the names of which depend on the command line options "istart" and "iend" (cf. SEC. 2.1). The summary contains, e.g., the computation time, warnings, and error messages.

3 OUTPUT2HRTF: POST-PROCESSING

The NumCalc results provide all information to derive the sound pressure at the (blocked) left and right ear canal entrances $p_{\mathrm{l,\,r}}$. If the direct method with an external sound

source away from the head was chosen, $p_{\mathrm{l,\,r}}$ is directly determined as the (mean) sound pressure at selected mesh elements at the left and right ears. If the reciprocal calculation was chosen, $p_{\mathrm{l,\,r}}$ is determined by using the sound pressure at the nodes of the evaluation grid. In the following, the authors describe the post-processing required to derive the HRTFs from $p_{\mathrm{l,\,r}}$. Based on this, the head-related impulse responses (HRIRs), which are the time-domain equivalent of the HRTFs, can be computed. The post-processing steps are started by the function "`output2hrtf`."

3.1 HRTF Referencing

Referencing is the first post-processing step required to remove the frequency response of the source from the calculated pressure values. It is performed only if the option *Reference* was selected in the Blender export add-on (cf. Fig. 3). The referencing is realized by a complex division of $p_{\mathrm{l,\,r}}$ by the reference pressure p_{ref} [1]

$$\mathrm{HRTF}_{\mathrm{l,r}}[\mathbf{x}_s,\, f] = \frac{p_{\mathrm{l,r}}[\mathbf{x}_s,\, f]}{p_{\mathrm{ref}}[\mathbf{x}_s,\, f]}, \tag{1}$$

where f denotes the frequency in Hz, $\mathbf{x}_s = (x, y, z)^\top$ the position of the sound source in m or the position of a node in the evaluation grid in the case of reciprocal calculations. p_{ref} is the pressure at the center of the head (interaural center) at $\mathbf{x}_0 = (0, 0, 0)^\top$ caused by a virtual monopole sound source at $\mathbf{x}_s$.

The definition of the virtual monopole for the calculation of p_{ref} depends on the type of the source set up. For the point source, it is given by

$$p_{\mathrm{ref,\,point}}[\mathbf{x}_s,\, f] = p_0 \frac{\mathrm{e}^{\mathrm{j}k||\mathbf{x}_s-\mathbf{x}_0||}}{4\pi||\mathbf{x}_s-\mathbf{x}_0||}, \tag{2}$$

where $p_0 = 0.1\,\mathrm{Pa}$ is the source amplitude, $k = 2\pi f/c$ the wave number in rad/m, and $||\mathbf{x}_s - \mathbf{x}_0|| = \sqrt{(\mathbf{x}_s-\mathbf{x}_0)^\top(\mathbf{x}_s-\mathbf{x}_0)}$ equals the radius of the source in spherical coordinates or the radius of a node in the evaluation grid in case of reciprocal calculations.

The volume velocity source is defined by [50, Eq. (6.71)]

$$p_{\mathrm{ref,\,velo}} = -\mathrm{j}\rho_0 c k Q_\mathrm{s} \frac{\mathrm{e}^{\mathrm{j}k||\mathbf{x}_s-\mathbf{x}_0||}}{4\pi||\mathbf{x}_s-\mathbf{x}_0||}, \tag{3}$$

where $Q_\mathrm{s} = v_0 A$ is the source strength in m^3/s, with the velocity $v_0 = 0.1$ m/s, and A the area of the velocity source in m^2, i.e., the summed area of the elements assigned to the materials *Left ear* and *Right ear*, respectively (cf. SEC. 1.4). Note that p_0 and v_0 are hard-coded in the Blender export add-on because they are not supposed to be modified by the user.

After the referencing, the source's frequency response is removed and the HRTF only reflects the influence of the head on the incoming (or outgoing) sound field. This also means that the HRTF magnitude response approaches 1 (i.e., 0 dB) for $f \to 0$ because the influence of head decreases when the wave length becomes large in comparison to the size of the head. Fig. 6 a and 6 b shows an example of the HRTF magnitude spectrum before and after the referencing.

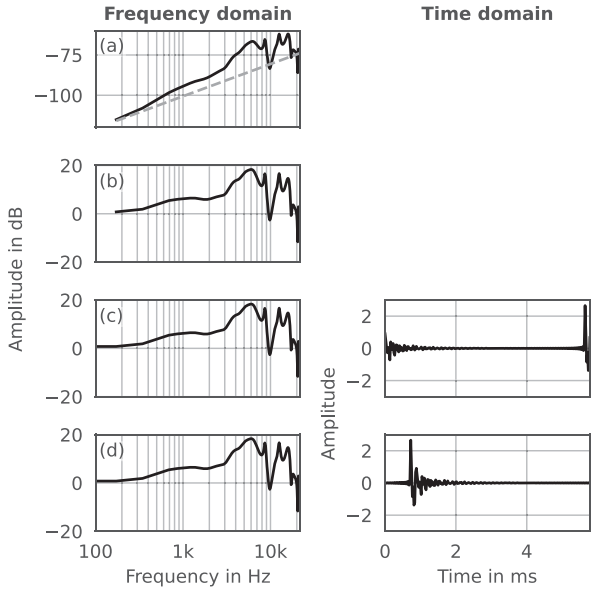

Fig. 6. Illustration of the post-processed HRTF (left panel) and HRIR (right panel) for a left ear HRTF and a source on the horizontal plane with an azimuth angle of 90 degrees. (a) Raw pressure resulting from `NumCalc` (black) and reference pressure according to Eq. (3) (gray). (b) Magnitude spectrum of a referenced HRTF. (c) Magnitude spectrum and time signal after the inverse Fourier transform. (d) Magnitude spectrum and time signal after cyclic shift (see text for more information).

