forum acusticum 2026
8-12 September · Graz, Austria

PINNA2HRTF: A PIPELINE FOR AUTOMATIC CALCULATION OF HEAD-RELATED TRANSFER FUNCTIONS FROM PINNA MESHES

Felix Perfler[1], Lukas Thalhammer[1], Piotr Majdak[1]

[1] *Acoustics Research Institute of the Austrian Academy of Sciences*

Abstract

Individual head-related transfer functions (HRTFs) can be numerically calculated from 3D ear meshes with the help of the software package Mesh2HRTF. However, the process from the individual ear geometry to a valid project for Mesh2HRTF can be tedious as it involves many steps: head generation, ear stitching, mesh grading, and the setup of the project directories. Here, we introduce Pinna2HRTF, a pipeline, automating that process by taking paired (left and right) pinna meshes stored as STL files as input and outputting a binaural HRTF set stored as a SOFA file. The pipeline, implemented in Python, generates an appropriately sized dummy head, closes the ear canals, stitches the ears onto the head, applies frequency-dependent mesh grading, exports ready-to-run project Mesh2HRTF directories, and runs Mesh2HRTF. Further, an optional stage called Mesh2PPM can be used to register ear geometries to a parametric pinna model (PPM), which then provides ear meshes of sufficient quality for the calculations with properties such as a closed surface, no artifacts, and uniform sampling. Pinna2HRTF is open source and available to the public.

Keywords: *Pinna2HRTF, individualized HRTFs, Mesh2HRTF, SOFA, pinna meshes, numerical acoustics*

1. Introduction

Listener-specific HRTFs are important for accurate spatial audio rendering [1]. While acoustic measurements remain the reference [2], numerical calculation from 3D geometry has become a viable alternative [3, 4, 5]. The software package Mesh2HRTF and its boundary element solver NumCalc can compute HRTFs from meshes, but preparing these meshes still requires considerable manual work. The steps involve: Generating a head, stitching ears onto it, grading the mesh, setting up the simulation. Further, if the ear meshes are not of perfect quality, cleaning up the artifacts, closing the surface, resampling are also required.

Pinna2HRTF automates this workflow. Starting with the left and right ear meshes provided as STL files, Pinna2HRTF handles head creation, ear-canal closing, ear stitching, mesh grading via `hrtf_mesh_grading` [6], and exporting the data project directories for the numeric calculations with NumCalc. The optional stage Mesh2PPM [8] can be used to register ear geometries to the BezierPPM, which is a parametric pinna model (PPM) [7] producing ear geometries that can be individualized and are suitable for HRTF calculations. The final outputs are SOFA files [9] containing binaural sets of HRTFs.

The code of Pinna2HRTF is publicly available at `https://github.com/Any2HRTF/Pinna2HRTF`. Here, we describe how to setup and use Pinna2HRTF.

2. Data layout and installation

Pinna2HRTF requires Python 3.11. The dependencies are installed via `uv sync`. In addition, `hrtf_mesh_grading` [6] and Mesh2HRTF [3, 5] need to be installed separately.

The optional inference step works in a fixed data directory structure, expecting the STL files of the left and right ears to be in the directories called:

```
Target STL Left
Target STL Right
```

respectively. The name of the files must be the same in both directories. Note that multiple files are supported in each directory, facilitating the processing of HRTFs of multiple subjects in a single run.

In the following of this article, we assume the root of the data directory to be `Data`. For pair-wise preprocessing, the paths to the left and right STL files are

***Corresponding author**: felix.perfler@oeaw.ac.at.*

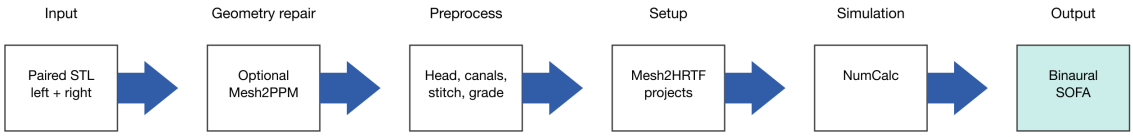

Figure 1: Pinna2HRTF workflow from paired STL files through optional Mesh2PPM registration, preprocessing, NumCalc, and SOFA export.

passed directly with `--left-path` and `--right-path`, as shown below.

