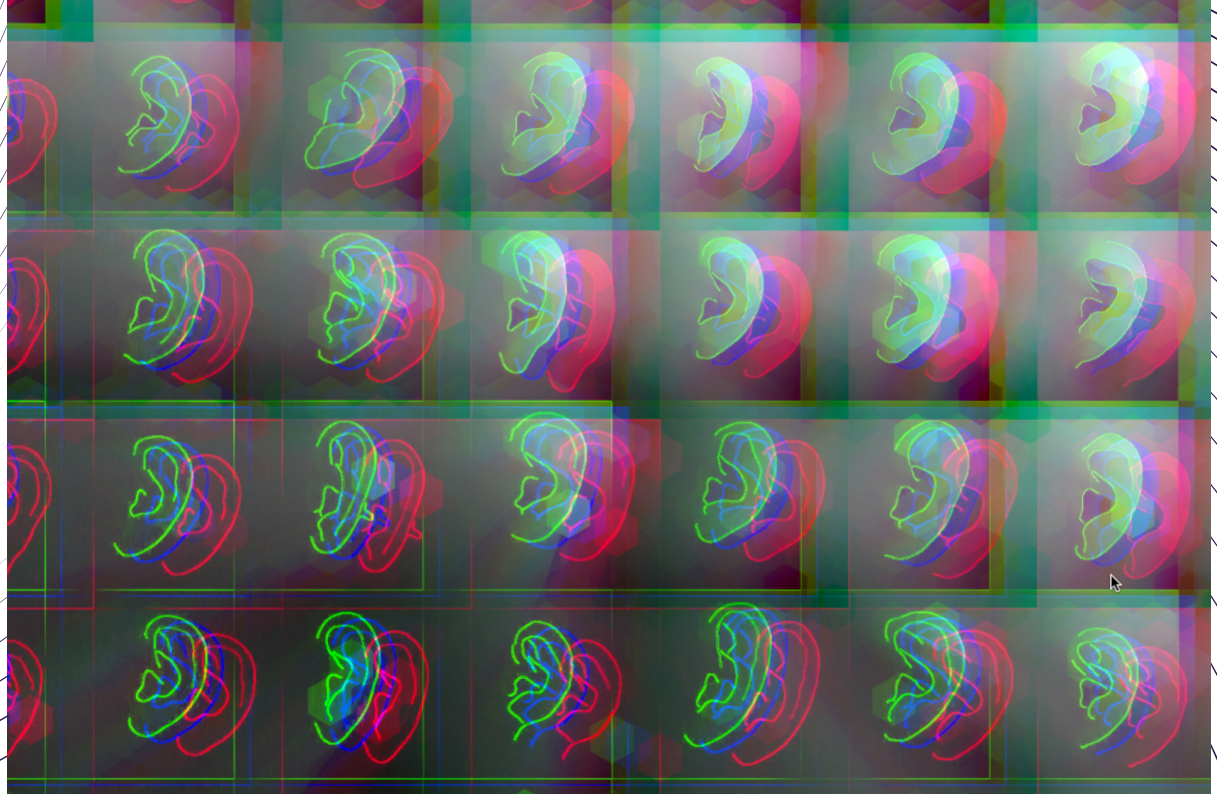

# A hybrid approach to structural modeling of individualized HRTFs

Generating and combining pinna responses, head-and-torso filtering, and interaural time difference data

Riccardo Miccini

Sound and Music Computing, 2021-02

Master's Project

# A hybrid approach to structural modeling of individualized HRTFs

Generating and combining pinna responses, head-and-torso filtering, and interaural time difference data

Riccardo Miccini

Sound and Music Computing, 2021-02

Master's Project

This document was authored in a Microsoft Visual Studio Code environment using the Markdown markup language, and typeset by the author using the LaTeX typesetting system. Most of the graphics shown herein are generated using `matplotlib`. The bibliography is typeset using `biblatex`.

The template has been created by Jesper Kjær Nielsen; Markdown was originally developed by John Gruber and Aaron Swartz; LaTeX was originally developed by Leslie Lamport, based on TeXcreated by Donald Knuth; conversion between the two languages was provided by Pandoc, developed by John MacFarlane. The image in the front cover was made using Python, CoGe VJ, and GIMP.

AALBORG UNIVERSITY
STUDENT REPORT

**Department of Architecture,**
**Design and Media Technology**
Aalborg University
www.aau.dk

**Title:**
A hybrid approach to structural modeling of individualized HRTFs

**Theme:**
HRTF Individualization

**Project Period:**
Fall Semester 2020

**Project Group:**
N/A

**Participant(s):**
Riccardo Miccini

**Supervisor(s):**
Simone Spagnol

**Copies:** 1

**Page Numbers:** 72

**Date of Completion:**
January 31, 2021

**Abstract:**

Providing users with a personalized HRTF set is paramount for an immersive VR experience, free from localization errors and inside-the-head sound perception. However, direct acoustic measurement of the user's HRTF requires specialized apparatuses and is often strenuous and expensive. We present a hybrid approach to HRTF modeling which requires only 3 anthropometric measurements and an image on the pinna contours. A prediction algorithm based on variational autoencoders synthesizes a pinna response from its contours, which is used to filter a measured head-and-torso response. The ITD is then manipulated to match that of a HUTUBS dataset subject minimizing the predicted localization error. The performances are evaluated using spectral distortion and a perceptual localization model. While the latter is inconclusive regarding the efficacy of the structural model, the former metric shows promising results with encoding HRTF datasets.

# Acknowledgements

As I am typing these words, I am reminded that my time at Aalborg University is about to be over: no more exams to postpone, no more deadlines I can extend. I would like to thank everyone at AAU, my classmates, teachers, and all of the campus staff for offering the best study environment I could possibly wish for and the tools to be my most successful self.

A very special mention goes to my supervisor Simone for his extraordinary dedication, support, patience, and technical insight. Thanks for making these past two years some of the most formative and prolific ones in my life. I wish you good luck with all your future endeavors, and I hope to see you soon, perhaps at the next physical conference.

I would also like to thank my colleague Jonas, who believed in my project so much that he ended up using my individualized HRTFs for his ass-kicking thesis, in what ended up being a desperate race against time, sleep deprivation, Unity3D quirks on OSX, and lab access restrictions.

Speaking of which. This past year has been crazy, for reasons we're all aware of. Therefore, I would like to thank all the people who helped me stay sane, focused, and motivated throughout it. I owe you one. Thank you Federico, Adriana, and Julia for the productive work sessions, less productive sessions involving just the right amount of wine, spontaneous audiovisual jams sessions, and overall for being the most amazing friends I could ask for. Thank you Chiara, Connor, Irina, Jan, Rares, and all the other people I met at Colab320, for offering me a physical space to escape to, the possibility of fostering my passion for music and geeky stuff, and for being some of the most welcoming and supportive people I've met here. Oh, and thank you Anne for putting up with me every day. How do you even do it? Please don't stop. I love you.

While I am stuck here in Denmark with the aforementioned people, my gratitude stretches all across Europe. To my friends in Italy — Chiara, Davide, Riccardo, Emanuele, Camilla, Giulia, Eugenio, Edoardo, and Fabio, as well as those scattered anywhere else — Claudio, Boris, Anita: you guys had a hell of a year. Thanks for holding tight, keeping me digital company, and providing ever-so-tangible relief during these months of uncertainty. I can't wait to hug you again.

As obvious as it may sound, none of this would have been possible without my family, who taught me the principles and values to live a successful life and set a good example from day 0; after so many years spent in education, I am still cherishing their lessons. Thanks to my parents Maria Luisa and Gianfranco, my siblings Natascia, Daniele, and Marco, my grandmother, and my little niece Matilde, whom I can't wait to play and take silly selfies with.

Finally, I would like to dedicate this work to my grandfather Natale, who unfortunately is not here to witness the fruits of his hard work and dedication to his family. I miss you every day. There are so many things I wish I could ask you or tell you about. I hope you're still proud of me.

# Contents

# Preface

This thesis was conceived as the final project of the Sound and Music Computing Master's program at Aalborg University, Copenhagen. The work presented herein was conducted during my last term, between September 2020 and January 2021, and was supervised by Simone Spagnol and Stefania Serafin. The report investigates the design, development, and evaluation of an HRTF individualization method based on deep learning algorithms.

In September 2019, prompted by my interest in machine learning, Simone took me under his wing, allowing me to consolidate my professional and academic background with peer-reviewed work and field research. Indeed, the foundation of this research originates from a article presented at the 2020 IEEE Conference on Virtual Reality and 3D User Interfaces by Simone and me in March 2020. Furthermore, essential elements for this work were collected during my internship at the University of Iceland in November and December 2019. Finally, the technical framework and knowledge base acquired during the first and second semester projects at AAU, delivered in December 2018 and June 2019 respectively, provided me with the confidence to take on this ambitious challenge.

Thus, this project is the culmination of more than 2 years of effort. After spending this time investigating deep learning technologies on diverse audio applications such as timbre morphing, speech enhancement, and lastly spatial audio, my curiosity is hardly quenched and new areas of interest arose. I look forward to spending the upcoming years working on them.

Aalborg University, January 31, 2021

Riccardo Miccini
<rmicci18@student.aau.dk>

# Chapter 1

# Introduction

Virtual, augmented, and mixed reality (collectively known as XR) research have made significant strides over the last decades, and acoustic virtual reality technologies have found applications in a wide array of fields such as navigation aids for the visually impaired [74], entertainment systems, and tools for acoustic and sound-quality measurements [8]. Binaural rendering technologies are therefore employed, to provide the user with a believable and immersive 3D auditory scene. This consists in processing a sound to simulate the effects of the user's head, pinnae, and torso, which vary depending on the location of the sound source. The effect of the user's anthropometry with respect to the direction of a sound is described mathematically by the *head-related transfer function* (HRTF), and will be further discussed in section 1.1. The word "binaural" implies the presence of two audio channels, a left and a right one, which are processed differently and delivered to the user's respective ear canals using headphones.

For the user to be able to experience a vivid and immersive 3D virtual environment, the two audio channels must adequately replicate the physical phenomena caused by the user's anthropometry. Throughout their lineage, evolutionary pressure caused humans' anatomy and brain to develop several sophisticated mechanisms allowing them to detect the position of sound sources and parse auditory scenes [50]: these are collectively known as *localization cues*. These cues can be divided into *interaural* — that is, based on the difference between the signal arriving at each ear canal — and *monaural*, which are extracted independently for each ear. Interaural cues are typically involved in the localization of sounds along the lateral direction, and come in the form of *interaural time difference* (ITD) and *interaural level difference* (ILD). The former is based on the difference between time-of-arrival of a given input stimulus, which may fluctuate by as much as $\pm 1$ ms between the ipsilateral and contralateral directions [65]. This cue is particularly sensitive to the spectral makeup of the source signal since the cochlea is only sensitive to phase difference of frequencies up to about 1.6 kHz and the distance between the pinnae

is finite; furthermore, the lack of a distinct temporal envelope may also hinder its effectiveness [7]. Conversely, for signals containing frequency components higher than 400 Hz, the acoustic shadow cast by the head introduces detectable differences in loudness. Monaural cues, also known as spectral cues, are described in the following section, and research about their physical origins and modeling is presented in chapter 2. Other types of localization cues are known to exist, such as *dynamic cues*: in this case, the subtle variations in the other cues occurring when the user moves their head are examined in order to resolve the precise location of a sound source [7].

Binaural audio rendering technologies usually employ generic HRTFs that are derived from standard head and torso simulators or averages of HRTFs from multiple subjects. However, these generic HRTFs fail to accurately replicate the spectral cues associated with the user's own anthropometry, causing front-back confusion and inside-the-head localization [81]. In order to provide the most immersive experience possible, individualized HRTFs must be acquired from the user, through a strenuous and time-consuming process of acoustical measurement, which usually requires a custom-built sound playback and recording apparatus. Fortunately, over the past few decades, alternatives to measured HRTFs have been researched and developed. This field of research is known as HRTF individualization.

The following sections provide an overview of the HRTF and its characteristics, as well as an introduction to the field of HRTF individualization.

## 1.1 The head-related transfer function

As mentioned earlier, HRTFs mathematically encode the impact of a user's morphology on an incoming sound as a function of its spatial location. This is due to the acoustic phenomena occurring on the interface between two different media — in this case, the air and the human body — such as reflection, diffraction, and diffusion, causing the human body to act as a filter. Most notably, the parts of the human body interacting with the incoming wavefront and therefore known to contribute to the HRTF are the shoulders, the torso, the head, and most prominently the pinnae of the listener [1]. Along the median plane, the reflection of soundwaves over the shoulders take the form of semicircular patterns originating at 90°, behaving like a variable comb filter [19] (see Figure 1.1 or, most prominently, Figure 7.1). The head plays a role in attenuating frequencies higher than $\sim$1.5 kHz (i.e. having a wavelength shorter than the head width) for contralateral locations along the horizontal plane [7].

The impact of the pinnae is most noticeable across the median plane, where it provides salient vertical localization cues in the form of spectral peaks and notches [24] — these can be seen in Figure 1.1. When isolated from the aforementioned contributions, it takes the name of *pinna-related transfer function* (PRTF). A summary

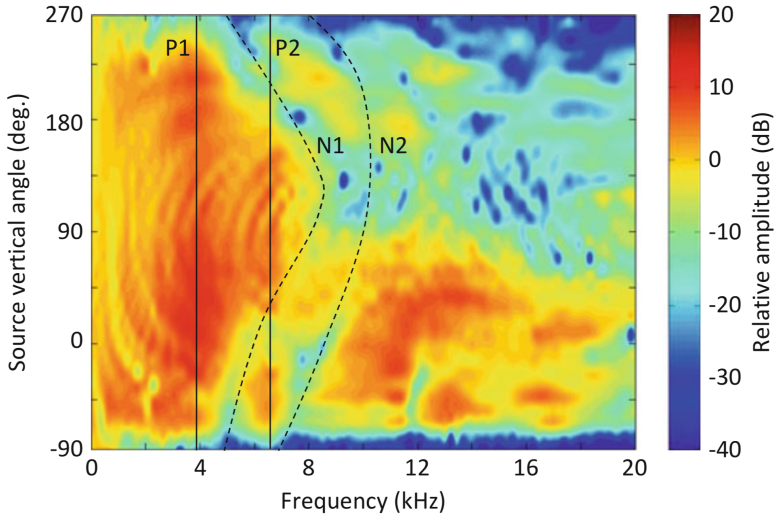


**Figure 1.1:** Median plane HRTF, with most prominent peaks and notches highlighted. Reproduced from [32].

of the relevant literature on the spectral features introduced by the pinnae and their relevance in spatial localization can be found in chapter 2.

An HRTF is usually encoded in the frequency domain as a minimum-phase system, bereft of its ITD information, which must be accounted for separately. The time-domain equivalent of the HRTF is the head-related impulse response (HRIR). Other methods for encoding HRTF information include surface spherical harmonics [21] and spherical wavelets [46].

Observing the HRIRs of a subject — such as those in Figure 1.2 — highlights the temporal appearance of the aforementioned contributions: for instance, the impact of the pinna is limited to the first 0.2 ms from the onset [61, 68], while shoulders and torso reflections appear as amplitude peaks delayed by 0.5 ms to 1 ms depending on the elevation angle. Across the horizontal plane, the onset delay caused by the distance between the pinnae as well as the head shadowing effect are also clearly visible.

The process of collecting HRTFs from a human subject usually involves playing an acoustic stimulus from each of the loci of the desired spatial grid and recording it from the entrance of the ear canal of each ear, using a pair of miniaturized in-ear microphones. Ensuring a consistent spacing between grid points and an adequate spatial resolution is an open challenge, and acoustic research labs around the world came up with ad-hoc solutions meeting their requirements, such as loudspeaker arrays or pivoting arms — e.g. the setup in Figure 1.3. These acoustical measurements are usually carried out in an anechoic or semi-anechoic environment to minimize the presence of wall reflection and reverberation in the recorded data.

The Dirac impulse is the most straightforward choice of audio stimulus, in that

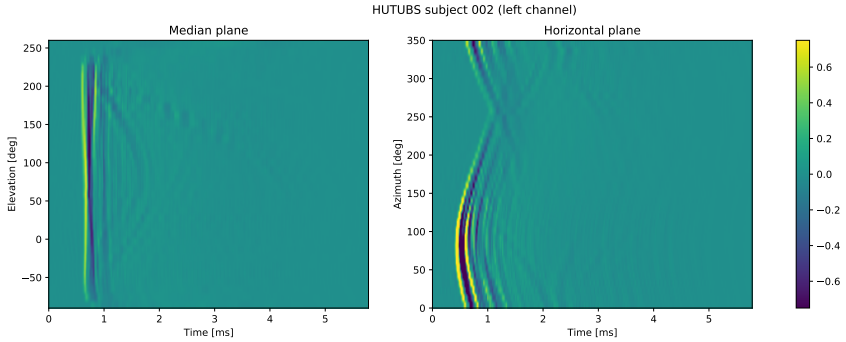


**Figure 1.2:** HRIR amplitudes plotted across the median and horizontal planes, showing shoulder reflection patterns and different onset times respectively.

the recorded output already constitutes the HRIR. However, when investigating acoustical systems where a high SNR is crucial, it is often desirable to perform measurements using a sweep signal [60]. In this case, a sinusoidal excitation signal is generated, with its instantaneous frequency increasing exponentially over time, and lying within the range of interest for the measurements, typically 20 Hz to 20 kHz. The HRIRs are thus recovered by convolving the resulting recorded sweep response with the inverse response of the original excitation.