3.2 HRIR Computation

HRIRs are computed if the option *Compute HRIRs* was selected (cf. Fig. 3) and requires referenced HRTFs calculated at equidistant frequencies $f = \{f_0, 2f_0, ..., Nf_0\}$ with $f_0 > 0$ Hz and $N > 0 \in \mathbb{N}$. To prepare the HRTFs for this step, data for $f = 0$ Hz are artificially defined because the BEM does not perform calculations at 0 Hz. The authors use a value of 1 (i.e., 0 dB) at 0 Hz because the head does not have any acoustic effect on the transfer function at this frequency.

Additionally, it is assumed that the sampling rate f_s is given by $2Nf_0$ and thus $f_s/2 = Nf_0$ (Nyquist frequency)[3]. Because of this assumption, the HRTF must be real-valued at $f = Nf_0$—otherwise the inverse Fourier transform would yield a complex time signal. This constraint is realized by using the absolute pressure at $f = Nf_0$. After these steps, the HRIRs are computed by applying the discrete inverse Fourier Transform to the HRTFs. Fig. 6(c) shows an example of such an HRIR.

After the inverse Fourier transform, the HRIR can be acausal as parts of the energy support appear at end of the impulse responses in cases where a node of the evaluation grid is closer to the ear than to the origin of coordinates. This acausality stems from the pressure division in Eq. (1), which causes the virtual impulse from the source to pass

[3] Alternatively, $f_s/2 = Nf_0 + f_0/2$ could be assumed; however, $f_s/2 = Nf_0$ makes it possible to compute HRIRs with lengths (in samples) equal to powers of two, which is often advantageous in digital signal processing.

the coordinate origin at $t = 0$ if the head were absent. To enforce causality, the HRIRs are cyclically shifted by $0.3 f_s/c$ samples (rounded to the next integer) resulting in the final HRIRs. Fig. 6(d) shows an examples of a final HRTF and HRIR.

3.3 Save and Merge Data

The final data are saved as SOFA files [24, 25] in the folder "`Output2HRTF`" (cf. Fig. 4). SOFA is a standardized file format which can be read by a large number of spatial audio applications (cf. SEC. 3.4). The results are stored in the conventions *SimpleFreeFieldHRTF* and *SimpleFreeFieldHRIR* if data for both ears was calculated, and in the conventions *GeneralTF* and *GeneralFIR* if data for one ear was calculated. Note that multiple SOFA files will be created if the calculations were set up for more than a single evaluation grid. If using a volume velocity source, as in the example presented in this article, the results for the left and right ears are located in two separate Mesh2HRTF project folders and need to be merged. This can be done with the function "`merge_sofa_files`."

3.4 Inspecting the Results

An inspection of the results is important to make sure that the HRTF calculations and post-processing were free of errors. A good first indicator are the report files in the "`Output2HRTF`" folder which are available automatically after the post-processing when using the Python version of "`output2hrtf`." For each calculated source, the file "`report_source*.csv`" contains a tabulated summary condensed from the "`NC*.out`" files including the computation time per frequency step. Known issues are listed in the file "`report_issues.txt`." The MATLAB version of "`output2hrtf`" saves the computation times as "`computationTime.mat`" and provides warning about detected issues directly in the console output of MATLAB.

There is a high chance that the calculation and processing went well, if "`report_issues.txt`" does not exist and no warnings were raised. In most cases, issues are related to a problematic mesh, e.g., a mesh with duplicate vertices or faces, intersections, or too large or irregular faces. A list with frequent issues and workarounds is provided in the Mesh2HRTF documentation [27].

Some issues can be revealed only by more deeply inspecting the results. One example for such an issue is a point source located too close to a mesh. For the inspection, the function "`inspect_sofa_files`" plots selected HRIRs and HRTFs as well as all HRIRs and HRTFs on the horizontal, median, and frontal planes. Figs. 7 and 8 show examples of such plots for HRTFs calculated for the mesh and parameters shown in Figs. 2 and 3, respectively. HRTFs were calculated at a distance of 1.5 m from the origin of coordinates, where source proximity effects are negligible [51]. Note the HRTFs convergence toward 0 dB with decreasing frequency reflecting the decreasing acoustic effect of the head with decreasing frequency. Further, note the ILDs approaching 20 dB at high frequencies for the most-lateral