3. HRTF calculation

3.1. Preprocessing

For each pair of left and right STL files passed to `hrtf-preprocessing`, the preprocessing includes:

1. creation of a dummy head sized to match head diameter as defined by the distance between left and right ear scan,
2. closing the ear canals if open,
3. cutting an area around each ear on the head and stitching the ear mesh to it,
4. running the mesh-grading algorithm [6],
5. preparing parameters for the numeric calculations by defining the microphone positions, assigning materials, and exporting project directories for NumCalc.

All these steps of preprocessing —from ear meshes to the export of project Mesh2HRTF projects—can be run with a single command:

```
uv run hrtf-preprocessing \
  --left-path "path/to/left.stl" \
  --right-path "path/to/right.stl" \
  --export-path "path/to/export" \
  --mesh-grading-executable \
    "path/to/hrtf_mesh_grading" \
  --Mesh2HRTF-path "path/to/Mesh2HRTF/mesh2hrtf" \
  --head-radius 75
```

The optional `--head-radius` flag places pinnae at the specified lateral head radius; if omitted, the pinnae remain at their original positions.
This creates the output directories containing the Mesh2HRTF projects:

```
path/to/export-Left
path/to/export-Right
path/to/export-intermediates
```

The individual steps of preprocessing can also be run individually. First, to create a dummy head from the ear meshes, run:

```
uv run python \
  HRTFCalculation/Preprocessing/src/create_head.py \
  --left_path "path/to/left.stl" \
  --right_path "path/to/right.stl" \
  --export_path "path/to/export.stl"
```

Then, to close the ear canal, to cut the ear geometry to fit the head mesh, and to stitch the ear to the head, run the following:

```
uv run python \
  HRTFCalculation/Preprocessing/src/ear_canal_closer.py \
  --ear_path "path/to/ear.stl" \
  --export_path "path/to/export.stl"

uv run python \
  HRTFCalculation/Preprocessing/src/cut_eararea.py \
  --head_path "path/to/head.stl" \
  --ear_path "path/to/closed/ear.stl" \
  --export_path "path/to/export.stl"

uv run python \
  HRTFCalculation/Preprocessing/src/head_stitcher.py \
  --head_path "path/to/cut/head.stl" \
  --ear_path "path/to/closed/ear.stl" \
  --export_path "path/to/export.stl"
```

respectively. Note that these operations need to be performed separately for each side of the head and sequentially, since files created in previous steps are required for the next ones. After stitching, `hrtf_mesh_grading` needs to be run on the resulting meshes. Finally, the script `material_assign_and_mesh2input.py` needs to be run to define the microphone positions, assign materials, and export project directories for NumCalc project folders through Mesh2HRTF. For the parameter description for Mesh2HRTF, we refer to the Mesh2HRTF wiki pages, with the most important parameters being the graded head mesh, source type (`"Left ear"` or `"Right ear"`), output path, program path, and frequency-vector definition, which can be set as flags in the commands.

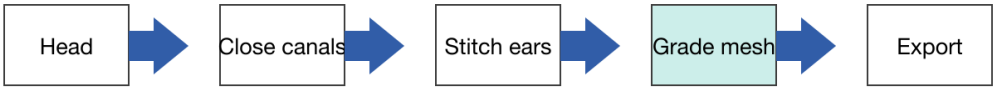

Figure 2: Preprocessing writes export-Left, export-Right, and export-intermediates folders.