**Figure 1.3:** Example of measurement apparatus for HRTF set acquisition on human subjects. Picture taken as part of the *Sound of Vision* research project[1].

## 1.2 HRTF Individualization

Most of the commonly used binaural audio rendering technologies rely on generic or standardized HRTF sets. HRTF individualization is an active field of research within the sound and music computing community, aiming at providing users with a personalized HRTF set that reduces the localization error thus improving the immersiveness of acoustic VR experiences. There exist several strategies for individualizing HRTF. According to [26], these can be divided into the following three macro-categories, which relate to the type of output obtained by the method at hand:

- **Selection**: the most suited HRTF set is chosen from a large dataset;
- **Adaptation**: a generic or selected HRTF set is manipulated to better fit the user;
- **Synthesis**: an HRTF set is generated from scratch, usually by means of numerical simulation of structural modeling.

Since the objective is to individualize the HRTFs according to the characteristics of a given user, these approaches must depend on user data, which can be classified into:

- **Anthropometrics**: this type of data is collected directly from the user, relates to their morphology, and can be expressed objectively. It includes anthropometric measurements of the torso, head, and pinnae, photographs of the subject, and 3D scans or point-clouds.
- **Perceptual feedback**: this includes indirect information provided by the user during the evaluation of a given HRTF or HRTF set, such as preference expressed using a Likert scale, or localization error. Methods based on perceptual feedback may be less susceptible to measurement biases.

Over the years, an abundance of techniques has been developed, combining one or more combinations of approaches and user data, and relying on tools and methods borrowed from the fields of digital signal processing, statistics, machine learning, and numerical modeling. The most notable ones are presented in chapter 2, with particular emphasis on those based on anthropometric data.

---

[1] `https://soundofvision.net/consortium-meeting-at-the-lodz-university-of-technology-lodz-poland-september-22-25-2015/`

This project builds upon the work in [51]. Throughout the project, the author investigates whether it is possible to synthesize individualized HRTF data using features automatically extracted from images of the pinnae. The project offers the following contributions:

- A deep-learning-based solution for synthesizing pinna responses from user pictures;
- An hybrid approach for combining the aforementioned pinna responses, the interaural time difference matched from a dataset, and the head-and-torso response through a customized structural HRTF model;
- An evaluation of the performances of the pinna responses for vertical localization, as well as a study of the impact of various hyperparameters and extensions.

This first chapter provided an introduction to binaural audio rendering technologies and the field of HRTF individualization. The rest of this document is structured as such: chapter 2 summarises the major contributions to the field found in the relevant literature. In chapter 3, the general structure of the individualization algorithm is presented, along with the relevant deep learning concepts, as well as the datasets used in this project. The implementation and results of the different sub-tasks involved in the project are elaborated in chapter 4, chapter 5, chapter 6, and chapter 7. These results are further discussed in chapter 8, while chapter 9 summarises the main highlights, providing concluding notes and suggestion for future improvements.

# Chapter 2

# Related work

The human's ability to perceive sound in a 3D environment and estimate its spatial origin has been a topic of scientific inquiry for more than a century. One of the earliest contributions come from [62], who in 1907 proposed the existence of perceptual cues for the localization of sound sources from lateral directions, namely the aforementioned interaural time difference and interaural level difference. A first attempt at measuring HRTFs of human subjects comes from the work of [80], where the frequency response of the pinnae was collected across the horizontal plane over intervals of 15°. Later, the work of [65] determined the numerical relation between ITD, ILD, and azimuth angle of a sound source.

Perceptual cues affecting the localization along the vertical direction are more elusive than their horizontal counterparts, and have been researched since the '70s. In particular, [29] established that spectral cues for vertical localization exist between 4 and 16 kHz, and that only sounds occupying this frequency range can be reliably localized along the median plane. These cues take the form of spectral peaks and notches. In [67], six resonant modes of the pinna have been identified (see Figure 2.1) which, according to [77], are thought to cause the most prominent peaks in the HRTF.

On the other hand, the exact origin of spectral notches is more difficult to trace. Initially, they were thought to relate to reflections on the concha walls causing the pinna to behave like a delay-and-add system in the time domain [4]. More recently, they have been explained by the interaction between propagating waves and the pressure anti-node forming in the upper pinna cavities [77]. While the center frequency of peaks is relatively insensitive to changes in elevation of the sound source [36], pinna notches, especially N1, are generally seen to increase with the elevation angle, providing a salient elevation cue [29]; conversely, notches exhibit little variation with changes in azimuth [47].

The specific contribution of the aforementioned spectral features to vertical sound localization is still a topic of inquiry. In their work, [35] achieved localization

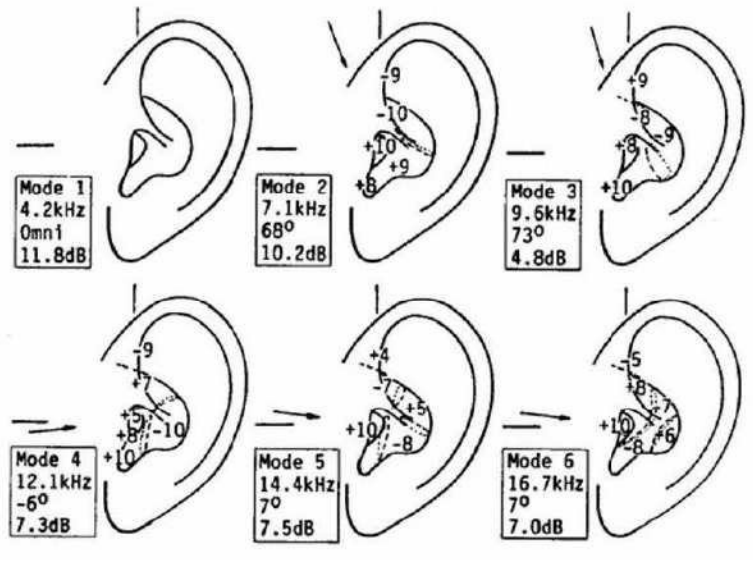


**Figure 2.1:** Pinna resonant modes with their frequencies, source angle, and responce (left corners). Reproduced from [67].

performances similar to the subjects' own HRTF for the front and rear portions of the median plane by synthesizing a parametric HRTF composed of only the first peak and the first two notches. A more recent experiment [33] further improved the localization performances for a larger subset of elevations by introducing a second peak.

The next sections will introduce the most relevant work on HRTF individualization using conventional and deep learning approaches, respectively.

## 2.1 Conventional approaches

Over the past few decades, several models aimed at undersanding and predicting the spectral features of the HRTF have been developed. Pioneering works such as the double-delay-and-add time-domain model [79], physical flange-and-cavity model [66], and the diffraction-reflection model [47] managed to mimic the characteristic peak-notch patterns seen in HRTFs, although it is unclear how they can be customized to fit a specific listener. Conversely, mathematical models based on *principal component analysis* (PCA) have also been proposed [41, 54], where the HRTF magnitude response can be approximated using a subset of principal components, each relating to different aspects such as azimuth or elevation of the sound source as well as anthropometry.

The individual contributions of head, torso, and pinna anatomy can be isolated and investigated, in order to replicate the spectral effects of the underlying physical

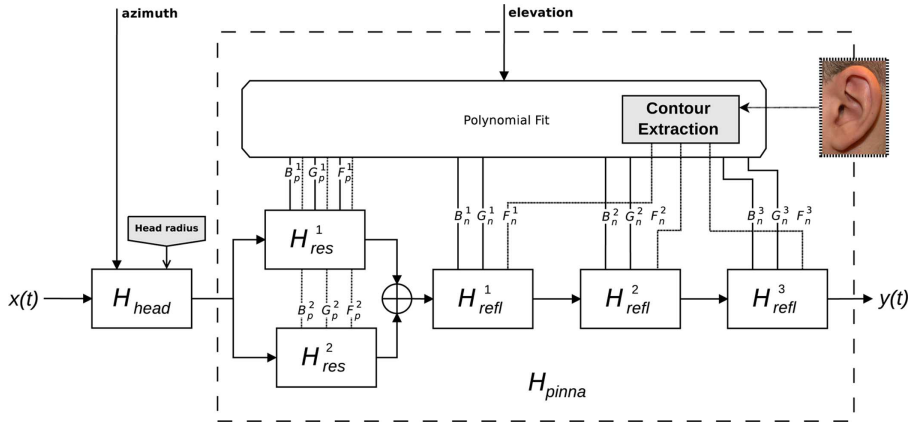


**Figure 2.2:** Structural model for synthesizing individualized HRTFs using pinna contours. Reproduced from [71].

phenomena. A *structural model* is a system whereby such spectral effects are independently modeled according to anthropometric data and combined to create a personalized HRTF. Structural modeling of HRTFs finds its origin in [10], where the physical sources of sound diffraction, delay, and reflection are simulated in the time domain. In more recent times, frequency-domain structural models of the pinna have also been proposed. A structural model of the pinna has been proposed in [71] (see Figure 2.2), whose parameters are given by a simple model that converts 2D reflection paths on three distinct pinna edges into notch frequencies. Using the same reflection model together with a subset of standard anthropometric parameters, a linear regression model to estimate N1 frequencies from individual anthropometry was also introduced [72], as well as a marginally improved one based on PCA [52]. Models to estimate spectral peaks from individual anthropometric parameters have also been recently proposed [57, 58]. Similarly, recent works employ multiple linear regression to predict the spectral features of HRTFs from the subjects' anthropometric parameters, obtaining synthetic HRTFs similar to the measured ones [34, 78].

The recent availability of large amounts of computing power paved the way for the application of numerical techniques such as *boundary element method* (BEM) [38] and the *finite-difference time-domain* (FDTD) [56]. These methods simulate the propagation of acoustic waves around the subject's pinnae, head, and torso, which are provided in the form of 3D meshes. While it is possible to closely match the spectral features of measured HRTFs, acquiring a sufficiently accurate 3D model of the subject proves challenging and may require specialized tools such as MRI or CT scans, as well as substantial human labor. Numerical simulation methods have also been used to investigate the physical phenomena affecting the characteristic of spectral peaks and notches [37, 77].

This section presented works on HRTF individualization based on numerical, structural, and machine learning methods with a focus on HRTF synthesis; refer to [26] for a more comprehensive treatise of the topic.

## 2.2 Deep learning approaches

The widespread adoption of deep learning technologies — discussed in section 3.1 — has affected the field of HRTF individualization as well, and several methods have been proposed. One of the first works is [48], where a stacked denoising autoencoder is trained to encode and reconstruct HRTFs from multiple subjects. The resulting latent representation is then manipulated using feedback from the user to optimize their localization performances. While this novel solution proves more effective than PCA-based encoding or simple averaging of multiple HRTF sets, it was evaluated using a simulated agent.

The solution devised in [82] similarly relies on user feedback to adjust the latent vector from which an HRTF is synthesized. However, their model features a richer input data representation comprising neighboring HRTFs and both frequency and time representations of the HRTF. Furthermore, the ANN architecture employed here is more sophisticated than the dense one used above. It consists of a conditional variational autoencoder featuring 3D convolutional layers and custom adaptive residual blocks capable of decomposing the latent variables into individual contributions. A user study with 20 participants shows that the individualized HRTFs score higher in terms of spatial accuracy. A block diagram of the solution is shown in Figure 2.3.

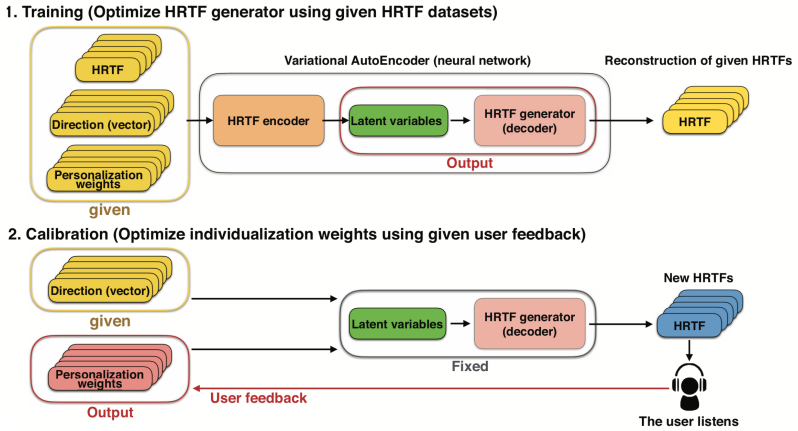


**Figure 2.3:** Overview of the HRTF individualization algorithm based on variational autoencoders, using perceptual feedback from the user to tune the latent parameters. Reproduced from [82].

Several methods involving anthropometric data as input have also been proposed. In particular, [16] uses a *deep neural network* (DNN) to predict the impulse

response of a subject's left pinna from measurements of the pinnae, head, and torso, obtaining a lower spectral distortion with regards to the reference HRTF, compared to the dataset average. Similarly, [13] trains 2500 DNNs to predict a given HRTF from the user's anthropometric data. Each DNN is trained on HRTFs associated with a specific spatial orientation, and the training time is reduced by initializing the weights of a DNN with those of the DNNs trained on adjacent orientations. The work in [44] uses two separate neural networks: a densely-connected one processing the anthropometric measurements of head and torso, and a convolutional one taking an image of the user's pinna as input. The outputs of these networks are then fed into a DNN which is trained to predict a given HRTF. The results show an improvement in localization accuracy when compared with the dataset average or a DNN model using anthropometric measurements alone.

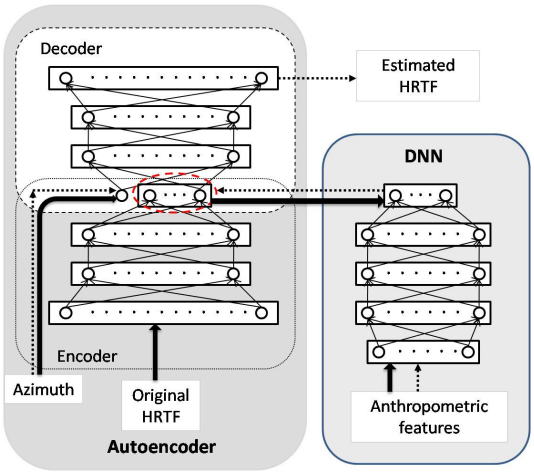


**Figure 2.4:** Architecture of an HRTF individualization system based on deep learning models. The thick lines show the data flow during training, while the dotted lines represent its use in evaluation. Reproduced from [14].

Several recent studies focused on autoencoding HRTF data. Most notably, [14] trains a dense autoencoder to reconstruct the magnitude response of an HRTF from its latent representation and its azimuth coordinate. The encoder part of the network projects each HRTF onto a latent space of reduced dimensionality. A separate DNN is then trained to predict these latent vectors using anthropometric data as input. The final system, composed of the aforementioned DNN and the decoder part of the network and shown in Figure 2.4, has been evaluated using a spectral distortion metric for HRTFs at selected azimuth angles, with no conclusive results. Lastly, in a very recent paper [15], a convolutional denoising autoencoder is trained on 2D frequency-elevation input features, derived from listener-specific directional transfer functions. The resulting compressed representations were then evaluated against PCA in terms of storage requirements and fidelity (measured

as spectral distortion), proving better according to both criteria. Table 2.1 shows a summary of the methods presented in this section.

**Table 2.1:** Summary of HRTF modeling or individualization works using deep learning.

| Ref. | Data | Model architecture | Input | Notes |
|---|---|---|---|---|
| [48] | CIPIC | Stacked denoising AE | HRTF | User feedback |
| [82] | CIPIC | CVAE with ResNet | 4D HRTF structure | User feedback |
| [16] | CIPIC | DNN | Anthropometrics | Target as HRIRs |
| [13] | CIPIC | 2500 DNNs | Anthropometrics | |
| [44] | CIPIC | CNN, DNN $\rightarrow$ DNN | Anthropometrics, edges | |
| [14] | CIPIC | AE, DNN | Anthropometrics | Only azimuth |
| [15] | ARI | Conv. denoising AE | 2D HRTF structure | Only compression |

# Chapter 3

# Methods and tools

The structural model presented in this project is based on a pipeline composed of several deep learning models, as well as conventional DSP blocks. It has been aptly defined as *hybrid* in that it combines synthesized, selected, and measured components. An overview of the processing pipeline and its constituting elements here can be seen in Figure 3.1. In particular, the architecture comprises:

- A deep learning sub-system made up of three distinct models, capable of synthesizing PRTFs from an image of the pinna or analogous 2D features.
- A DSP sub-system implementing a structural model where an HRTF set comprising only of shoulders and head reflection effects is filtered using the pinna response described above, converted into impulse responses, and processed to match the ITD of a fitting subject from an HRTF database.