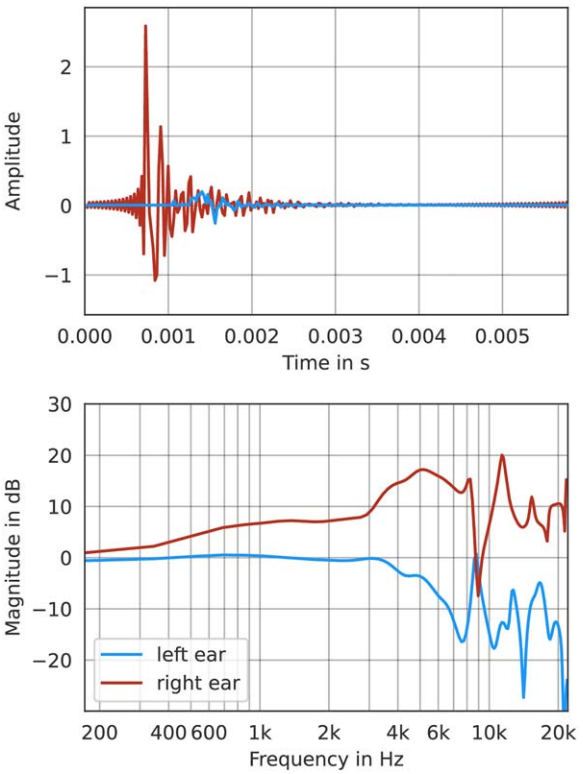

Fig. 7. Example of an HRIR (top) and HRTF magnitude-spectrum (bottom) for a source on the horizontal plane with an azimuth angle of 270° (at the right ear side. The azimuth angle denotes the counter clock-wise angle in the *x*/*y*-plane). HRTFs were calculated for sources at a distance of 1.5 m from the origin of coordinates.

source directions. Small ILDs might indicate numerical issues in the BEM calculation caused by the point source being too close to the mesh, see SEC. 1.4. Also, note the first wave front of the HRIRs shown along the horizontal plane, having the familiar sinusoidal pattern of acoustically measured HRIRs. Finally, note the absence of extreme amplitudes in both the HRIRs or HRTFs, which is an indication for numerical stability obtained in the results from the BEM.

A third possibility to inspect the results is listening to signals spatialized with the calculated HRTFs. The website of the "Handbook of audio technology"[4] lists applications supporting the spatialization with SOFA files, such as the SPARTA Binauraliser Plugin.[5] However, listening requires an additional post-processing step to ensure a natural sound color. For the most common case of headphone listening, a so-called headphone compensation filter is required to remove unwanted coloration introduced by the headphone itself. Unfortunately, this requires matching headphone filters generated from measured or simulated headphone transfer functions [52] that might not always be available. However, the frequency response of headphones is often designed to approximate the diffuse field

[4]https://www.audio-technology.info/ (Chapter Binaural Technology).

[5]https://leomccormack.github.io/sparta-site/.

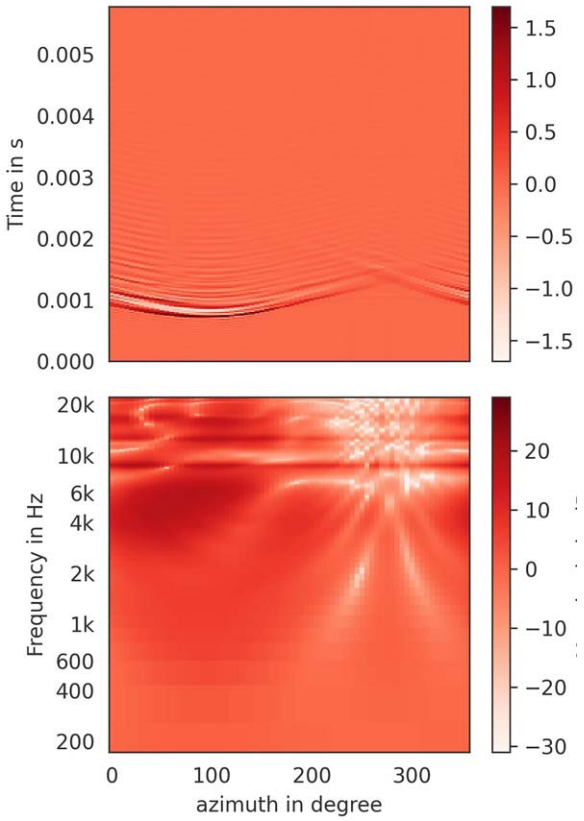

Fig. 8. Example of HRIRs (top) and HRTFs (bottom) of the left ear for sound sources located on the horizontal plane. The azimuth angle denotes the counter clock-wise angle in the *x*/*y*-plane. HRTFs were calculated for sources at a distance of 1.5 m from the origin of coordinates.

transfer function (DFTF) [53], in this case defined as the energetic average of the HRTFs across source positions and ears (omitting the frequency dependency for brevity here and below)

$$\mathrm{DFTF} = \sqrt{\sum_{n=0}^{N-1} |\mathrm{HRTF}|^2_{l,n}\, w_n + \sum_{n=0}^{N-1} |\mathrm{HRTF}|^2_{r,n}\, w_n}, \quad (4)$$

where the index n denotes the source position, $|\cdot|$ the absolute spectrum, and $\sum_n w_n$ normalized area weights for numerical integration. The average across the ears is taken to maintain interaural differences.