3.2. NumCalc and postprocessing

After the preprocessing steps, NumCalc is run to compute the HRTFs. To this end, the two project folders are arranged in a common project directory and NumCalc is run with:

```
uv run hrtf-numcalc \
  --project-path "path/to/Projects" \
  --numcalc-path "path/to/NumCalc" \
  --max-instances 1 \
  --max-cpu-load 90
```

This results in a large number of files, which can then be converted to individual SOFA files and merged into a binaural SOFA file with:

```
uv run hrtf-sofa \
  --left-project "path/to/Projects/Left" \
  --right-project "path/to/Projects/Right" \
  --output-dir "path/to/HRTF" \
  --overwrite
```

This creates SOFA files in the individual left and right project folders and writes the merged binaural SOFA files to `path/to/HRTF`.

4. Optional inference via Mesh2PPM and BezierPPM

If the input ear meshes are not suitable for direct HRTF calculation, an optional inference step can be used before running the HRTF workflow described above.

In order to register the ears to the BezierPPM [7], Mesh2PPM [8] can be used. Mesh2PPM is a pretrained neural network that infers BezierPPM parameters, which are then used to parametrize the BezierPPM and synthesize an individualized ear geometry. To this end, the inference is run on the data directory:

```
uv run hrtf-inference \
  --data_dir /path/to/Data
```

This performs the inference and writes the results into four new directories:

```
Prediction Parameters Left
Prediction Parameters Right
Prediction STL Left
Prediction STL Right
```

The first two directories store the inferred BezierPPM parameters; the latter two directories store the registered ear meshes, which can be used as input to the HRTF workflow described above.

5. Conclusions

Pinna2HRTF automates the workflow from pinna meshes to a SOFA file containing individual binaural set of HRTFs. It handles head creation, ear stitching, mesh grading, and setup of the Mesh2HRTF project directories in a few command-line calls. The optional inference stage provides BezierPPM-based ear registration. Pinna2HRTF is available at `https://github.com/Any2HRTF/Pinna2HRTF`.

6. Acknowledgments

We thank Katharina Pollack and Wolfgang Kreuzer for their help in configuring Mesh2HRTF and Florian Pausch for providing Mesh2PPM.

7. Funding

F. Perfler is supported by a DOC Fellowship (A 27288) of the Austrian Academy of Sciences at the Acoustics Research Institute. Partial funding was received from the European Union's Horizon 2020 research and innovation program under grant agreement No. 101017743 (project "SONICOM").

References

[1] J. Blauert, *Spatial Hearing: The Psychophysics of Human Sound Localization*. MIT Press, 1996.

[2] P. Majdak, P. Balazs, and B. Laback, "Multiple exponential sweep method for fast measurement of head-related transfer functions," *J. Audio Eng. Soc.*, vol. 55, no. 7/8, pp. 623–637, 2007.

[3] H. Ziegelwanger, W. Kreuzer, and P. Majdak, "Mesh2HRTF: An open-source software package for the numerical calculation of head-related transfer functions," in *Proc. 22nd Int. Congr. Sound Vib.*, 2015.

[4] 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.

[5] 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. 71, no. 7/8, pp. 502–514, 2023.

[6] H. Ziegelwanger, W. Kreuzer, and P. Majdak, "A priori mesh grading for the numerical calculation of

the head-related transfer functions," *Appl. Acoust.*, vol. 114, pp. 99–110, 2016.

[7] F. Perfler, F. Pausch, K. Pollack, N. Holighaus, and P. Majdak, "Parametric model of the human pinna based on Bezier curves and concave deformations," *Comput. Biol. Med.*, vol. 188, p. 109817, 2025.

[8] F. Pausch, F. Perfler, N. Holighaus, and P. Majdak, "Prediction of parameters of a pinna model from synthetic geometries using a vision transformer," *J. Acoust. Soc. Am.*, vol. 159, no. 6, pp. 5578–5598, 2026, doi: 10.1121/10.0043919.

[9] Audio Engineering Society, *AES69-2022: AES standard for file exchange – Spatial acoustic data file format*, 2022.