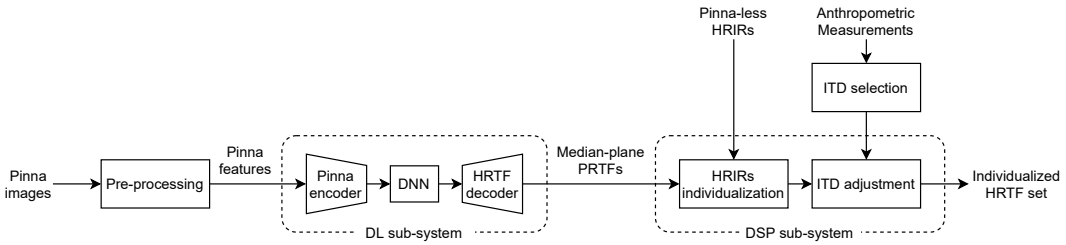


**Figure 3.1:** HRTF individualization pipeline, with its input, outputs, and constituting elements.

The first part of the solution comprises the following three building blocks:

- A *variational autoencoder* (VAE) whose encoder is used for deriving a compact representation of input pinna 2D features — such as pictures, depth maps, or extracted edges — called $z_{\text{ear}}$.

- A *conditional variational autoencoder* (CVAE) whose decoder is similarly used to synthesize a pinna frequency response from a compact representation, called $z_{\text{hrtf}}$.
- A *deep neural network* (DNN) capable of predicting the compressed representation $z_{\text{hrtf}}$ from $z_{\text{ear}}$.

Figure 3.2 provides an overview of the relationship between each component, as used during training and evaluation (in green). The software architecture for the deep learning sub-system was developed in Python using the PyTorch[1] open-source deep-learning library. The library provides a wide range of optimized operations on tensor data with optional GPU-based hardware acceleration, as well as an automatic differentiation module used for computing the gradient during training and a selection of state-of-the-art optimization algorithms.

The development of the necessary software framework was assisted by the PyTorch Lightning[2] open-source library, which provides a high-level interface for the most common tasks involved with training and deploying a deep learning model — including training, validation, testing, data generation, and metrics logging and monitoring. The proper handling and propagation of configuration data such as model hyperparameters and training and data processing arguments was ensured by the widespread usage of JSON files, command-line arguments, and flags parsed by the relevant scripts. Furthermore, the TensorBoard[3] interface was used for monitoring the model performances during training, compare model variations, and visualize their results such as generated data or prediction errors.

The models employed in the first step are trained separately on their respective relevant datasets and then combined into a prediction script capable of generating an individualized PRTF. The prediction script can be used independently from the training and development code base, and only requires the pre-trained model weights to work. The theoretical background of each of these deep learning architectures is covered in chapter 4, chapter 5, and chapter 6.

A separate collection of scripts takes care of performing the last steps of the pipeline and generate an individualized HRTF set in a format compatible with most binaural rendering engines; this is described in chapter 7. The scripts are implemented using MATLAB and combined into an evaluation script parametrized according to the user anthropometry and the PRTFs generated by the deep learning sub-system. Finally, the data analyses presented in the results sections of each of the following chapters were performed in an interactive Python notebook environment called JupyterLab[4] with the help of data processing and visualization tools from

[1] `https://pytorch.org/`
[2] `https://www.pytorchlightning.ai/`
[3] `https://www.tensorflow.org/tensorboard/`
[4] `https://jupyter.org/`

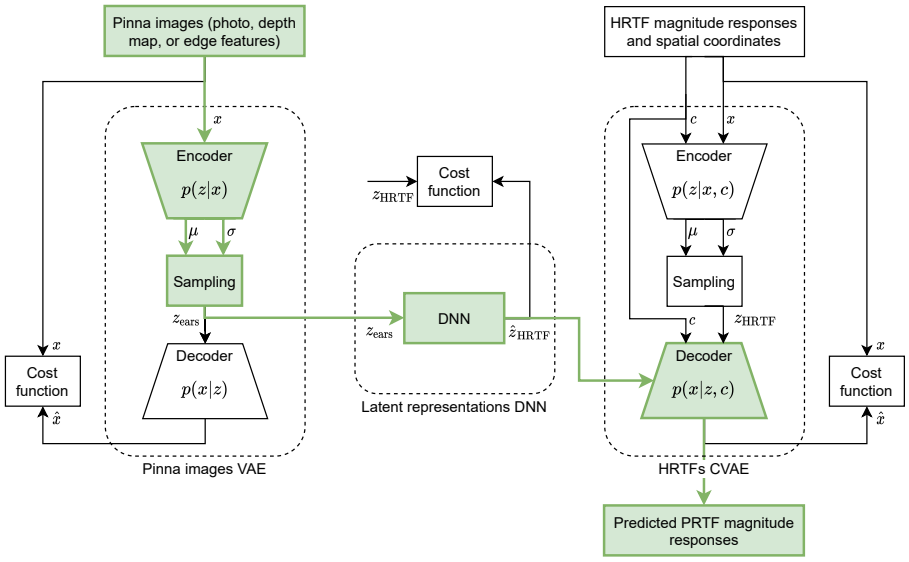


**Figure 3.2:** Overview of the pinna response individualization sub-system. The dotted rectangles represent each distinct deep learning model; the arrows and blocks highlighted in green are the ones used during evaluation.

the SciPy[5] toolbox for science and engineering.

The following sections provide an introduction to deep learning and its core concepts, and present the datasets used in this project.

## 3.1 Deep learning

Deep learning is a branch of machine learning interested in *artificial neural networks* (ANN), i.e. algorithms consisting of several computational layers and nonlinear operations. ANNs were originally inspired by biological systems, such as how neurons in the brain receive impulses from their neighbors through their dendrites and propagate them further through their axons. Nevertheless, they can also be seen as systems capable of approximating any continuous function within a given compact interval, as proven mathematically in 1989 [18]. An ANN is characterized by its trainable parameters: their amount determines the complexity of a model. An abundance of network topologies, architectures, and other heuristics has been developed, some also inspired by biological processes. The word "deep" implies the presence of a large number of successive layers.

The field of deep learning has remained dormant due to the computational costs and complexity associated with training. However, over the past decade, the

[5] `https://www.scipy.org/`

development of several contributions led to its resurgence. These contributions include pre-training techniques based on restricted Boltzmann machines (RBM) capable of speeding up convergence [31], as well as the adoption of GPU hardware for computing, which resulted in super-human performances at the 2012 ILSVRC challenge [42]. Since then, deep learning has gradually superseded traditional techniques in an ever-increasing number of fields such as computer vision, natural language processing, recommendation systems, and so forth. In more recent years, GPU-accelerated deep learning frameworks such as Tensorflow, Keras, and PyTorch as well as the availability of large-scale datasets greatly increased the accessibility of solutions based on deep learning.

For a deep learning model to become effective at performing a given task, it must first be trained. This is done by iteratively updating its trainable parameters to minimize a cost function, which usually is a metric of difference between the network output and its expected target. First, the slope of the cost function with regards to all the trainable parameters, also known as *gradient*, is computed through the process of *back-propagation*. Since a neural network can be seen as a large composite function, this consists of applying the chain rule for derivation across all the layers of the network.

Subsequently, an optimization algorithm is used to adjust the trainable parameters. This is commonly done through a process of gradient descent, where the trainable parameters are updated by subtracting a value proportional to the gradient. Nevertheless, more complex optimization algorithms have been developed, which offer adaptive learning rates for each parameter or account for momentum; a more comprehensive treatise on optimization algorithm can be found in [64].

The process of training described above is performed over a subset of data, called mini-batch. Compared to computing the gradient over the entire dataset, this has the advantage of introducing noise that may help leap out of suboptimal minima, although convergence might be slower. A trained model must be effective at performing its task on unseen data that were not part of the training set. This ability is called generalization, and is acquired when the neural network has learned the variance in the data caused by its intrinsic patterns and structure. If left training for too long, a model might begin learning the random fluctuations in the data caused by noise — a process known as *overfitting*. This can be avoided by occasionally computing the cost function on a small batch of unseen data, called validation set, and interrupting the training as soon as the model performances on the validation set appear to degrade.

## 3.2 Datasets

The availability of large amounts of data is paramount for fitting deep learning models, and large scale datasets exist for several computer vision tasks. However,

due to the complexity associated with collecting HRTF measurements on human subjects, the number and size of such datasets are much more limited. Moreover, combining multiple HRTF datasets proves challenging: the intrinsic differences in acquisition setup, lab conditions, and post-processing may comprise a larger source of variations than the differences between individual HRTFs [14]. Similarly, user data such as anthropometric measurements and ear images or mesh data are not regulated by any standard or convention and are thus equally heterogeneous. The following subsections introduce some of the most popular datasets with are relevant for the tasks at hand, motivate their usage or lack thereof, and present the post-processing and data augmentation methods adopted in the training and evaluation stages.

### 3.2.1 HRTFs

An HRTF set is a collection of impulse or magnitude responses belonging to a given subject and recorded over a finite spatial grid of sound sources. An HRTF dataset comprises multiple HRTF sets, usually one for each subject. HRTF sets from different datasets may vary substantially from each other, for example in terms of their spatial resolution, characteristics of sound emitters and receivers (model, frequency response), recording environment, and so forth.

For this reason, the *Spatially Oriented Format for Acoustics* (SOFA) was developed, with a focus on interchangeability and extendability, allowing the storage of spatially-oriented audio information (such as room impulse responses, headphone impulse responses, loudspeaker directivity, etc) along with its metadata. Each of these types of data is characterized by a different set of variables, attributes, and fields; the naming scheme of the data fields and their dimensionality constitute a *SOFA convention*, and are regulated by the AES standard AES69-2015[6]. In the case of HRTF data, the convention is called `SimpleFreeFieldHRIR`, and it includes:

- Spatial coordinates of the receivers, expressed using the cartesian system;
- Tensor of HRIRs of size $M \times R \times N$, representing the number of spatial positions from which the impulses were collected, the number of receiver channels, and the number of samples on each response, respectively;
- Tensor containing the spatial coordinates for each of the positions, expressed using a spherical system;
- Orientations of the various element involved, expressed as unit vectors pointing up and ahead;
- Metadata about the HRTF set, such as database name, version history, licensing, and issuing author and organization;

[6] `https://www.aes.org/publications/standards/search.cfm?docID=99`

- Metadata about the HRTF acquisition setup, such as description of the sound source, receivers, room, as well as relevant external documentation.

Thanks to recent standardization efforts, the most prominent HRTF datasets have already been converted and released in the SOFA format. The ones that have been considered by this research were the CIPIC dataset [2], which is one of the first publicly available datasets comprising human subjects and thus extensively featured in the literature; the ARI dataset [49] also featuring human subjects; the VIKING dataset [73, 75] collected using a standard KEMAR mannequin and custom-molded pinnae from life-like statues; HUTUBS [9, 22], a relatively new dataset which also features high-resolution 3D head models of its subjects, as well as simulated responses. The characteristics of each dataset are summarized in Table 3.1.

**Table 3.1:** Summary of HRTF datasets.

| Dataset name | Subjects | Positions | Additional data |
|---|---|---|---|
| CIPIC | 45 | 1250 | Anthropometric meas., pinna photos |
| ARI | 200+ | 1550 | Anthropometric meas. |
| VIKING | 21 | 1513 | Anthropometric meas., 3D pinna scan |
| HUTUBS | 96 | 440 | Anthropometric meas., 3D head scan |

Ultimately, the HUTUBS dataset was chosen due to its relatively large number of subjects and the availability of rich anthropometric data such as head, torso, and pinnae measurements and 3D models of the head. An example of the median plane response from a HUTUBS subject is shown in Figure 3.3.

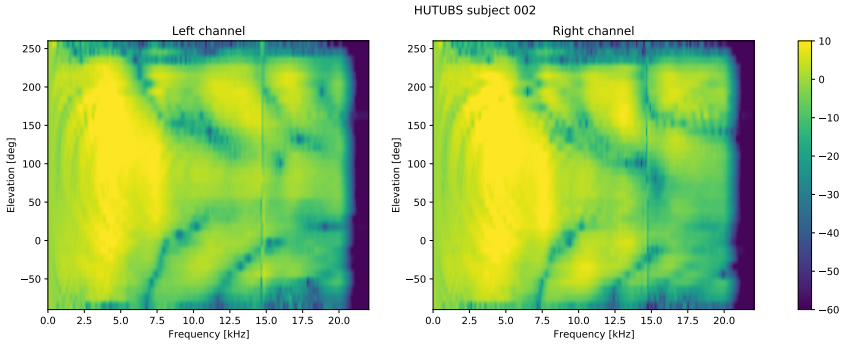


**Figure 3.3:** Example median plane HRTFs from a HUTUBS subject.

### 3.2.2 Ears images

The kind of user data employed in this project must fulfill the following two requirements: on one hand, it must be available in a given HRTF dataset, so as

to make it possible to train a deep learning model on pairs of input and desired output. On the other hand, it must be easy to collect from the user, ideally through inexpensive means.

Ear pictures can be trivially gathered from users, in a way that is nonintrusive and fast. For these very reasons, ear recognition is an actively researched area within the field of biometrics.

**Figure 3.4:** Images of the ears from an AMI subject. The first picture shows the left pinna (flipped); the others show the right one facing left, zoomed in, down, straight, right, and up.

The ear image datasets used in this project can be divided into two categories: publicly available collections of annotated pictures, commonly used for ear detection and recognition tasks, and images extracted from a 3D representation of a subject for which HRTFs are available. The formers include the AMI Ear Database [25] comprising high-quality images of the pinnae of 100 subjects, taken from 7 different yaw and pitch angle combinations and under consistent illumination, position, and camera settings (see Figure 3.4); the AWE Dataset [20] featuring 10 pinnae images of varying quality and size for each of its 100 subjects, gathered from the Internet, cropped around the ear, and annotated; the IITD Ear Dataset [43], composed of 493 images from 125 subjects, taken under a variety of indoor lighting conditions and angles. While there is no correspondence between the subjects of these datasets and the HRTFs mentioned in the previous section, it was theorized that extending the training set with them would improve the generalization abilities of the autoencoder. Indeed, it is believed that by introducing some variability in orientation, illumination, and camera setting, the network should be able to derive a more robust internal representation.

In order to predict HRTFs from ear images, there must be a correspondence between the two sources of data. Since no HRTF is available for any of the images in the datasets mentioned above, pinna images of subjects with a known HRTF must be obtained. These were generated from the 3D head meshes of 55 HUTUBS subjects out of the 58 for which a model was available, discarding repeated measurements and non-human subjects.

A script was developed to load the mesh, place it in a 3D scene together with a camera pointing on either side of the head, align the entrance of the ear canal with the cameras, and render it into either a depth map or a grayscale image. In

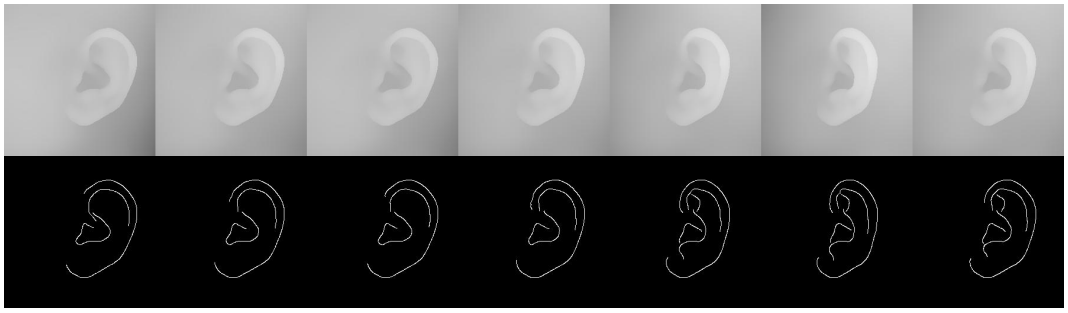

**Figure 3.5:** Pinna depth (top row) and edge (bottom row) features from a HUTUBS subject, over a range of yaw (azimuth) variations.

the latter case, the scene was illuminated using a directional lamp, simulating an infinitely far point light source, located at the camera position and pointing towards the head.

Due to the limited amount of available data, augmentations were introduced in the form of slight variations in camera yaw and pitch, using the ear canal entrance as the pivot point so as to keep it at the center of the image. Finally, a third dataset consisting of ear shape outlines has been created by feeding the depth map dataset into a Canny edge detection algorithm [12]. These two latter features are shown in Figure 3.5.

# Chapter 4

# Encoding of pinna images

This first part of the research focused on deriving useful features from pinna images, which can later be used as predictors for the HRTFs. Specifically, the features correspond to the compressed encoding, also known as *latent representation*. For this task, it was decided to employ a variant of autoencoders called *variational autoencoders* (VAE) [40].

Regular autoencoders are a class of artificial neural networks that can learn a compressed data representation in an unsupervised manner. They are composed of an encoder network $f()$ and a decoder network $g()$ such that $g(f(x)) = g(z) = \hat{x}$, where $z$ is the latent representation and $\hat{x}$ is the reconstructed input. The layers within the encoder and decoder are arranged so as to create a computational bottleneck where the two are linked. When trained to minimize the mean squared error between the true and predicted data, the autoencoder learns to encode and decode its input through a lower-dimensional space.

VAEs, on the other hand, are significantly different in that they are probabilistic models mapping an input sample to a probability distribution. Furthermore, the latent space distribution is constrained into an isotropic multivariate Gaussian with the help of a Kullback-Leibler divergence term in the loss function. These properties prevent the input data points to be encoded in specific sparse areas of the latent space and make VAEs particularly suitable as generative models; however, in this case, only the encoder part is used. This encoder network approximates the probability $p(z|x)$ — that is, the distribution of the latent variable $z$ given the input data $x$. This distribution is parametrized by its mean value $\mu$ and the logarithm of its variance $\sigma$ (for numerical stability), which are the two outputs of the encoder. During training, a value of $z \sim \mathcal{N}(\mu, \sigma)$ is sampled from the distribution and fed into the decoder, which attempts to reconstruct the original input.