The DFTF can be computed directly from the HRTFs and its inverse is often a good and easily realizable alternative to a headphone compensation filter. Applying the inverse DTFT yields the directional transfer function (DTF)

$$\mathrm{DTF} = \frac{\mathrm{HRTF}}{\mathrm{DFTF}} \quad (5)$$

AQ5 that can be used for listening via headphones.

Fig. 9 shows the inverse DTFT for the head in Fig. 2 and an even simpler approximation by means of a parametric bell filter. The inverse DTFT was generated with "`compute_dtfs`" that also returns the directional transfer functions. Note that some applications including the SPARTA Binauraliser directly offer the functionality to

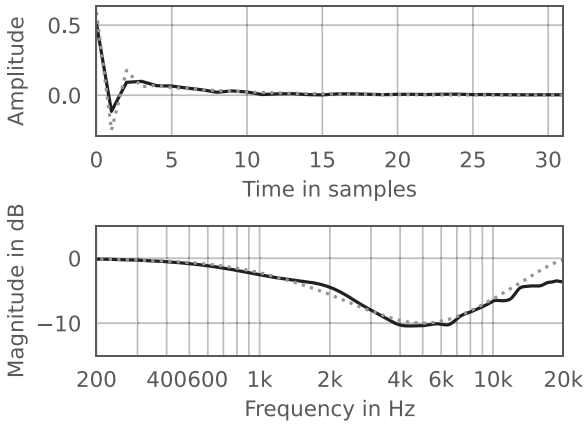

Fig. 9. Example of an inverse DFTF (black, solid) and its approximation by a bell filter (gray, dotted). The bell filter was generated with "`pyfar.dsp.filter.bell`" using a center frequency of 5 kHz, a gain –10 dB, and a quality 0.75.

compute DTFs as part of the auralization chain. If the HRIRs are required at different sampling rates, "`resample_sofa_file`" can be used for the conversion.

4 SCRIPTING AND TESTING

The entire Mesh2HRTF pipeline shown in Fig. 1 is fully scriptable. This opens the possibility for batch processing of large numbers of Mesh2HRTF projects, which can, for example, be relevant for research in the field of HRTF individualization. The functions "`process_multiple_outputs2hrtf`" and "`remove_outputs`" can be used to apply the post-processing to an arbitrary number of project folders and remove intermediate results to free disk space.

The function "`manage_numcalc`" can be used to automatically parallelize HRTF calculations. It can manage multiple project folders and starts a new NumCalc instance for the calculation of the next frequency, whenever sufficient resources are available. To this end, the available memory, number of threads, and CPU load are monitored using the *psutil* package [54]. The default settings use all available RAM and threads and start new instances if less than 90% of the CPU are used. However, all parameters can be controlled by the user to offer more flexibility. For example, a single threaded HRTF calculation for 128 frequencies and the left-ear mesh shown in Fig. 2 done on an Intel i7 processor (eight cores/threads, 4 GHz) with 32 GB RAM required over 7 hours. A manual leverage of the resources between three parallel threads (a conservative estimate to avoid overloading the RAM) reduced the calculation time to approximately 3 hours. By using "`manage_numcalc`," the calculation could be further reduced less than 2 hours.

Mesh2HRTF 1.x also introduces automatic testing of large parts of the workflow using the pytest framework[6] to increases the reliability of the code. This currently includes tests of the Blender export add-on, NumCalc compi-

[6]https://pytest.org/.

lation and calculations, and the entire Python functionality of Mesh2HRTF. NumCalc is tested by calculating transfer functions of a point source, volume velocity source, and plane wave scattered by a rigid and sound soft sphere, and comparing the results against the corresponding analytical solutions.

5 CONCLUSION

The initial development of Mesh2HRTF started in 2013 and resulted in a release of the version 0.4 in 2015 [26]. It formed a solid basis for many HRTF calculations worldwide with more than 30 citations. In 2019, backward incompatible changes in the Python API of Blender v2.80 required a backward incompatible update of the Mesh2HRTF parameter setup to support future versions of Blender. This was taken as a chance to refactor the Mesh2HRTF workflow and introduce new features. The corresponding contributions from the Technical University of Berlin, the ÖAW, and others triggered an important further development of Mesh2HRTF.

In this process, the export interface and the underlying code were completely refactored improving the clarity, usability, and modularity, converging in the code basis for Mesh2HRTF versions 1.x. It provides an easy access to custom and frequency-dependent boundary conditions, the definition of the source type, and custom evaluation grids. NumCalc, the numerical core of Mesh2HRTF, was extended by command-line options to ease parallel calculations and accelerate the numerical calculations. Python functions were added rendering the entire Mesh2HRTF pipeline completely scriptable in a single programming language, offering an improved modularity and consistent version control. Additionally, automatic tests of the project folder created by the Blender export add-on, NumCalc calculations, and the post-processing are available, increasing the quality and reliability of the code. The improved online documentation makes Mesh2HRTF 1.x more easily accessible to new users.

Mesh2HRTF 1.x code and releases are available at https://github.com/Any2HRTF/Mesh2HRTF under the European Public License 1.2 (EUPL 1.2) permitting unrestricted use for everyone. More information as well as online tutorials can be found at https://mesh2hrtf.org.