In practice, since this stochastic sampling process does not allow the gradient to propagate through, the latent vector is computed as $z = \mu + \varepsilon\sigma$ where $\varepsilon$ is a random variable distributed according to a standard Gaussian: this is known as the

*reparametrization trick*. An example of a VAE architecture can be seen in Figure 4.1.

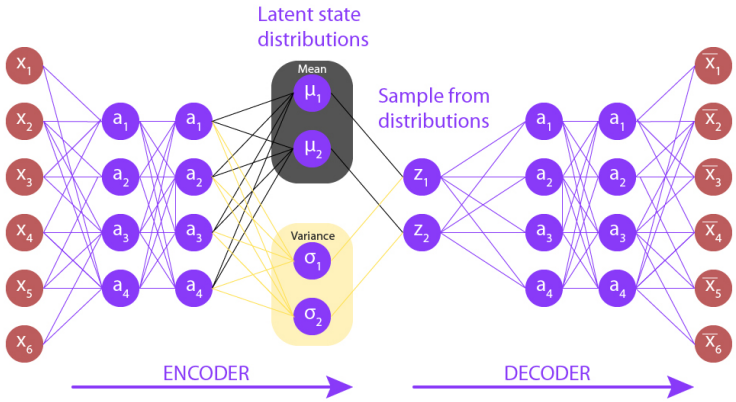


**Figure 4.1:** Variational autoencoder architecture, with its inputs, outputs (both in red) and hidden layers (in purple) highlighted[1].

Like most other ANN architectures, autoencoders may employ several different types of layers. One of the earliest and most common ones is called *dense* or *fully-connected* layer. A network composed entirely of dense layers is known as a *multilayer perceptron* (MLP). A dense layer is composed of computational nodes, called neurons, which are connected to all of the nodes of the preceding layer. Each node computes a weighted sum of its input, adds a bias, and applies a nonlinear activation function.

Despite being extremely powerful, dense layers bear a large number of trainable parameters and do not account for the spatial arrangement of its input. *Convolutional layers* address these shortcomings by computing the dot product between the input and a tensor of arbitrary shape and values, called kernel. In this way, the weights of the layers, corresponding to the values of the kernels, are shared across all of the input features. The reduction in the number of trainable parameters provides a regularization effect that is known to improve generalization.

For computer vision tasks, 2D convolution is a popular choice; nevertheless, other types of convolutional layers such as 1D or 3D are commonly used for time-domain audio signals and video or point cloud data respectively. 2D-convolutional layers take a 3D input, such as an image with 3 color channels or a single grayscale one, apply a stack of convolutional kernels, and output a number of features called filters or channels. The shape of the kernel, the number of channels, and the behavior around the edges of the input are all hyperparameters of the convolutional layer. ANNs employing convolutional layers are also called *convolutional neural networks* (CNN), and typically include other types of layers such as nonlinear

[1] `https://www.jeremyjordan.me/variational-autoencoders/`

activations, pooling layers for dimensionality reduction, or batch normalization for improving convergence — see Figure 4.2 (left) for an example.

Very deep neural networks are susceptible to the gradual decrease in gradient magnitude at each successive layer, which causes training to be increasingly slow or halt entirely. This problem is known as *vanishing gradient*. Residual networks (ResNet) [28] were developed to address this problem. ResNets are a class of very deep CNNs composed of multiple consecutive blocks which, in turn, typically contain one or more batch normalization, activation, and convolutional layers, also arranged consecutively. Besides the main computational path described above, each residual block implements a secondary path, known as *skip connection*, that connects the first and last layers directly; these can be seen in Figure 4.2 (center). In this way, the convolutional layers within a block need only learn the difference — hence the name "residual" — between its input and output. This is known to provide a richer gradient to later layers, which greatly benefits training; thus, ResNets can be extremely deep. Neural networks employing residual blocks have found widespread adoption in image classification, image recognition, and other computer vision tasks.

Yet another configuration of convolutional layers is found in the work of [76], where a block of convolutional layers, humorously called *inception module*, is employed. Within one such module, several convolutional layers with differently sized kernels are applied in parallel, and their respective output features are concatenated or summed. The rationale behind this design choice is to let the network discover the most relevant feature for the task at hand, which may be expressed by kernels of different sizes at different points of the computational graph. Deep inception networks feature a large number of layers: to limit their computational complexity, each convolutional layer is preceded by another one of kernel shape $1 \times 1$, which reduces the dimensionality of its input. This topology is exemplified in Figure 4.2 (right).

## 4.1 Proposed architecture

Based on the architectures described above, three different VAE topologies have been designed and implemented. These are:

- **ConvVAE**: a variational autoencoder composed of stacks of $3 \times 3$ 2D convolution, batch normalization, and activation layers. The encoder and decoder networks follow a roughly symmetrical structure. The former ends with two dense layers — one for the mean and one for the log-variance — which feed on the flattened features from the convolutional stacks. Analogously, the decoder begins with a dense layer, whose output is reshaped and fed to the subsequent convolutional stacks. The number of stacks, channels on

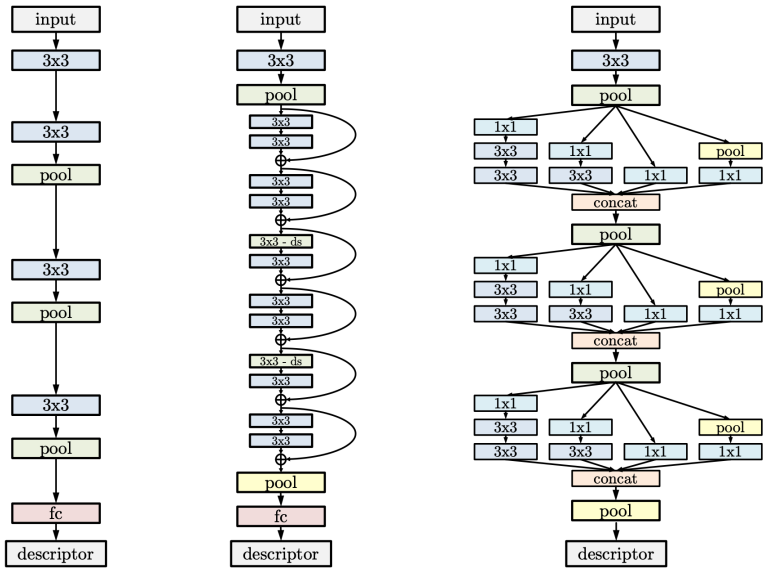


**Figure 4.2:** Convolutional neural network topologies. From left to right: classical CNN, residual layers, and inception modules.

each convolutional layer, and the size of the latent space are all configurable parameters. The chosen activation function is *Leaky ReLU*.

- **ResNetVAE**: a VAE based on residual blocks, as implemented in the `torchvision`[2] library. The encoder comprises a plain convolutional frontend, four successive residual blocks, average pooling, and two fully-connected layers. The decoder follows a similar structure but in reverse order. The makeup of each residual block, the latent space size, and the arrangement of the first and last layers are all hyperparameters. The activation function is employed here *ReLU*.

- **InceptionVAE**: a VAE based on inception modules. Each module is composed of four convolutional stacks with kernels of shape $1 \times 1$, $3 \times 3$, $5 \times 5$, and $7 \times 7$ respectively, preceded by a single $1 \times 1$ convolutional layer for dimensionality reduction. Each stack is composed of 2D convolution, batch normalization, and ReLU activation. Furthermore, two extra paths with max-pooling layers of shape $3 \times 3$ and $5 \times 5$ are provided. The input of each parallel path is padded to obtain equal output shapes, and their respective outputs are summed together. The modules are arranged into four stages, each comprising several modules and separated by downsampling or upsampling stacks for encoder and decoder, respectively. Finally, dense layers with no activations are featured before and after the network bottleneck. The configurable hyperparameters are the size of the latent space and the number

[2] `https://pytorch.org/docs/stable/torchvision`

of inception modules per stage.

Each of the three models was trained in a similar way, i.e. by fitting a small batch of images, computing the output and the latent distribution parameters of $z$, calculating the cost function, and updating the trainable weights using the Adam optimizer [39]. The learning rate was initially set to $1 \times 10^{-4}$ and scheduled to decrease every time there would be 30 epochs with no improvement with an exponential decay factor of 0.5624, corresponding to a tenfold reduction after four iterations. Each model was allowed to train for up to 1000 epochs; if an improvement in the validation set was registered, the new model weights would be stored. If no improvement were registered for more than 100 epochs, the train would terminate.

The three datasets described in subsection 3.2.2, consisting of depth maps, renderings, and edge features, were used. Each dataset was parametrically generated with an image size of $256 \times 256$ pixels and variations in pitch and yaw from $-15^\circ$ to $15^\circ$ with a step of $5^\circ$, for a total of 49 different orientations. This accounted for 5390 images of left and right pinnae, belonging to 55 HUTUBS subjects. Of these, 392 images belonging to 4 subjects were set aside for testing, while a randomly selected subset corresponding to 10 % of the remaining ones was used for validation during training. For the network trained on grayscale renderings, the AMI, AWE, and IITD datasets were merged before this last step. Finally, to further extend the size of the datasets, three types of noises were introduced: gaussian noise, salt and pepper noise, and speckle noise, to be applied during training. The noise algorithms were taken from the `skimage`[3] library. Since each noise type, as well as no noise, had a 25 % probability of being applied, the datasets were effectively augmented by a factor of 4.

## 4.2 Results

Initially, to determine the best performing model, each of them was trained on the dataset of pinna renderings derived from HUTUBS (see subsection 3.2.2). The training was conducted on the Aalborg University's machine learning workstation, equipped with three Nvidia TITAN X Pascal graphics cards with 12 GB of memory each, an Intel Xeon E5-2620 v3 hexa-core CPU, and 132 GB of RAM. Each model was allowed to train for up to 1000 epochs, and the batch size was set to maximize the available resources of a single GPU, i.e. a value of 64, 8, and 32 for the **ConvVAE**, **ResNetVAE**, and **InceptionVAE** models respectively.

One variation of each model was used, parametrized by the values shown in Table 4.1. With these hyperparameters, the models comprised approximately 5 000 000, 18 000 000, and 5 000 000 trainable weights respectively. The **ResNetVAE**

[3] `https://scikit-image.org/`

model, originally developed to work on RGB images, takes a 3-channel input, which accounts for its considerably higher complexity. To ensure a fair comparison between the networks, their latent spaces were fixed to 64 dimensions.

**Table 4.1:** Summary of the model hyperparameters. Cells are filled only when applicable.

| Hyperparameter | **ConvVAE** | **ResNetVAE** | **InceptionVAE** |
|---|---|---|---|
| Decoder channels | [256, 256, 128, 64, 32] | | |
| Encoder channels | [32, 64, 128, 256, 256] | | |
| Input channels | 1 | | |
| Encoder lin. size | | 512 | |
| Encoder type | | resnet18 | |
| First conv | | False | |
| Repeats per block | | | 1 |
| KL coefficient | 0.01 | 0.001 | 0.001 |
| Latent size | 64 | 64 | 64 |

To monitor performances, metrics such as overall loss, reconstruction loss term, Kullback-Leibler divergence loss term, and current learning rate were collected at every epoch during training for both the training and validation sets. Furthermore, pictures containing pairs of original and reconstructed images were generated for a single batch of validation data, every 10 epochs. These data could then be visualized in realtime through TensorBoard.

The **ConvVAE** model trained for 266 epochs over one hour and a half before halting its progress and reaching convergence, the **ResNetVAE** model trained for 413 epochs over approximately 31 hours, and the **InceptionVAE** model trained for 554 epochs over 14 hours. Due to differences in the scaling of the loss function among the different models, it was not possible to compare their training history graphs. Once each model was trained, they were evaluated on the basis of their reconstruction performances. The test set, composed of pinna images from 4 unseen subjects, was fed onto each network and the resulting reconstructed images were visualized. A sample of it can be seen in Figure 4.3.

To objectively evaluate the performances of the models, the following three image similarity metrics — available in the `scikit-image` library — were computed on the test set: normalized root-mean-square deviation (NRMSD), peak signal-to-noise ratio (PSNR), and structural similarity index (SSIM) [83]. These results are shown in Table 4.2. Somewhat surprisingly, the metrics seem to concur that the **ConvVAE** model achieves the best performances, despite a visual inspection of the image in Figure 4.3 seems to contradict this verdict. Due to the lack of an objectively identifiable best-performing model, it was decided to proceed with **InceptionVAE**, which offers the most perceptually truthful reconstruction with the

**Figure 4.3:** Example of pinnae from test set subjects, reconstructed by the different VAEs. From top to bottom: input data, **ConvVAE**, **ResNetVAE**, and **InceptionVAE**.

least amount of trainable parameters.

**Table 4.2:** Summary of the image similarity metrics for the three models.

| | NRMSD | PSNR | SSIM |
|---|---|---|---|
| **ConvVAE** | 0.064 | 24.556 | 0.898 |
| **ResNetVAE** | 0.084 | 22.159 | 0.891 |
| **InceptionVAE** | 0.073 | 23.359 | 0.897 |

After choosing the most suitable network architecture for encoding pinna images, the following step consisted in investigating the network's behavior with different 2D features — namely the pinna depth maps and edges. Thus, the **InceptionVAE** model was trained on these two other datasets, using the same hyperparameters mentioned above. Furthermore, a third training was performed, based on a combined dataset comprising HUTUBS grayscale renderings and pictures from the AMI, AWE, and IITD ears datasets. Analogously to the first step, reconstructed images were generated from the training set and shown in Figure 4.4. Furthermore, the image similarity metrics mentioned above were computed on each training attempt and summarized in Table 4.3.

**Table 4.3:** Summary of the image similarity metrics for the 2D input features.

| | NRMSE | PSNR | SSIM |
|---|---|---|---|
| Combined images | 0.071 | 23.598 | 0.901 |
| HUTUBS depth maps | 0.019 | 37.163 | 0.988 |
| HUTUBS edge features | 0.958 | 19.905 | 0.9 |

It is possible to notice how extending the pinna image training set with real-world photographs caused a decrease in reconstruction performances. Nevertheless,

both the model trained on the depth maps and the one trained on edge features show a satisfactory level of coherence. However, the similarity metrics once again portray a different picture, as indicated by the fact that the edge features training seems to score worst.

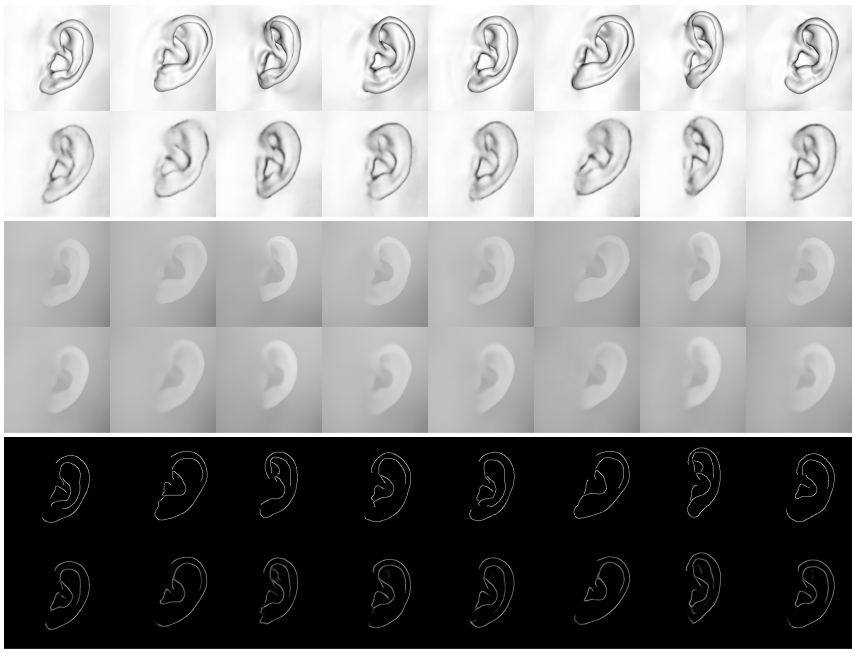

**Figure 4.4:** Comparison of different 2D features reconstructed by **InceptionVAE**. Each column corresponds to a test set pinna (frontal direction only). The pairs of rows show input and reconstructed data on the combined dataset, depth maps, and extracted edges, respectively.

Another way of evaluating the performance of the models is by assessing whether the latent dimensions extracted by the encoders correlate with the anthropometrics measurements available in the dataset. This idea follows the assumption that, since the latter are known to correlate with HRTF spectral features, by transitive relation, for any latent representation to be a good predictor of said spectral features, it must also correlate with the anthropometric measurements. The test set data was fed into the encoder sub-network and the distribution mean output layer was used for the value of $z$. During training, the $z$ vector is subject to a sampling process where a random variable $\varepsilon \sim \mathcal{N}(0, 1)$ is parametrized according to the mean and log-variance predicted by the encoder. However, during testing and production, $\varepsilon$ is set to 0, which is equivalent to discarding the $\sigma$ output and only using $\mu$ as the latent vector.