The authors thank Harald Ziegelwanger for the original development of all Mesh2HRTF versions 0.x within the FWF-funded project LocaPhoto. The work on Mesh2HRTF 1.x was partially supported by the European Union within the project SONICOM (grant number: 101017743, RIA action of Horizon 2020). The authors thank Jens Ahrens and Sebastian T. Prepeliţă for their insightful comments on earlier versions of this work.

7 REFERENCES

[1] H. Møller, "Fundamentals of Binaural Technology," *Appl. Acoust.*, vol. 36, pp. 171–218 (1992 Mar.). AQ6 https://doi.org/10.1016/0003-682x(92)90046-u.

[2] A. S. Bregman, *Auditory Scene Analysis. The Perceptual Organization of Sound* (MIT Press, Cambridge, MA, 1994). https://doi.org/10.7551/mitpress/1486.001.0001.

[3] P. Majdak, R. Baumgartner, and C. Jenny, "Formation of Three-Dimensional Auditory Space," in J. Blauert and J. Braasch (Eds.), *The Technology of Binaural Understanding* (Springer, Berlin, Germany, 2020). https://doi.org/10.1007/978-3-030-00386-9_5.

[4] E. M. Wenzel, M. Arruda, D. J. Kistler, and F. L. Wightman, "Localization Using Nonindividualized Head-Related Transfer Functions," *J. Acoust. Soc. Am.*, vol. 94, no. 1, pp. 111–23 (1993 Jul.). https://doi.org/10.1121/1.407089.

[5] J. C. Middlebrooks, "Virtual Localization Improved by Scaling Nonindividualized External-Ear Transfer Functions in Frequency," *J. Acoust. Soc. Am.*, vol. 106, no. 3, pp. 1493–1510 (1999 Sep.). https://doi.org/10.1121/1.427147.

[6] P. Majdak, B. Masiero, and J. Fels, "Sound Localization in Individualized and Non-Individualized Crosstalk Cancellation Systems," *J. Acoust. Soc. Am.*, vol. 133, no. 4, pp. 2055–68 (2013 Apr.). https://doi.org/10.1121/1.4792355.

[7] R. Baumgartner, P. Majdak, and B. Laback, "Modeling Sound-Source Localization in Sagittal Planes for Human Listeners," *J. Acoust. Soc. Am.*, vol. 136, pp. 791–802 (2014 Aug.). https://doi.org/10.1121/1.4887447.

[8] F. Brinkmann, A. Lindau, and S. Weinzierl, "On the Authenticity of Individual Dynamic Binaural Synthesis," *J. Acoust. Soc. Am.*, vol. 142, no. 4, pp. 1784–1795 (2017 Oct.). https://doi.org/10.1121/1.5005606.

[9] F. Brinkmann and S. Weinzierl, "Audio Quality Assessment for Virtual Reality," in M. Geronazzo and S. Serafin (Eds.), *Sonic Interactions in Virtual Environments*, Human-Computer Interaction Series, pp. 145–178 (Springer, Cham, Switzerland, 2022). https://doi.org/10.1007/978-3-031-04021-4.

[10] C. Guezenoc and R. Séguier, "HRTF Individualization: A Survey," presented at the *145th Convention of the Audio Engineering Society* (2018 Oct.), paper 10129.

[11] K. Pollack, W. Kreuzer, and P. Majdak, "Modern Acquisition of Personalised Head-Related Transfer Functions: An Overview," in B. F. Katz and P. Majdak (Eds.), *Advances in Fundamental and Applied Research on Spatial Audio*, chapter 4 (IntechOpen, Rijeka, 2022). https://doi.org/10.5772/intechopen.102908.

[12] W. Kreuzer, P. Majdak, and Z. Chen, "Fast Multipole Boundary Element Method to Calculate Head-Related Transfer Functions for a Wide Frequency Range," *J. Acoust. Soc. Am.*, vol. 126, no. 3, pp. 1280–1290 (2009 Sep.). https://doi.org/https://doi.org/10.1121/1.3177264.

[13] S. T. Prepeliţă, J. G. Bolaños, V. Pulkki, L. Savioja, and R. Mehra, "Numerical Simulations of Near-Field Head-Related Transfer Functions: Magnitude Verification and Validation With Laser Spark Sources," *J. Acoust. Soc. Am.*, vol. 148, no. 1, pp. 153–166 (2020 Jul.). https://doi.org/10.1121/10.0001409.

[14] Comsol Multiphysics GmbH, "COMSOL," https://www.comsol.com/ (accessed May 13, 2022).

[15] FastBEM, "FastBEM," https://fastbem.com (accessed May 13, 2022).

[16] A. Inc., "Ansys," https://www.ansys.com (accessed November 4, 2022).

[17] U. Iemma and V. Marchese, "AcouSTO," http://acousto.sourceforge.io (accessed May 13, 2022).

[18] F. Hecht, "New Development in FreeFem++," *J. Numer. Math.*, vol. 20, no. 3-4, pp. 251–265 (2012). https://doi.org/10.1515/jnum-2012-0013.

[19] T. Betcke and M. Scroggs, "Bempp," https://bempp.com/ (accessed May 13, 2022).

[20] P. M. Juhl and V. C. Henríques, "OpenBEM," http://www.openbem.dk/ (accessed May 13, 2022).