In order to verify the presence of such correlation, each latent vector was paired with the anthropometric data — included in the HUTUBS dataset — from its corresponding pinna image, and the Pearson coefficient was computed for each pair of feature, yielding a correlation matrix such as the one shown in Figure 4.5.

From the figure, it is evident that while there exists some correlation between the anthropometric data and $z$, the mapping is completely arbitrary, with some of the latent dimensions expressing no significant correlation. Furthermore, the feature that correlated the most with the latent dimensions was the viewpoint pitch angle in the rendering (labeled as "el" in the figure), followed by the pinna height, the fossa height, and the pinna rotation angle. Thus, a great portion of the latent space was spent encoding the dataset augmentations.

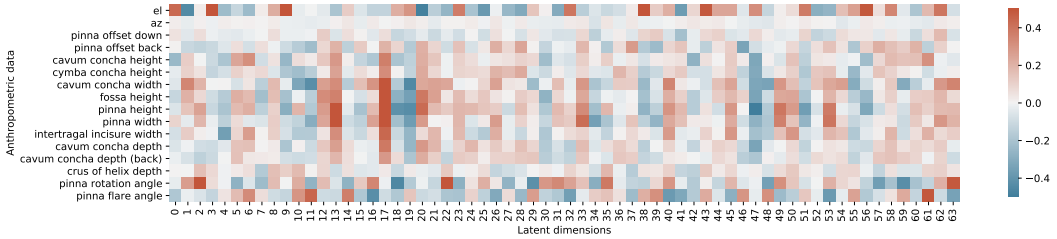


**Figure 4.5:** Pearson correlation coefficient (color) between anthropometric data (y-axis) and latent dimensions (x-axis), calculated on the VAE trained on pinna edges.

Finally, it was deemed relevant to inspect the topology of the latent space. Different pairs of latent dimensions were used as the $x$ and $y$ coordinates of a scatterplot, where each point corresponds to an input image. The points were then colored according to the labels derived earlier, consisting of the anthropometric data and the rendered dataset augmentations. Figure 4.6 exemplifies such visualization. In this case, by observing Figure 4.5, two anthropometric measurements were picked, and the latent dimensions correlating the most with them used as $x$ and $y$ coordinates. Although not entirely linear, the distribution of points expressing different characteristics is quite noticeable.

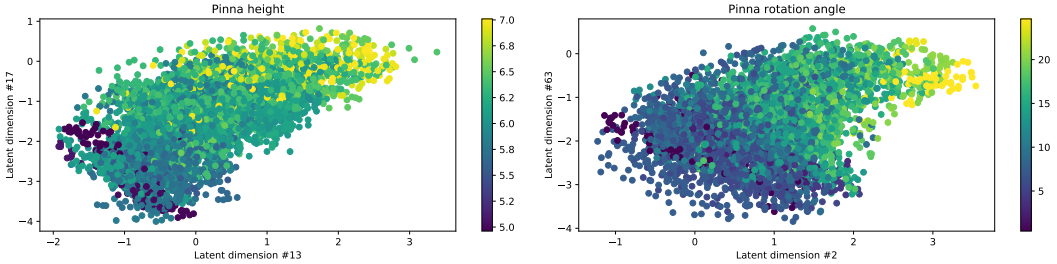


**Figure 4.6:** Example of encoded data points, calculated on the VAE trained on pinna edges, arranged across a pair of latent space dimensions, and colored according to a relevant feature (titles).

# Chapter 5

# Encoding of HRTF

Similar to what was done in the previous chapter, the second part of this project consisted in autoencoding the HRTF magnitude responses. As mentioned earlier, novel output data — in this case, a customized magnitude response — can be synthesized by sampling the latent space of a trained VAE. HRTFs, however, depend not only on the individual characteristics of the users expressed by their latent parameters but also on the spatial coordinate of interest. Therefore, the distribution $p(x|z)$ approximated by the VAE decoder must be conditioned by said spatial coordinates. This is achieved by using a variant of the VAE called *conditional variational autoencoder* (CVAE) [69].

CVAEs are a class of generative models analogous to VAEs but trained in a supervised fashion: the encoder learns a probability distribution $p(z|x, c)$ where $c$ is a vector of data labels, while the decoder learns the distribution of the input data based on the latent variables $z$ and the labels $c$, i.e. $p(x|z, c)$.

When used with categorical labels, CVAEs can generate unseen yet coherent data belonging to a given class $c$. This is similar to the deep learning architecture presented in [82], where $c$ consisted of subject IDs and spatial coordinates, both provided as one-hot encoded vectors. However, for this project, it was deemed beneficial to use a continuous representation of the spatial orientations, which allows for the synthesis of HRTFs at arbitrary spatial locations.

## 5.1 Proposed architecture

Since previous attempts at leveraging the spatial hierarchy of HRTF magnitudes using 1D convolutional layers proved ineffective [51], a fully-connected architecture was employed. Thus, the CVAE network implemented herein comprises a customizable number of dense layers with ReLU activation. The final encoder layers, tasked with predicting the mean and log-variance of $z$, employ dense layers without any activation function; this is because clamping the data into an arbitrary range may

degrade the network convergence. The decoder network follows a roughly mirrors the topology of the encoder, with its last layer also employing a linear activation for analogous reasons.

The hyperparameters of the model are its input size (i.e. the number of real-valued HRTF frequency bins), the number and size of the hidden layers in the encoder and decoder, the dimensionality of the latent space, and the conditioning labels constituting the vector $c$. To condition both the encoder and the decoder, $c$ is concatenated to the HRTF magnitude response and to the randomly sampled latent vector, respectively.

The input used by the models is composed of pairs of HRTF magnitude responses and data labels. Since the SOFA specification (described in subsection 3.2.1) requires HRTF to be stored as time-domain impulse responses, a pre-processing pipeline proved necessary. Specifically, individual HRIRs are extracted from a SOFA file, along with their respective spatial coordinates consisting of azimuth and elevation angles. Subsequently, they are converted to the frequency domain using an $n$-point discrete Fourier Transform optimized for real-valued signals. The number of frequency bins $n_{\text{fft}}$ is a customizable hyperparameter of the system, and yields a complex-valued spectrum comprising $\lfloor n_{\text{fft}} \rfloor + 1$ real-valued frequency bins. The resulting frequency response is then converted to a magnitude response, expressed in logarithmic units, and clipped to a dynamic range of 120 dB.

An optional processing step was also implemented and applied before the logarithmic conversion, allowing for the extraction of the spectral envelope, roughly corresponding to the HRTF peaks. This feature was implemented by finding all the peaks in the frequency-domain signal and applying a cubing interpolation between them. Alternatively, the difference between the original response and the spectral peaks could be extracted, approximately corresponding to the spectral notches. Unfortunately, however, HRTFs processed with this step proved particularly difficult to train with, supposedly due to the smoothening effect of the spectral envelope, which introduces noise in the extracted notches. Thus, the results gathered on these featured are not included below.

During training, individual HRIRs were extracted from the HUTUBS dataset, pre-processed using the aforementioned pipeline, and randomly bundled into mini-batches. The test set comprised the same four subject kept aside during the pinna images autoencoding, while the validation set used during training was composed of a random 20 % subset of all other HRTFs.

Similarly to the pinna images autoencoding described in the previous chapter, the Adam optimizer was used to update the trainable parameters of the model, with learning rate scheduler and early stopping mechanism configured analogously, and an initial learning rate of $1 \times 10^{-5}$. In this case, the model was allowed to train for up to 5000 epochs. The following section describes the results of experiments performed with the aforementioned CVAE.

## 5.2 Results

The model described in the previous section was trained on two variations of the HUTUBS dataset: one containing HRTFs across the entire spatial grid and one comprising only the median plane data. Indeed, it was hypothesized that training with a larger dataset would improve the generalization capabilities of the model, thus avoiding overfitting. This is, however, at odds with the fact that only a subset of the latent space is going to be used in the final HRTF individualization pipeline, which only requires a median plane response. Nevertheless, a model trained over the entire spatial grid may be used to generate a complete PRTF response.

**Table 5.1:** Summary of the model hyperparameters.

| Hyperparameter | Dense CVAE |
|---|---|
| Decoder layer sizes | [32, 64, 128, 256] |
| Encoder layer sizes | [256, 128, 64, 32] |
| Labels | "el", "az" |
| Latent size | 32 |

The CVAE trained on the full-grid dataset for 4626 epochs over the course of 42 hours; training on the median plane data took 4950 epochs over the course of 3 and a half hours. In both instances, the training set was divided into batches of 256 samples each, and the network hyperparameters were assigned according to Table 5.1. Each model comprised 222 000 trainable parameters. Similar to the previous chapter, the training was performed on dedicated hardware, and metrics such as cost function and learning rate were collected and visualized in realtime. Furthermore, a subset of the validation set comprising 24 randomly chosen HRTF magnitude responses was fed into the network every 50 epochs, in order to continuously monitor its reconstruction performances.

Several further training schemes were attempted, involving a lower number of frequency bins and a conditioning vector $c$ composed of elevation data only. However, the former parameter caused a considerable drop in performance. Similarly, the latter setup made it impossible to distinguish data points belonging to the frontal or posterior halves of the median plane. Therefore, these combinations of parameters were discarded in favor of those in Table 5.1. The trained models were evaluated according to the *spectral distortion* (SD), which is often used in the relevant literature [13, 56, 71] and is calculated as:

$$\mathrm{SD}_{\mathrm{dB}}(H, \hat{H}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(20\log_{10}\frac{|H(f_i)|}{|\hat{H}(f_i)|}\right)^2}$$

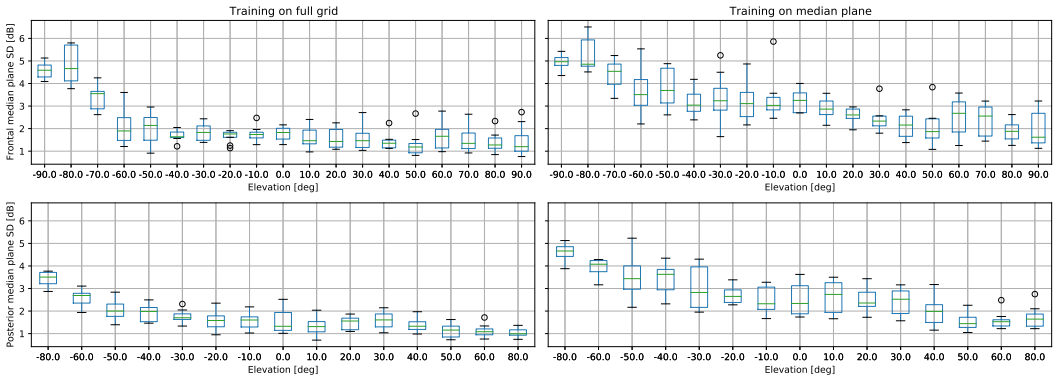


**Figure 5.1:** Box plot of the spectral distortion over a range of elevations, calculated for the two halves of the median plane (rows) on the test set using models trained on the full spatial grid or the median plane only (columns).

where $H$ is the original HRTF magnitude response, $\hat{H}$ is the predicted one, $N$ is the number of frequency bins considered, and $f_i$ is a given frequency bin.

Since changes in overall energy level do not affect localization along the vertical direction, any spurious offset between the magnitude responses of the two HRTFs under examination must be removed. This is achieved by minimizing the SD over a range of offsets applied on the predicted HRTF, using a simple grid-search algorithm. The SD values for the trainings — shown in Figure 5.1 — are based on a frequency range of 500 Hz to 16 kHz and an offset range of $-5$ dB to 5 dB. The figure is based on data from the median plane, which comprises 34 HRTFs per pinna, for a total of 272 data points.

One of the main disadvantages of adopting a vertical-polar coordinate system is the discontinuity in the coordinate values occurring at 90° (above the head) and at $-90°$. Indeed, the CVAE network cannot use the elevation label alone to determine which half of the median plane is the current HRTF belonging to, while the azimuth label jumps abruptly from 0° to 180°. To verify whether a continuous representation of the elevation coordinate would benefit the model performances, another set of training was performed, on the same two datasets, but with HRTFs labeled according to an interaural-polar coordinate system.

Again, the CVAE was trained on the complete HUTUBS dataset for 4300 epochs over 39 hours and on the median plane data for 7084 epochs over 3 hours. The results of this training are shown in Figure 5.2.

Table 5.2 shows the mean spectral distortion of all four trained models, which represents the theoretical maximum performance achievable by the entire system. Indeed, inevitable $z_{\text{hrtf}}$ prediction errors introduced by the neural network presented in the next chapter will cause the overall performance to fall below these values, which assume a perfect prediction of the HRTF latent representations.

**Table 5.2:** Summary of the model performances for each training strategy.

| Model training | SD [dB] |
|---|---|
| Full grid | 1.91 |
| Median | 2.93 |
| Full grid (interaural) | 1.84 |
| Median (interaural) | 2.73 |

From the table, it is possible to notice that while there is a slight increase in performance from vertical-polar to interaural-polar systems, the amount of training data accounts for most of the difference in spectral distortion. Thus, both the hypotheses mentioned above proved correct.

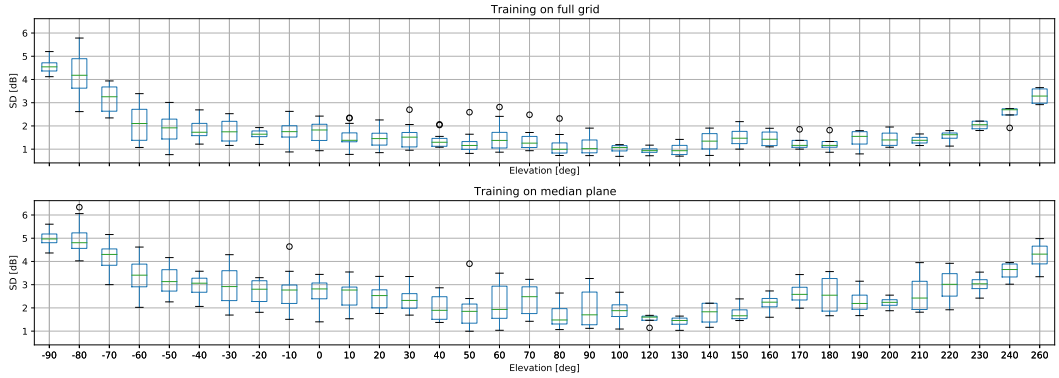


**Figure 5.2:** Box plot of the spectral distortion over a range of elevations, calculated for the median plane (expressed in interaural-polar coordinates) on the test set using models trained on the full spatial grid or on the median plane only.

Similarly to in section 4.2, the correlation between anthropometric parameters of the pinnae and the latent dimensions was computed and visualized as a correlation matrix in Figure 5.3. In this case, there s to be very little correlation between the two. This may indicate that the task of predicting the latent representation of the HRTFs from that of the 2D pinna features is not entirely feasible. Alternatively, it may indicate that $z_{\text{hrtf}}$ is encoded in such a way that does not induce a monotonic relationship with $z_{\text{ear}}$ — i.e. it follows a different but hopefully predictable pattern.

Finally, Figure 5.4 shows a random selection of predicted HRTFs from an unseen test subject, using both the best (full grid, interaural coordinates) and worst (median, vertical coordinates) performing models. The figure shows how, in both cases, synthesized spectral notches lack depth, although the full grid model can quite faithfully trace the response contour.

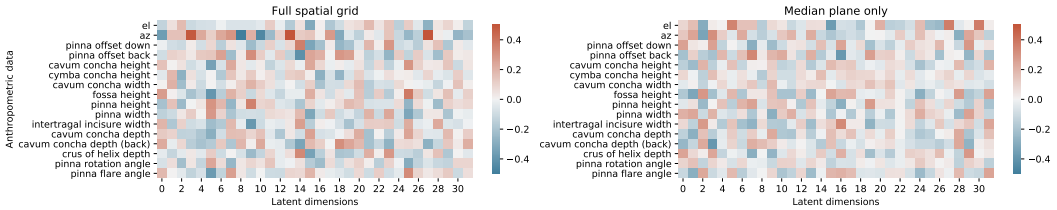


**Figure 5.3:** Pearson correlation coefficient (color) between anthropometric data (y-axis) and latent dimensions (x-axis), calculated on the CVAE based on vertical-polar coordinates (subplots).

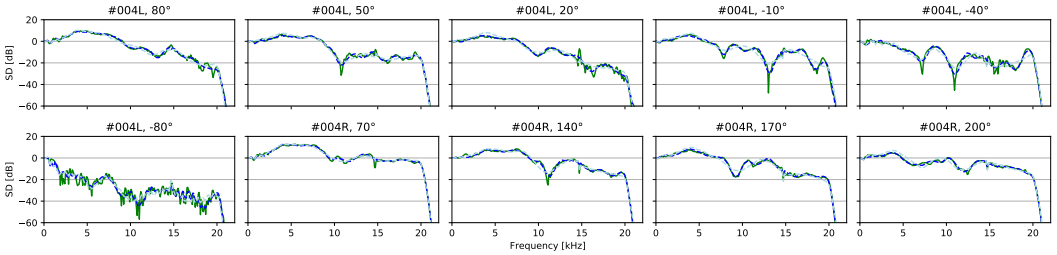


**Figure 5.4:** Example predicted HRTFs from an unseen subject. In solid green: input data; in dashed blue: prediction from *Full grid (interaural)* model; in dashed light blue: prediction from *Median* model.