[21] V. C. Henríques and P. M. Juhl, "OpenBEM - An Open Source Boundary Element Method Software in Acoustics," in *Proceedings of the INTERNOISE: Noise and Sustainability* (Lisbon, Portugal) (2010 Jun.).

AQ7

[22] J. Saarelma, *Finite-Difference Time-Domain Solver for Room Acoustics Using Graphics Processing Units*, Master's thesis, Aalto University, Espoo, Finland (2013 Nov.).

[23] J. Saarelma, "Parallel FDTD," https://github.com/juuli/ParallelFDTD (2014 Jul.).

[24] AES Standards Committee, "AES Standard for File Exchange - Spatial Acoustic Data File Format," *AES69-2022* (2022 Sep.).

[25] P. Majdak, F. Zotter, F. Brinkmann, et al., "Spatially Oriented Format for Acoustics 2.1: Introduction and Recent Advances," *J. Audio Eng. Soc.*, vol. 70, no. 7/8, pp. 565–584 (2022 Jul.). https://doi.org/10.17743/jaes.2022.0026.

[26] H. Ziegelwanger, W. Kreuzer, and P. Majdak, "Mesh2HRTF: An Open-Source Software Package for the Numerical Calculation of Head-Related Transfer Functions," presented at the *Proceedings of the 22nd International Congress on Sound and Vibration* (Florence, Italy) (2015 Jul.).

[27] The Mesh2HRTF Developers, "Mesh2HRTF," https://github.com/Any2HRTF/Mesh2HRTF/wiki (accessed Sep. 1, 2022).

[28] B. Foundation, "Blender," https://www.blender.org/ (accessed May 13, 2022).

[29] P. Cignoni and A. Muntoni, "MeshLab," https://www.meshlab.net (accessed May 13, 2022).

[30] P. Cignoni, M. Callieri, M. Corsini, et al., "MeshLab: An Open-Source Mesh Processing Tool," in *Proceedings of the Eurographics Italian Chapter Conference*, pp. 129–136 (Salerno, Italy) (2008 Jul.). https://doi.org/10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136.

[31] Autodesk, "MeshMixer," https://www.meshmixer.com/ (accessed May 13, 2022).

[32] S. Marburg, "Six Boundary Elements per Wavelength: Is That Enough?" *J. Comput. Acoust.*, vol. 10, no. 1, pp. 25–51 (2002 Mar.). https://doi.org/10.1142/S0218396X02001401.

[33] A. Reichinger, P. Majdak, R. Sablatnig, and S. Maierhofer, "Evaluation of Methods for Optical 3-D Scanning of Human Pinnas," in *Proceedings of the International Conference on 3D Vision*, pp. 390–397 (Seattle, WA) (2013 Jun.).

[34] M. Dinakaran, F. Brinkmann, S. Harder, et al., "Perceptually Motivated Analysis of Numerically Simulated Head-Related Transfer Functions Generated by Various 3D Surface Scanning Systems," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 551–555 (Calgary, Canada) (2018 Apr.). https://doi.org/10.1109/ICASSP.2018.8461789.

[35] W. Kreuzer, K. Pollack, P. Majdak, and F. Brinkmann, "Mesh2HRTF / NumCalc: An Open-Source Project to Calculate HRTFs and Wave Scattering in 3D," in *Proceedings of the Euroregio BNAM Joint Acoustics Conference*, pp. 443–452 (Alborg, Denmark) (2022 May).

[36] Z. S. Chen, G. Hofstetter, and H. A. Mang, "A Symmetric Galerkin Formulation of the Boundary Element Method for Acoustic Radiation and Scattering," *J. Comp. Acous.*, vol. 5, no. 2, pp. 219–241 (1997 Jun.). https://doi.org/10.1142/S0218396X97000137.

[37] W. Kreuzer, "Numerical Simulation of Sound Propagation in and Around Ducts Using Thin Boundary Elements," *J. Sound Vibr.*, vol. 534, paper 117050 (2022 Sep.). https://doi.org/10.1016/j.jsv.2022.117050.

[38] H. Ziegelwanger, W. Kreuzer, and P. Majdak, "A-priori Mesh Grading for the Numerical Calculation of the Head-Related Transfer Functions," *Applied Accoustics*, vol. 114, pp. 99–110 (2016 Dec.). https://doi.org/10.1016/j.apacoust.2016.07.005.

[39] J. Möbius and L. Kobbelt, "OpenFlipper: An Open Source Geometry Processing and Rendering Framework," in *Proceedings of the 7th International Conference on Curves and Surfaces*, pp. 488–500 (Avignon, France) (2010 Jun.).

[40] T. Palm, S. Koch, F. Brinkmann, and M. Alexa, "Curvature-Adaptive Mesh Grading for Numerical Approximation of Head-Related Transfer Functions," in *Proceedings of the Fortschritte der Akustik (DAGA)*, pp. 1111–1114 (Vienna, Austria) (2021 Aug.).

[41] D. Sieger and M. Botsch, "The Polygon Mesh Processing Library," http://www.pmp-library.org (accessed Nov. 6, 2022).

[42] F. Brinkmann, M. Dinakaran, R. Pelzer, et al., "A Cross-Evaluated Database of Measured and Simulated HRTFs Including 3D Head Meshes, Anthropometric Features, and Headphone Impulse Responses," *J. Audio Eng. Soc.*, vol. 67, no. 9, pp. 705–718 (2019 Sep.). https://doi.org/10.17743/jaes.2019.0024.