# Chapter 6

# Prediction of encoded representations

The previous two chapters were concerned with the conversion of pinna images and HRTFs to and from an encoded representation, respectively. The current chapter covers the translation of the encoded representation of the pinnae 2D features $z_{\text{ear}}$ into the HRTF encoded representation $z_{\text{hrtf}}$. This can be formulated as a supervised learning task where $z_{\text{ear}}$ constitutes the input data and $z_{\text{hrtf}}$ is the target. An artificial neural network has therefore been designed and implemented to perform this task, in the form of a deep MLP.

The prediction of the HRTF latent representation is believed to be possible because, to faithfully reconstruct its input, the pinna images VAE must encode information pertaining to the individual morphology of the pinnae within its latent dimensions. According to widely established existing literature [1, 29, 77], the anthropometric parameters of the pinnae are known to affect their frequency response, although their exact impact is an open topic of inquiry. While manually extracted anthropometric data such as those available in the CIPIC and HUTUBS datasets are arbitrarily chosen and prone to systematic measurement biases, in this work it is hypothesized that automatically derived ones may offer additional insight.

The following sections describe the network architecture and its training process, as well as answer the research question posed above, using objective and perceptual metrics.

## 6.1 Proposed architecture

As mentioned above, a multilayer perceptron architecture — generally called deep neural network (DNN) — was implemented. In chapter 5, the spatial coordinates of a given HRTF were used to condition the CVAE encoder and decoder, to be able to sample points in the latent space belonging to the given spatial orientation. However, this external conditioning is not sufficient to ensure the complete dis-

entanglement between the latent representation of a given HRTF and its spatial coordinates.

Indeed, a preliminary PCA analysis of the CVAE latent vectors during the first few training epochs showed how the majority of the variance was explained by the elevation and azimuth of a given HRTF. Although a decrease of their impact was observed during subsequent training epoch, HRTFs belonging to the same subject did not map exactly onto the same latent space point, proving that their spatial coordinates still played a role in their latent encoding. To address the aforementioned phenomenon, the azimuth and elevation labels $c$ must be accounted for in the training set.

The resulting model is therefore composed of an input layer taking the $z_{\text{ear}}$ vector plus a spatial coordinate vector, a number of hidden fully-connected layers with leaky ReLU activation, and an optional dropout layer for regularization. The number of hidden layers, their sizes, and the dropout rate are all hyperparameters of the system. The size of the input and output layers are also user-defined and correspond to the size of the latent spaces for pinna images and HRTFs, respectively.

To train this model, a pair of input data and target output is needed for each HRTF of each subject. Both inputs and outputs are extracted from the pinna images VAE and HRTFs CVAE, respectively, using the encoders of each model. The resulting latent vectors are then stored along with labels comprising metadata such as the subject ID, the pinna under consideration (left or right), and spatial coordinates — this only applies to $z_{\text{hrtf}}$. This information was later used during training to associate each target vector $z_{\text{hrtf}}$ with its respective input $z_{\text{ear}}$ and spatial coordinates vector.

## 6.2 Results

For this model, several datasets were tested, differing in the content of the target HRTF latent data and input features. The former included all the $z_{\text{hrtf}}$ vectors extracted from the CVAE trained across the entire spatial grid, those extracted from the same model but filtered by median plane only, and median plane $z_{\text{hrtf}}$ extracted from the CVAE trained on the median plane only. The first training procedures reflected the intuition mentioned in section 5.2 that using more data prevents the model from overfitting, whereas the last two accounted for the need of using the deep learning sub-system for generating median-plane PRTFs only, thus focusing on that subset of data. Regarding the input features used as predictors, the choice included the latent representations extracted from the pinna images, depth maps, and contours, using the **InceptionVAE** model described in chapter 4.

The training was performed according to the same procedure and using the same hyperparameters mentioned in the previous chapter except for the number of epochs, set to 5000 for the training on the complete spatial grid and 10 000 for

the others, the learning rate set to $1 \times 10^{-3}$, and early stopping patience set to 30. The model was parameterized according to Table 6.1. Furthermore, to better detect potential overfitting, the validation set was constructed by randomly selecting 10 subjects from the training set and setting them aside. Each of the 18 initial trainings took anywhere between 3 to 20 minutes, usually converging within 30 to 150 epochs.

**Table 6.1:** Summary of the model hyperparameters.

| Hyperparameter | DNN |
|---|---|
| Hidden layers | [256, 256, 128, 64] |
| $z_{\text{ear}}$ size | 64 |
| $z_{\text{hrtf}}$ size | 32 |
| Labels | "el", "az" |
| Dropout rate | 0.4 |

After training all the different versions of the DNN, their average spectral distortion was computed for HRTFs along the median plane. The specifications of each training procedure along with its resulting SD is summarized in Table 6.2, while Figure 6.1 compares the SD of predicted HRTFs for models trained on the entire spatial grid, on the basis of the input features used as predictors. The SD is computed by feeding the pinna 2D features into the VAE encoder, cascaded with each respective DNN and VAE decoder, which are picked according to the training strategies mentioned above — see the green data flow in Figure 3.2. Thus, these values include the contribution of the other networks and reflect the overall performances of the deep learning sub-system. Both the box plot and the table show a clear degradation of the reconstruction performances for its theoretical maximum, calculated in the previous chapter.

**Table 6.2:** Summary of the SD for the first 18 trainings, sorted by dataset (rows) and input features (columns).

| | Depth maps | Edges | Images |
|---|---|---|---|
| Full grid | 5.14 | 6.34 | 4.92 |
| Full grid (m.p. only) | 4.89 | 4.86 | 4.86 |
| Median plane | 4.71 | 4.77 | 4.74 |

Besides the training scheme mentioned earlier, two other alternative approaches were tested. One involved using the latent representation of the HRTFs encoded using an interaural-polar system for spatial coordinates instead of the conventional vertical-polar one. As mentioned in the previous chapters, this system provides a

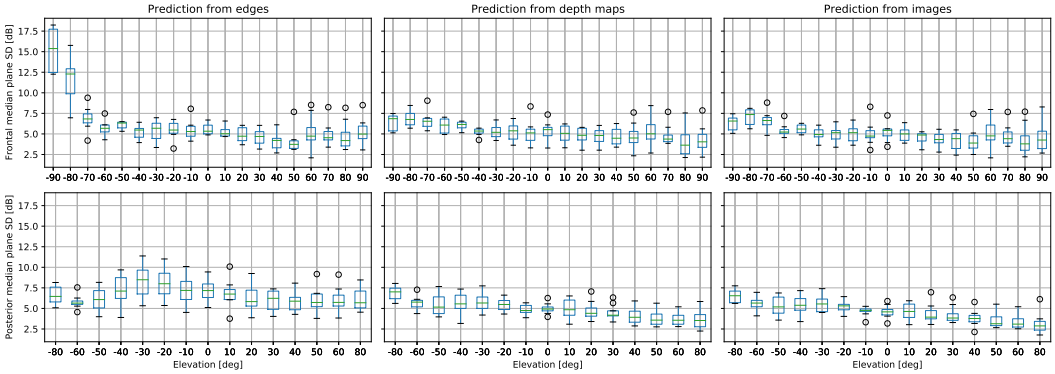


**Figure 6.1:** Box plot of the spectral distortion over a range of elevations, calculated for the median plane (rows) on the test set using models trained on the full spatial grid using different predictors (columns).

continuous monotonic change of elevation across the entire median plane, which was thought to help the CVAE in modeling the relationship between the frontal and posterior halves of the median plane. Models trained according to this strategy were evaluated using their respective CVAE decoders. Figure 6.2 offers a comparison between this and the previous training, showing that in most cases the model performances dropped.

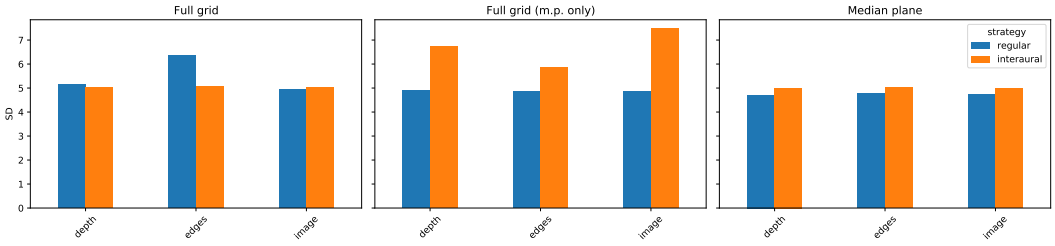


**Figure 6.2:** Comparison between regular training and interaural-polar coordinate system (colors), for a range of input features (x-axis) and datasets (subplots).

The last approach consisted in predicting a subset of the principal components of the HRTFs latent vectors instead of their raw values. PCA constructs orthogonal — i.e. mutually uncorrelated — linear combinations of the input features that successively explain as much variance as possible. The rationale behind this was to simplify the regression task by reducing the number of prediction targets and projecting the output data onto a more interpretable set of basis. In this case, the PCA loading and score matrices were calculated on the training set alone, and the number of principal components became a hyperparameter of the system. The number of PCA components was set to 8, which according to a preliminary analysis, could explain up to 95 % of the variance in $z_{\text{hrtf}}$. Since the HRTFs CVAE decoder

performs arbitrary weighting and scaling of its input data, the latent vectors were normalized before applying the PCA by subtracting their mean and dividing by their variance.

Once the DNN was trained with all the combinations of input features and datasets, it was possible to compare the efficacy of all the various strategies. These are shown in Table 6.3 and in Figure 6.3. Both the figure and the table show similar performances across the entire range of training schemes, with a slight advantage for the regular procedure using vertical-polar coordinates. Furthermore, the SD seems to be unaffected by the type of input feature used, while the best results are generally achieved by training on the median plane dataset and using the CVAE decoder trained on the median plane only. This final point proved unexpected because, according to the SD values extracted in the previous chapter, this latter model was the worst-performing one.

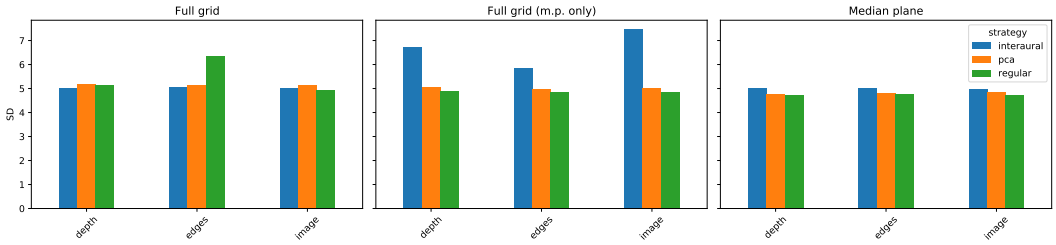


**Figure 6.3:** Comparison between the three training strategies (colors), for a range of input features (x-axis) and datasets (subplots).

**Table 6.3:** Summary of the performances of the various models, divided by training strategy (rows) and input features (columns).

| Strategy | Dataset | Depth maps | Edges | Images |
|---|---|---|---|---|
| Regular | Full grid | 5.14 | 6.34 | 4.92 |
| | Full grid (m.p. only) | 4.89 | 4.86 | 4.86 |
| | Median plane | 4.71 | 4.77 | 4.74 |
| Interaural | Full grid | 5.01 | 5.08 | 5.04 |
| | Full grid (m.p. only) | 6.71 | 5.86 | 7.47 |
| | Median plane | 5.00 | 5.01 | 4.97 |
| PCA | Full grid | 5.17 | 5.17 | 5.16 |
| | Full grid (m.p. only) | 5.06 | 4.98 | 5.02 |
| | Median plane | 4.77 | 4.81 | 4.84 |

Finally, Figure 6.4 shows a random selection of predicted HRTFs from an unseen test subject, synthesized with models from the first three rows of Table 6.3. The

figure shows how the spectral notches are particularly shallow, although the general contour of the response is synthesized faithfully.

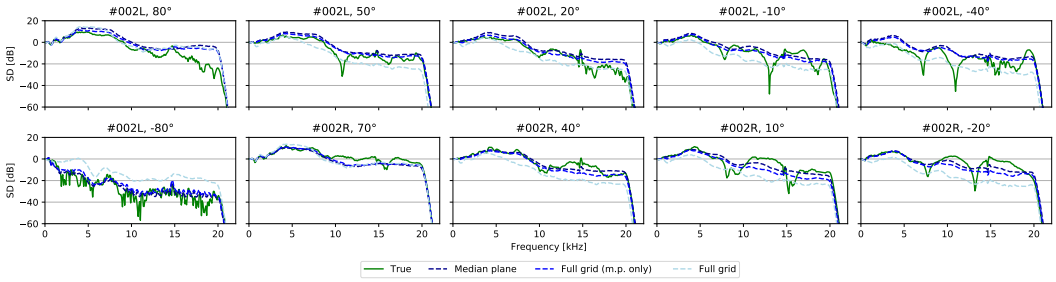


**Figure 6.4:** Example predicted HRTFs from an unseen subject, comparing the original HRTF magnitude response (solid green) with the predictions from DNN models — and their respective CVAE decoders — trained on three dataset variations (dotted shades of blue), using the regular strategy.

# Chapter 7

# HRTF set creation

The final part of the solution consists in using the pinna response synthesized through the previous steps to generate an entire HRTF set. While the system described in the previous chapters is capable of generating pinna responses over a spatial grid of different azimuths and elevations, HRTFs notches and other spectral features are known to be relatively stable across the horizontal direction [47], so only the responses along the median plane are extracted and used, limiting their impact to monaural cues only. This is in line with the fact that pinna images alone do not contain the necessary predictors for deriving binaural cues, which are known to be related to the anthropometry of the head. For these reasons, spectral features and binaural cues that are not caused by the pinnae must be accounted for separately.

The structural model presented herein identifies and integrates two external contributions: the effect of head, torso, and shoulders, and the interaural time difference. The former is derived from one of the VIKING subjects consisting of a KEMAR mannequin with its original pinnae removed and the slots filled with a silicone baffle, resulting in the HRTFs of a pinna-less subject [73] as seen in Figure 7.1. The spectral features introduced in this way are only a coarse approximation of the effect of shoulders and torso. However, these are thought to provide localization cues at frequencies below 3 kHz, which are only crucial when localizing narrow-band sounds under that threshold [1]. Furthermore, an accurate shoulder-reflections model must account for the relative angle between the torso and the head, thus requiring inertial measurement units or other head-tracking technology.

The ITD, on the other hand, is extracted from a HUTUBS subject and applied to the HRIRs of the generated set. The relevant HUTUBS subject is chosen using the HRTF selection algorithm presented in [70], where three anthropometric parameters — corresponding to head width, head depth, and shoulder circumference — are used as features of a linear regression model predicting a horizontal localization

error metric. The metric is computed for all HUTUBS subjects, and the one minimizing the error is selected. Since the spatial grid employed by the VIKING dataset differs from the HUTUBS one, an interpolation algorithm to convert the former to the latter was implemented accordingly. The choice of adopting the spatial grid of the HUTUBS dataset was mostly arbitrary, although it simplifies the comparison between its subjects and an individualized HRTF.

Finally, it is important to remember that only the magnitude of the pinna responses are synthesized in the previous steps, meaning that any phase information is missing. Thus, the pinna contribution is applied to the pinna-less HRIRs as a minimum-phase IIR filter constructed to match the magnitude response as close as possible, derived using the Yule-Walker method [23] of sufficiently large order.

A detailed explanation of the HRTF set creation algorithm can be found in the sections below, along with a description of its performances.

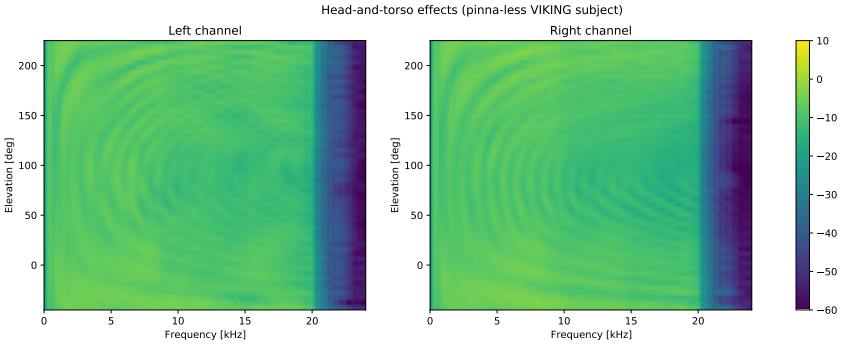


**Figure 7.1:** Median plane response for the VIKING pinna-less subject Z, used to model head, shoulder, and torso effects.

## 7.1 Proposed algorithm

The HRTF set generation algorithm has been implemented using MATLAB, and comprises the following five building blocks:

- `select_ITD.m`: a function implementing the ITD-based selection algorithm described earlier, taking head width, head depth, and shoulders circumference as arguments, selecting the most viable HUTUBS subject, and returning a list of left and right ears onset delays with their respective spatial coordinates. The ITD estimation algorithm used to extract the delays is part of the Auditory Modeling Toolbox[1], implemented from [3].

- `interpolateDelays.m`: this function interpolates the onset delay times derived above, so as to match the HUTUBS spatial grid. The vertical resolution

[1] `http://amtoolbox.sourceforge.net/`

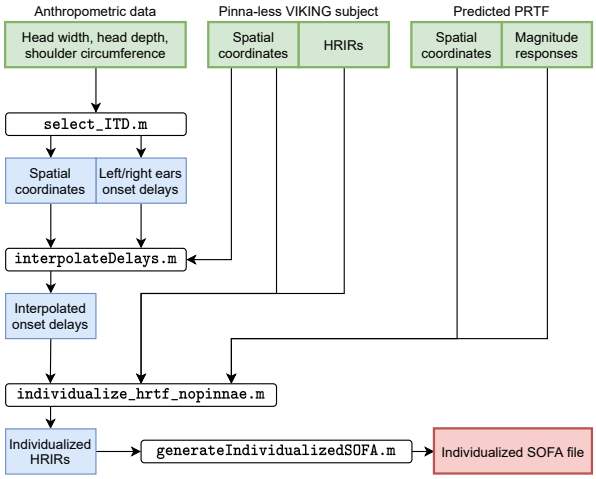


**Figure 7.2:** Block diagram of the HRTF set generation process. Green boxes represent input data, blue boxes represent intermediate outputs, and the red box is the final output.

of the VIKING dataset is exactly twice that of HUTUBS, so a simple 1D linear interpolation was employed. The function takes a list of HRTF onset delays with their respective spatial coordinates and a list of new spatial coordinates, and for each elevation angle, it extracts the old and new coordinates and uses them as arguments for MATLAB's `interp1q` function.

- `individualize_hrtf_nopinnae.m`: this is the function performing the actual individualization steps. Its inputs are a set of frequency-domain pinna magnitude responses, a set of time-domain impulse responses for torso, head, and shoulders — both together with their respective spatial coordinates, and a list of interpolated delays. For each unique elevation in the desired final spatial grid (represented by the list of spatial coordinates for each pinna-less HRIR) two IIR filters are constructed, one for the front half and one for the rear half of the median plane, using MATLAB's `yulewalk` function with an order of 60. Subsequently, each left-right pair of HRIRs belonging to the current elevation is processed using one of the filters, with the front filter being used for azimuths between 90° and 270° and the rear one for all other cases. Finally, the pre-existing onset delay is calculated using the `itdestimator` mentioned in the first script, and the difference between it and the selected delays in samples is used to zero-pad or crop the HRIRs. Since the amplitude of the HRIRs is very close to 0 the at either end due to windowing [75], a fast circular shifting algorithm is used, thus easily preserving the original impulse length. This process is shown in Figure 7.3.

- `generateIndividualizedSOFA.m`: this function simply takes care of encoding

the HRIRs generated above into a SOFA file which can be read by compatible binaural rendering engines. The script fills all the mandatory metadata, and places the receivers in the right location using the head width anthropometric measurement.

- `test_nopinnae.m`: this script combines all of the previous steps and optionally visualizes the resulting HRTF set.

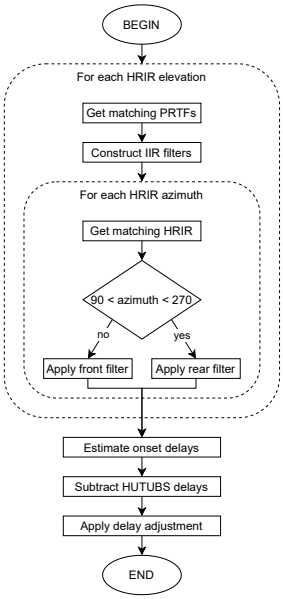


**Figure 7.3:** Flowchart of the HRTF individualization algorithm.

Thus, the inputs for this final part of the solution comprise the three anthropometric parameters of head and shoulders, the median plane pinna response generated using the deep learning models, and HRTFs of the pinna-less VIKING subject. Figure 7.2 illustrates how each script is involved in the complete HRTF set generation process, while Figure 7.3 presents the HRTF individualization algorithm.

## 7.2 Results

While the previous chapters assessed the performance of their respective building blocks by means of spectral distortion, this section evaluates the resulting HRTFs based on their localization performances.

Conducting an evaluation study on human subjects is perhaps the most faithful way of determining the suitability of an individualized HRTF set. This requires gathering the data shown in Figure 7.2, generating the HRTF set, and comparing the

vertical localization error between a generic HRTF and the individualized one over a range of spatial locations, similarly to [33] or [49]. Usually, such a test is performed in a controlled environment, using a pair of headphones whose response is known and can be accounted for, and with the help of personnel capable of assisting in gathering the anthropometric data. However, with the current restrictions of gatherings and restricted access to the university facilities, this proved unfeasible. Indeed, the factors which could not be controlled — such as headphones type, background noise level, and amount of user training with the task at hand — may have severely degraded the validity of the findings. Thus, a psychoacoustic median plane localization model was employed.

The localization model adopted here is described in [5] and comprises a spectral auditory processing block and a spatial mapping one. In the first block, the *directional transfer function* (DFT) is extracted from each of the subject's HRTFs, by subtracting direction-independent response (i.e. their average) of each ear. These are clustered into a number of sagittal planes, and convolved with an input stimulus. The output is then filtered using a 28-band gammatone filterbank approximating the human cochlear filtering. The last step of the auditory processing involves extracting the positive spectral gradients from each filter, similarly to how the dorsal cochlear nucleus is thought to work [63].

In the second part of the localization model, a comparison between the target representation and a template is used to derive a measure of distance, which is then mapped to the probability of a given response using a nonlinear transfer function. Subsequently, the weight of each ear is adjusted according to the sagittal plane under investigation and fed into a sensory-motor mapping function simulating the conversion between auditory perception and motor response. Finally, the resulting values, which represent the response probability for a given spatial direction, are scaled into a probability mass vector with its sum equal to one.

The probability mass vector computed using the localization model can then be used to derive two psychoacoustic performance parameters: the *quadrant error rate* (QE) and the *polar root-mean-square error* (PE). The former represents the fraction of responses within $\pm 90^\circ$ from the target angle, also called local responses, while the latter corresponds to the RMSE of the local responses. These psychoacoustic metrics, together with a visualization of the distribution of the predicted responses, are the tools used to evaluate the generated HRTFs. An implementation of the localization model and the aforementioned metrics is available in the Auditory Modeling Toolbox[2].

Due to their vast number, it was impractical to test every combination of deep learning models, therefore the following evaluation is limited to the most relevant cases. Specifically, models using $z_{\text{ear}}$ vectors derived from the pinna depth maps

[2] `http://amtoolbox.sourceforge.net/`

were excluded since these would be the hardest 2D features to obtain. Furthermore, only DNNs trained according to the regular strategy — i.e. no PCA decomposition or interaural-polar coordinates — were tested, as they almost consistently offer the best performances in comparison. This yielded the combinations summarized in Table 7.1.

**Table 7.1:** Model combinations used for the evaluation.

| VAE input features | DNN training strategy | DNN and CVAE training set |
|---|---|---|
| Edges | Regular | Full spatial grid |
| Images | Regular | Full spatial grid |
| Edges | Regular | Median plane only |
| Images | Regular | Median plane only |

The setups shown above were used to generate the PRTFs of each of the test set subjects, which were then converted into HRTF sets using the algorithm described in the previous section, and finally evaluated with the localization model. The measurements of head width, depth, and shoulder circumference are also used for the sake of completeness, although the binaural cues they introduce are not relevant for vertical localization.

The performances of the individualized HRTF sets are compared against a baseline generic HRTF set corresponding to the first HUTUBS subject, i.e. the FABIAN head-and-torso simulator [45]. The template used by the localization model corresponds to the measured HRTFs of the given test subject, the input stimulus is an impulse, and the model is evaluated over a vertical grid of 10°.

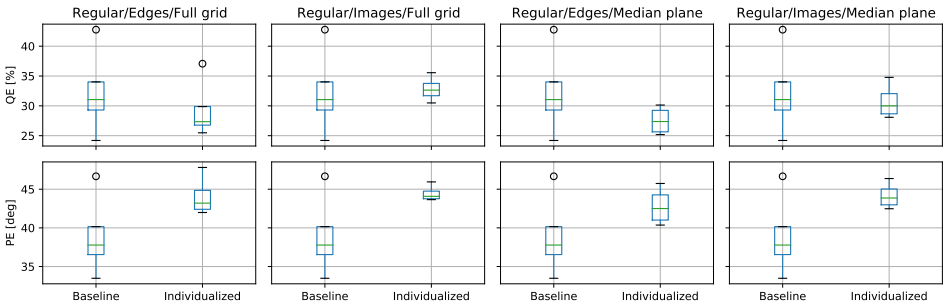


**Figure 7.4:** Box plot of the psychoacoustic metrics (rows) computed using the localization model, for each of the four chosen model combinations (columns).

An overview of the psychoacoustic metrics can be seen in Figure 7.4. According to the plot, most of the models reduce the quadrant error rate by a small amount, up to 5 %. However, the polar error seems to be increasing by up $\sim 5^\circ$ on average. Most notably, the standard deviation of the metrics for the individualized HRTFs

is significantly lower than the baseline's, indicating a more consistent behavior across the test set. Figure 7.5 exemplifies these results, showing how the predicted responses are focused around the listener's zenith, while front-back inversion still occurs — although less prominently — for frontal sound sources. The following chapter attempts to explain some of the patterns observed in the result sections.

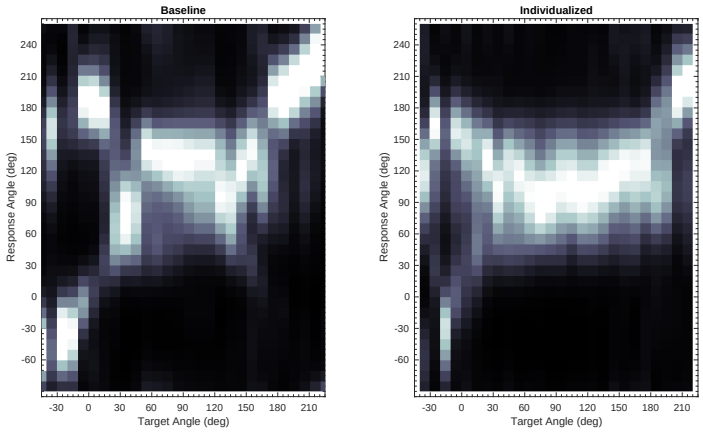


**Figure 7.5:** Plot showing the probability distribution (color) of the predicted responses (y-axis) against the target angle (x-axis).

# Chapter 8

# Discussion

The VAE models employed for extracting the compressed representation $z_{\text{ear}}$ from 2D images seem to be capable of successfully reconstructing their input. However, this is only true for trainings performed on homogenous datasets such as with the HUTUBS renderings, depth maps, or edge features. Indeed, the visual fidelity of **InceptionVAE** trained on the combined dataset of renderings and real-world images was much lower than its counterparts (see Figure 4.4). Furthermore, the reconstruction performances are in part dependent on the decoder sub-network, which is not needed in the final pipeline.

As seen in section 4.2, the characteristic expressing the most correlation with the latent dimensions was the camera pitch angle within the rendering scene, which was part of the dataset augmentations applied to increase its size. Since this characteristic is irrelevant to the task at hand, this strategy might not have been a desirable one, although it is thought to have improved the generalization abilities of the networks.

Due to the lack of matching anthropometric measurements, it is not possible to derive a similar correlation matrix for the data points in the combined dataset of HUTUBS grayscale renderings and real-world pinna images. Thus, it is impossible to determine whether a given latent dimension equally affects the rendered and real-world data. Should this not be the case, meaning that the real-world images are encoded in a separate and non-overlapping subspace — or in other words, they are described primarily by a different set of spatial dimensions — its pre-trained weight would prove completely unusable when employed with a user pinna image as part of the DL sub-system. This argument supports the adoption of simpler edge features, which are more resilient to differences in lighting conditions, skin tone, presence of hair, or other characteristics that the VAE model may encode but that offer no predictive power.

While considering the CVAE whose decoder constitutes the last stage of the deep learning sub-system, it is important to notice that the model is trained on

HRTF magnitude responses. Therefore, it is sensible to assume that, amongst its compressed representations, there may be latent dimensions responsible for imparting the effect of head, shoulders, and torso. However, the DNN input consists only of spatial coordinates and $z_{\text{ear}}$ vectors, which are not trustworthy predictors of the head, shoulders, and torso impact. This is why the output of the CVAE decoder has been referred to as PRTF despite originating from the HRTF data. In order to discard the influence of head, shoulders, and torso and isolate the pinna response, one could derive a pinna-less response such as the one in Figure 7.1 and subtract it from the HRTF. However, this only applies to datasets of non-human subject, where the anatomy of the mannequin is maintained constant.

Unfortunately, as seen in section 4.2, there is no guarantee that any of these latent factors are mutually orthogonal, and any given spectral cue constituting the HRTFs may be embedded within multiple latent dimensions. As seen earlier, one of the main challenges with using variational autoencoders is the interpretability of their representation. Indeed, a given latent variable may account for several observable or perceptual factors at once or provide no discernable contribution to the encoding. A common technique known as $\beta$-VAE [30] addresses this issue by introducing a scaling factor to the KL-divergence term of the loss function. By increasing its weight, the VAE encoder is forced to closely match the prior latent distribution $p(z)$, thereby promoting a more disentangled internal representation. This, however, comes with a trade-off in reconstruction performances. When parameterizing the autoencoder models used in this project, the $\beta$ hyperparameter — shown as *KL coefficient* in Table 4.1 — was instead deliberately set to a small value in order to promote a more faithful reconstruction. A recently developed extension of $\beta$-VAE [11] suggests modulating the coefficient value during training, effectively training the model like a conventional autoencoder and later increasing the impact of the KL-divergence, in order to retain both reconstruction performances and latent space disentanglement.

When observing the distribution of the SD over the range of elevations, a clear pattern emerges, whereby HRTFs at low elevations present the largest errors. This may be explained by the steep change in response for these HRTFs, clearly noticeable in the input HUTUBS data shown in Figure 3.3 and known to be caused by torso-induced damping [9]. Although the deep autoencoder used for the experiment should be capable of modeling this nonlinear relationship, the abrupt changes in response observed here may have degraded its reconstruction performances.

Another interesting result from the CVAE training is the similar performances — again measured in terms of spectral distortion — between models conditioned with a $c$ vector expressed in vertical-polar coordinates against those expressed in interaural-polar coordinates. According to the values in Table 5.2, the latter offer a decrease in SD of only $\sim$0.1 dB. Despite representing the elevation using a

continuous value appears to be intuitively advantageous, the CVAE model may be equally benefitting from the monotonic relationship between elevation angle and notch frequency provided by the vertical-polar coordinate system.

Surprisingly, there seems to be little difference in performances between DNNs trained using latent representations extracted from different 2D pinna features. Two alternative hypotheses for this phenomenon are advanced: in the worst case, the $z_{\text{ear}}$ are not particularly useful for the task at hand and the DNN minimizes its prediction error by assigning low weights and arbitrary biases to the feature inputs, thereby coalescing all input data points in approximately the same output. Alternatively, the features extracted by the pinna images VAE from each of the datasets are so similar that they happen to have almost the same prediction power. Feeding the DNN with random inputs and comparing the standard deviation of the predicted outputs with the standard deviation of the $z_{\text{hrtf}}$ targets shows a slight decrease in prediction variance, which might validate the first hypothesis. Unfortunately, understanding the behavior of the convolutional filters within a CNN is particularly difficult, thus neither case can be proven.

When coupled with the other parts of the system, the trend in performances of the CVAE models trained on the full spatial grid and those trained on the median plane only seem to reverse. If proven correct, the first hypothesis mentioned in the previous paragraph might also explain this phenomenon. Assuming that the latent representation of the complete HRTF dataset is more spread out than its median plane counterpart, it would be more difficult for the DNN to estimate its centroid without sacrificing the reconstruction performances of one of its subsets.

Another noticeable aspect of the DNN training is its brief duration. Most DNNs reached convergence within 30 to 150 epochs, including those trained on the full grid dataset comprising 36 080 training data points. A previous iteration of the DNN training procedure, where the train and validation sets were split on an individual data point basis (similarly to how described in section 5.1) instead of on a subject basis, managed to train for much longer, i.e. up to 7000 epochs. However, the models would soon start overfitting; this was not reflected in the ill-constructed validation set, and training would go on indefinitely. The issue was detected by independently monitoring the performances on the test set during training, which revealed a steep increase in SD. In order to further ameliorate the impact of overfitting, the dropout rate of the activations was increased to 40 %, although no significant difference was observed. Furthermore, when splitting the train and validation sets on a subject basis, random chance plays a more substantial role in determining the performances on the test set. Indeed, if subjects with similar characteristics as the ones in the test set are assigned to the validation set, the model will not learn from them and the overall performances will be lower than the opposite scenario. An obvious remedy for this issue consists in increasing the size of the dataset.

When considering the combination of synthesized, measured, and selected elements into an individualized HRTF set, a noticeable limitation consists in using only 2D features from one of the pinnae when extracting the PRTF, which is then used for both channels. Although vertical localization is thought to be a mostly monaural process, it is also known that both ears contribute equally to localization along the median plane [59]. Therefore, slight differences and asymmetries in pinna anatomy that are not conveyed by the individualized responses may degrade the localization performances.