[43] F. Brinkmann, M. Dinakaran, R. Pelzer, et al., "The HUTUBS Head-Related Transfer Function (HRTF) Database," *FG Audiokommunikation* (2019 May). https://doi.org/10.14279/depositonce-8487.

[44] H. Ziegelwanger, P. Majdak, and W. Kreuzer, "Numerical Calculation of Listener-Specific Head-Related Transfer Functions and Sound Localization: Microphone Model and Mesh Discretization," *J. Acoust. Soc. Am.*, vol. 138, no. 1, pp. 208–222 (2015 Jul.). https://doi.org/10.1121/1.4922518.

[45] P. Stitt and B. F. G. Katz, "Sensitivity Analysis of Pinna Morphology on Head-Related Transfer Functions Simulated via a Parametric Pinna Model," *J. Acoust.*

Soc. Am., vol. 149, no. 4, pp. 2559–2572 (2021 Apr.). https://doi.org/10.1121/10.0004128.

[46] S. Marburg and B. Nolte (Eds.), *Computational Acoustics of Noise Propagation in Fluids - Finite and Boundary Element Methods* (Springer, Berlin, Germany, 2008).

[47] S. Sauter and C. Schwab, *Boundary Element Methods*, Springer Series in Computational Mathematics (Springer Berlin, Germany, 2011). https://doi.org/10.1007/978-3-540-68093-2.

[48] Z.-S. Chen, H. Waubke, and W. Kreuzer, "A Formulation of the Fast Multipole Boundary Element Method (FMBEM) for Acoustic Radiation and Scattering From Three-Dimensional Structures," *J Comput Acoust*, vol. 16, no. 2, pp. 303–320 (2008 Jun.). https://doi.org/10.1142/S0218396X08003725.

[49] R. Coifman, V. Rokhlin, and S. Wandzura, "The Fast Multipole Method for the Wave Equation: A Pedestrian Prescription," *IEEE Antennas Propag. Mag.*, vol. 35, no. 3, pp. 7–12 (1993 Jun.). https://doi.org/10.1109/74.250128.

[50] E. G. Williams, *Fourier Acoustics: Sound Radiation and Nearfield Acoustical Holography* (Academic Press, San Diego, CA, 1999), 1st ed. https://doi.org/10.1016/B978-0-12-753960-7.X5000-1.

[51] D. S. Brungart and W. M. Rabinowitz, "Auditory Localization of Nearby Sources. Head-Related Transfer Functions," *J. Acoust. Soc. Am.*, vol. 106, no. 3, pp. 1465–1479 (1999 Sep.).

[52] I. Engel, D. L. Alon, K. Scheumann, J. Crukley, and R. Mehra, "On the Differences in Preferred Headphone Response for Spatial and Stereo Content," *J. Audio Eng. Soc.*, vol. 70, no. 4, pp. 271–283 (2022 Apr.). https://doi.org/10.17743/jaes.2022.0005.

[53] H. Møller, D. Hammershøi, C. B. Jensen, and M. F. Sørensen, "Design Criteria for Headphones," *J. Audio Eng. Soc.*, vol. 43, no. 4, pp. 218–232 (1995 Apr.).

[54] G. Rodola, "Cross-Platform Lib for Process and System Monitoring in Python," https://github.com/giampaolo/psutil (2022 Nov.).

[55] A. J. Burton and G. F. Miller, "The Application of Integral Equation Methods to the Solution of Exterior Boundary-Value Problems," *Proc. R. Soc. London A.*, vol. 323, pp. 201–210 (1971 Jun.).

[56] M. Duffy, "Quadrature Over a Pyramid or Cube of Integrands With a Singularity at a Vertex," *SIAM J. Numer. Anal.*, vol. 19, no. 6, pp. 1260–1262(1982 Dec.).

[57] G. Krishnasamy, L. Schmerr, T. Rudolphi, and F. Rizzo, "Hypersingular Boundary Integral Euqations: Some Applications in Acoustic and Elastic Wave Scattering," *J. Appl. Mech.*, vol. 57, no. 2, pp. 404–414 (1990 Jun.).

[58] J. Lachat and J. Watson, "Effective Numerical Treatment of Boundary Integral Equations: A Formulation for Three-Dimensional Elastostatics," *Int. J. Numer. Methods Eng.*, vol. 10, no. 5, pp. 991–1005 (1976 Jan.).

[59] Y. Saad, *Iterative Methods for Sparse Linear Systems* (Society for Industrial and Applied Mathematics, Philadelphia, PA, 2003), 2nd ed. https://doi.org/10.1137/1.9780898718003.

[60] P. Sonneveld, "CGS, A Fast Lanczos-Type Solver for Nonsymmetric Linear Systems," *SIAM J. Sci. Comput.*, vol. 10, no. 1, pp. 36–52 (1989). https://doi.org/10.1137/0910004. AQ8

A.1. NUMCALC

NumCalc is the C++ BEM kernel of Mesh2HRTF, which numerically solves the homogenous Helmholtz equation (excitation source is a vibrating element) as well as the non-homogenous Helmholtz equation (source is a point source away from the surface). NumCalc is based on a direct formulation of the boundary integral equation [47, Chapter 3.4.2], which is discretized using collocation with constant elements. It is known that the boundary integral equation formulation for the Helmholtz equation yields a non-unique solution at certain critical frequencies (cf. [55]). To deal with this problem, a formulation using the Burton-Miller Method [55] is used in NumCalc. The singular and hypersingular integrals involved with this discretization are solved using formulations based to the methods proposed by Duffy and Krishnasamy et al. [56, 57]. For the (weakly) singular integrals each triangle is subdivided into six subtriangles and on each of the subtriangles the quadrature is done using 13 Gauss nodes (order 4). For the regular and quasi singular integrals a similar idea as in Lachat and Watson [58] is used. An a-priori error estimator is used to determine the necessary order of the Gauss quadrature, in which the overall error tolerance is set to 10^{-3}, which was found to be a good compromise between accuracy and speed. To avoid having very high quadrature orders (the order of the Gauss quadrature is limited to a range from 2 to 6), elements are subdivided into four sub-elements if the distance between the element midpoint and collocation node is smaller than 1.3 times the area of the element. The factor 1.3 has been set heuristically and in practice it is a good compromise between accuracy and speed of the quadrature. AQ9

The resulting linear system $\mathbf{Ax} = \mathbf{b}$ is solved using a preconditioned conjugate gradient squared solver [59, 60], in which the iteration is stopped if the residual $\mathbf{r}_i = \mathbf{Ax}_i - \mathbf{b}$ after the i-th iteration step satisfies $\frac{||\mathbf{r}_i||}{||\mathbf{r}_0||} < 10^{-9}$. Note that to keep Mesh2HRTF simple to use, all the tolerances and parameters with respect to them are hard-coded. However, in future releases it is planned to include command line parameters, so users can modify these tolerances.

A detailed description of NumCalc would be beyond the scope of this paper. To that end, the authors refer to [35]. Also, a full journal publication concerned with NumCalc, its implementation, and its achievable accuracy is planned in the near future.

THE AUTHORS


Fabian Brinkmann


Jeffrey Thomsen


Sergejs Dombrovskis


Katharina Pollack


Stefan Weinzierl


Piotr Majdak

Fabian Brinkmann received an M.A. degree in communication sciences and technical acoustics in 2011 and Dr. rer. nat. degree in 2019 from the Technische Universität of Berlin, Germany. He focuses on the fields of signal processing and evaluation approaches for spatial audio. Fabian is a member of the AES, German Acoustical Society (DEGA), and the European Acoustics Association (EAA) technical committee for psychological and physiological acoustics.

•

Jeffrey Thomsen holds a B.Sc. in Engineering Physics and M.Sc. in Audio Communication Technology, both from the Technische Universität of Berlin. His work and research focuses mainly on the fields of signal processing, algorithm development, and numerical simulations. Jeffrey is a member of the German Acoustical Society (DEGA).

•

Sergejs Dombrovskis loves identifying and developing new market opportunities for exciting technology. His 2010 M.Sc. in Automotive Engineering from Chalmers University of Technology reflects both lifetime interest in cars and his broad product development perspective on engineering. Currently his scientific focus is on advancing methodology of car audio assessment and individualization to help with design of more desirable, higher-fidelity, and more accessible car audio experiences. Sergejs is a member of the Audio Engineering Society and an audio enthusiast in search of scientifically better sound.

•

Katharina Pollack studied electrical engineering audio engineering at the Technical University and the University of Music and Performing Arts in Graz and is currently doing her Ph.D. in the field of spatial hearing at the Acoustics Research Institute in Vienna. Her main research interest is making personalized head-related transfer functions more accessible to the public, with the focus on parametric and statistic approaches for pinna shape deformation. She is an active member of the Austrian Acoustics Association, Austrian section of the Audio Engineering Society, and Austrian section of the Institute of Electronics and Electrical Engineering.

•

Stefan Weinzierl is the head of the Audio Communication Group at the Technische Universität Berlin. His research is focused on audio technology, virtual acoustic reality, room acoustics, and musical acoustics. With a diploma in physics and sound engineering, he received his Ph.D. in musical acoustics. He is coordinating a Master's program in Audio Communication and Technology at TU Berlin and has coordinated international research consortia in the field of virtual acoustic reality (SEACEN) and music information retrieval (ABC_DJ).

•

Piotr Majdak studied electrical and audio engineering at the University of Technology and the University of Music and Performing Arts (KUG), both in Graz, Austria. In 2008, he received his Ph.D. degree in psychoacoustics and signal processing, and since 2015, he is habilitated in the field of acoustics and audio engineering. He works at the Acoustics Research Institute (ARI) of the Austrian Academy of Sciences at a better understanding of the mechanisms underlying spatial hearing. Computer algorithms, toolboxes, and reproducible research are his keys to reaching out (amtoolbox.org and sofaconventions.org). Piotr is member of the Acoustical Society of America, Association for Research in Otolaryngology, and Austrian and German Acoustic Associations; chair of the technical committee for Psychological and Physiological Acoustics of the European Acoustics Association; and president of the Austrian section of the Audio Engineering Society.

AQ10