Finally, a trend emerges when comparing listener angle responses generated using the localization model against the target angles: the predicted angle seems to cluster around subject's zenith, as shown by the flat white area at the center of Figure 7.5 (right plot). As indicated in Figure 3.3, spectral notches gradually softens for elevations around 90°. Thus, it is thought that the shallow spectral features synthesized by the deep learning sub-system are interpreted by the localization model as belonging to elevations above the head, causing much of the local prediction error.

# Chapter 9

# Conclusions

This project investigated the prediction of salient spectral localization cues occurring across the median plane from images of the user's pinnae using deep learning models. These pinna responses were then used to filter a generic shoulder reflection response measured acoustically over a spherical spatial grid. Subsequently, the ITD of the resulting HRTF set was manipulated to match that of a HUTUBS dataset subject with similar anthropometric parameters. Finally, the results were evaluated using objective metrics and a perceptual localization model.

The implemented system fulfills the initial requirement of generating customized HRTF sets from easily obtainable user data. However, the metrics considered in the evaluation are inconclusive regarding the efficacy of this structural model, and more investigation is needed. Nevertheless, it is believed that a thorough optimization of each of the building blocks of the pipeline may provide substantial performance gains.

Inspecting each building block of the solution yielded the following findings:

- The variational autoencoder is capable of extracting latent parameters from 2D pinna features. However, it is unsure whether they are good predictors of HRTF spectral characteristics, compared to conventional anthropometric measurements such as those available in several HRTF datasets. More user data would be needed, although training on larger and more heterogeneous datasets seem to degrade performances;
- The conditional VAE can effectively encode an entire HRTF dataset and behaves faithfully on unseen subjects, although the reconstructed notches are not particularly prominent;
- The deep neural network used for mapping the two latent representations is prone to overfitting due to the lack of training data where there is a correspondence between 2D pinna features and HRTFs. Furthermore, it may benefit from using additional predictors.

The HRTFs generated using the overall pipeline appear to vary smoothly across the spatial dimensions and are free from audible artifacts. One of the variants of the pipeline, based on pinna edge features, has been employed in a recent project by another SMC student, which aimed at evaluating different HRTF profiles within a gamified, interactive 3D environment. Within the environment, users were asked to shoot targets they believed to be the source of a wideband, buzzing sound. Based on the data collected in his project, the individualized HRTF improve the vertical localization error by 1° and the horizontal error 2.5° compared to the generic HRTF used as baseline in section 7.2, which proves promising despite the lackluster results shown by the psychoacoustic metrics.

If considered independently from the rest of the solution, the CVAE may find applications in compressing HRTFs or interpolating HRTF sets over a finer spatial grid, similarly to [6]. Its relatively low number of trainable parameters makes it suitable for embedded applications such as hearing aids or wireless earbuds. In fact, the memory footprint of the trained weights and latent vectors is considerably lower than the entire HUTUBS dataset, with the compression benefits compounding as more subjects are included. Furthermore, it could constitute the foundation of an alternative HRTF individualization approach, perhaps based on perceptual feedback from the user as in [82].

Each of the pipeline sub-systems presented herein could benefit from additional work. With regards to the pinna images VAE, given that faithful reconstruction is not necessarily a proxy for interpretable latent space encoding, it may be useful to choose a model architecture and set of input features based on the correlation between the latent space vectors extracted from each user's pinna image and its anthropometric measurements.

Alternatively, such a correlation might be enforced by adequately conditioning the loss function. Another way of promoting the emergence of useful predictors may consist in training each of the models involved in the DL sub-system (shown in Figure 3.2) at the same time, in such a way that the pinna VAE loss is dependent on the prediction performances of the DNN. However, such an interconnected system may prove particularly complex to implement and train.

A simpler version of the solution mentioned above may consist in simply implementing an ANN comprising only the networks and sub-networks used during evaluation, linked together and trained simultaneously so as to minimize the mean squared error of the original and predicted HRTFs. However, it is not known whether the dimensionality-reduction behavior of the autoencoders would manifest.

The lack of large-scale HRTF datasets containing rich anthropometric data such as head scans or pinna images is one of the main hindrances to the application of deep learning techniques. Recent efforts have been made with regards to generating arbitrarily sized synthetic PRTF sets [27], [17]. These are based on a dataset of

pinnae 3D scans, which are morphed using PCA and subsequently fed into an FM-BEM numerical simulation software to extract their response. Large synthetic datasets such as this one may improve the performances and generalization abilities of both the CVAE and the DNN sub-systems by providing more data to learn from.

Furthermore, some of the model hyperparameters were have been maintained constant across a range of trainings, in order to isolate the impact of other parameters and simplify the comparison. However, it may be desirable to investigate the impact of hyperparameters such as the number of latent dimensions of both $z_{\text{ear}}$ and $z_{\text{hrtf}}$ and the scaling coefficient applied to the KL divergence term.

Finally, the head-and-torso effect provided by the VIKING pinna-less subject could be adapted to reflect the anatomical differences between users, by means of frequency scaling parametrized according to a set of anthropometric measurements [53, 55].

# Bibliography


[1] V. Ralph Algazi, Carlos Avendano, and Richard O. Duda. "Elevation Localization and Head-Related Transfer Function Analysis at Low Frequencies". In: *The Journal of the Acoustical Society of America* 109.3 (Feb. 27, 2001), pp. 1110–1122. DOI: 10.1121/1.1349185.

[2] V.R. Algazi et al. "The CIPIC HRTF Database". In: *Proceedings of the 2001 IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics (Cat. No.01TH8575)*. New Platz, NY, USA: IEEE, 2001, pp. 99–102. DOI: 10.1109/ASPAA.2001.969552.

[3] Areti Andreopoulou and Brian F. G. Katz. "Identification of Perceptually Relevant Methods of Inter-Aural Time Difference Estimation". In: *The Journal of the Acoustical Society of America* 142.2 (Aug. 1, 2017), pp. 588–598. DOI: 10.1121/1.4996457.

[4] D. W. Batteau and Hugh Esmor Huxley. "The Role of the Pinna in Human Localization". In: *Proceedings of the Royal Society of London. Series B. Biological Sciences* 168.1011 (Aug. 15, 1967), pp. 158–180. DOI: 10.1098/rspb.1967.0058.

[5] Robert Baumgartner, Piotr Majdak, and Bernhard Laback. "Modeling Sound-Source Localization in Sagittal Planes for Human Listeners". In: *The Journal of the Acoustical Society of America* 136.2 (Aug. 2014), pp. 791–802. DOI: 10.1121/1.4887447.

[6] Sunil Bharitkar et al. "Stacked Autoencoder Based HRTF Synthesis from Sparse Data". In: *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. Honolulu, HI, USA: IEEE, Nov. 2018, pp. 356–361. DOI: 10.23919/APSIPA.2018.8659495.

[7] Jens Blauert. *Spatial Hearing: The Psychophysics of Human Sound Localization*. The MIT Press, 1996. DOI: 10.7551/mitpress/6391.001.0001.

[8] Jens Blauert, ed. *The Technology of Binaural Listening*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. DOI: 10.1007/978-3-642-37762-4.

[9] Fabian Brinkmann et al. "A Cross-Evaluated Database of Measured and Simulated HRTFs Including 3D Head Meshes, Anthropometric Features, and Headphone Impulse Responses". In: *Journal of the Audio Engineering Society* 67.9 (Sept. 21, 2019), pp. 705–718. DOI: 10.17743/jaes.2019.0024.

[10] C. P. Brown and R. O. Duda. "A Structural Model for Binaural Sound Synthesis". In: *IEEE Transactions on Speech and Audio Processing* 6.5 (Sept. 1998), pp. 476–488. DOI: 10.1109/89.709673.

[11] Christopher P. Burgess et al. *Understanding Disentangling in $\beta$-VAE*. Apr. 10, 2018. arXiv: 1804.03599 [cs, stat]. URL: http://arxiv.org/abs/1804.03599.

[12] J. Canny. "A Computational Approach to Edge Detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-8.6 (Nov. 1986), pp. 679–698. DOI: 10.1109/TPAMI.1986.4767851.

[13] Tzu-Yu Chen, Po-Wen Hsiao, and Tai-Shih Chi. "Exploring Redundancy of HRTFs for Fast Training DNN-Based HRTF Personalization". In: *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. Honolulu, HI, USA: IEEE, Nov. 2018, pp. 1929–1933. DOI: 10.23919/APSIPA.2018.8659704.

[14] Tzu-Yu Chen, Tzu-Hsuan Kuo, and Tai-Shih Chi. "Autoencoding HRTFS for DNN Based HRTF Personalization Using Anthropometric Features". In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Brighton, United Kingdom: IEEE, May 2019, pp. 271–275. DOI: 10.1109/ICASSP.2019.8683814.

[15] Wei Chen et al. "HRTF Representation with Convolutional Auto-Encoder". In: *MultiMedia Modeling*. Ed. by Yong Man Ro et al. Vol. 11961. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 605–616. DOI: 10.1007/978-3-030-37731-1_49.

[16] Chan Jun Chun et al. *Deep Neural Network Based HRTF Personalization Using Anthropometric Measurements*. undefined. 2017. URL: /paper/Deep-Neural-Network-Based-HRTF-Personalization-Chun-Moon/6a1a68113adc597bae11db67418739be4f35dad0.

[17] Corentin Guezenoc and Renaud Seguier. "Dataset Augmentation and Dimensionality Reduction of Pinna-Related Transfer Functions". In: *148th Audio Engineering Society Convention*. Audio Engineering Society, 2020.

[18] G. Cybenko. "Approximation by Superpositions of a Sigmoidal Function". In: *Mathematics of Control, Signals and Systems* 2.4 (Dec. 1, 1989), pp. 303–314. DOI: 10.1007/BF02551274.

[19] Richard O. Duda, V. Ralph Algazi, and Dennis M. Thompson. "The Use of Head-and-Torso Models for Improved Spatial Sound Synthesis". In: Audio Engineering Society, Oct. 1, 2002. URL: http://www.aes.org/e-lib/browse.cfm?elib=11294.

[20] Žiga Emeršič, Vitomir Štruc, and Peter Peer. *Ear Recognition: More Than a Survey*. Feb. 1, 2019. arXiv: 1611.06203 [cs]. URL: http://arxiv.org/abs/1611.06203.

[21] Michael J. Evans, James A. S. Angus, and Anthony I. Tew. "Analyzing Head-Related Transfer Function Measurements Using Surface Spherical Harmonics". In: *The Journal of the Acoustical Society of America* 104.4 (Oct. 1, 1998), pp. 2400–2411. DOI: 10.1121/1.423749.

[22] Brinkmann Fabian et al. "The HUTUBS Head-Related Transfer Function (HRTF) Database". In: (2019). In collab. with Technische Universität Berlin and Technische Universität Berlin. DOI: 10.14279/depositonce-8487.

[23] B. Friedlander and B. Porat. "The Modified Yule-Walker Method of ARMA Spectral Estimation". In: *IEEE Transactions on Aerospace and Electronic Systems* AES-20.2 (Mar. 1984), pp. 158–173. DOI: 10.1109/TAES.1984.310437.

[24] Mark B. Gardner and Robert S. Gardner. "Problem of Localization in the Median Plane: Effect of Pinnae Cavity Occlusion". In: *The Journal of the Acoustical Society of America* 53.2 (Feb. 1, 1973), pp. 400–408. DOI: 10.1121/1.1913336.

[25] Esther Gonzalez, Luis Alvarez, and Luis Mazorra. *AMI Ear Database*. URL: http://ctim.ulpgc.es/research_works/ami_ear_database/#whole.

[26] Corentin Guezenoc and Renaud Seguier. "HRTF Individualization: A Survey". In: Audio Engineering Society, Oct. 7, 2018. URL: http://www.aes.org/e-lib/browse.cfm?elib=19855.

[27] Corentin Guezenoc and Renaud Séguier. "A Wide Dataset of Ear Shapes and Pinna-Related Transfer Functions Generated by Random Ear Drawings". In: *The Journal of the Acoustical Society of America* 147.6 (June 2020), pp. 4087–4096. DOI: 10.1121/10.0001461.

[28] Kaiming He et al. *Deep Residual Learning for Image Recognition*. Dec. 10, 2015. arXiv: 1512.03385 [cs]. URL: http://arxiv.org/abs/1512.03385.

[29] Jack Hebrank and D. Wright. "Spectral Cues Used in the Localization of Sound Sources on the Median Plane". In: *The Journal of the Acoustical Society of America* 56.6 (Dec. 1, 1974), pp. 1829–1834. DOI: 10.1121/1.1903520.

[30] I. Higgins et al. "Beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework". In: *ICLR*. 2017.

[31] G. E. Hinton. "Reducing the Dimensionality of Data with Neural Networks". In: *Science* 313.5786 (July 28, 2006), pp. 504–507. DOI: 10.1126/science.1127647.

[32] Kazuhiro Iida. *Head-Related Transfer Function and Acoustic Virtual Reality*. Singapore: Springer Singapore, 2019. DOI: 10.1007/978-981-13-9745-5.

[33] Kazuhiro Iida and Yohji Ishii. "Effects of Adding a Spectral Peak Generated by the Second Pinna Resonance to a Parametric Model of Head-Related Transfer Functions on Upper Median Plane Sound Localization". In: *Applied Acoustics* 129 (Jan. 1, 2018), pp. 239–247. DOI: 10.1016/j.apacoust.2017.08.001.

[34] Kazuhiro Iida, Hikaru Shimazaki, and Masato Oota. "Generation of the Amplitude Spectra of the Individual Head-Related Transfer Functions in the Upper Median Plane Based on the Anthropometry of the Listener's Pinnae". In: *Applied Acoustics* 155 (Dec. 1, 2019), pp. 280–285. DOI: 10.1016/j.apacoust.2019.06.007.

[35] Kazuhiro Iida et al. "Median Plane Localization Using a Parametric Model of the Head-Related Transfer Function Based on Spectral Cues". In: *Applied Acoustics*. Head- Related Transfer Function and Its Applications 68.8 (Aug. 1, 2007), pp. 835–850. DOI: 10.1016/j.apacoust.2006.07.016.

[36] Yuvi Kahana and Philip A. Nelson. "Boundary Element Simulations of the Transfer Function of Human Heads and Baffled Pinnae Using Accurate Geometric Models". In: *Journal of Sound and Vibration* 300.3 (Mar. 6, 2007), pp. 552–579. DOI: 10.1016/j.jsv.2006.06.079.

[37] Yuvi Kahana and Philip A. Nelson. "Numerical Modelling of the Spatial Acoustic Response of the Human Pinna". In: *Journal of Sound and Vibration* 292.1 (Apr. 25, 2006), pp. 148–178. DOI: 10.1016/j.jsv.2005.07.048.

[38] Brian F. G. Katz. "Boundary Element Method Calculation of Individual Head-Related Transfer Function. I. Rigid Model Calculation". In: *The Journal of the Acoustical Society of America* 110.5 (Oct. 29, 2001), pp. 2440–2448. DOI: 10.1121/1.1412440.

[39] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. Jan. 29, 2017. arXiv: 1412.6980 [cs]. URL: http://arxiv.org/abs/1412.6980.

[40] Diederik P. Kingma and Max Welling. "An Introduction to Variational Autoencoders". In: *Foundations and Trends® in Machine Learning* 12.4 (2019), pp. 307–392. DOI: 10.1561/2200000056. arXiv: 1906.02691.

[41] Doris J. Kistler and Frederic L. Wightman. "A Model of Head-related Transfer Functions Based on Principal Components Analysis and Minimum-phase Reconstruction". In: *The Journal of the Acoustical Society of America* 91.3 (Mar. 1992), pp. 1637–1647. DOI: 10.1121/1.402444.

[42] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Communications of the ACM* 60.6 (May 24, 2017), pp. 84–90. DOI: 10.1145/3065386.

[43] Ajay Kumar and Chenye Wu. "Automated Human Identification Using Ear Imaging". In: *Pattern Recognition* 45.3 (Mar. 1, 2012), pp. 956–968. DOI: 10.1016/j.patcog.2011.06.005.

[44] Geon Lee and Hong Kim. "Personalized HRTF Modeling Based on Deep Neural Network Using Anthropometric Measurements and Images of the Ear". In: *Applied Sciences* 8.11 (Nov. 7, 2018), p. 2180. DOI: 10.3390/app8112180.

[45] Alexander Lindau, Torben Hohn, and Stefan Weinzierl. "Binaural Resynthesis for Comparative Studies of Acoustical Environments". In: Audio Engineering Society, May 1, 2007. URL: http://www.aes.org/e-lib/browse.cfm?elib=14017.

[46] H. Liu, Y. Fang, and Q. Huang. "Efficient Representation of Head-Related Transfer Functions With Combination of Spherical Harmonics and Spherical Wavelets". In: *IEEE Access* 7 (2019), pp. 78214–78222. DOI: 10.1109/ACCESS.2019.2921388.

[47] Enrique A. Lopez-Poveda and Ray Meddis. "A Physical Model of Sound Diffraction and Reflections in the Human Concha". In: *The Journal of the Acoustical Society of America* 100.5 (Nov. 1, 1996), pp. 3248–3259. DOI: 10.1121/1.417208.

[48] Yuancheng Luo, Dmitry N. Zotkin, and Ramani Duraiswami. "Virtual Autoencoder Based Recommendation System for Individualizing Head-Related Transfer Functions". In: *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. New Paltz, NY, USA: IEEE, Oct. 2013, pp. 1–4. DOI: 10.1109/WASPAA.2013.6701816.

[49] Piotr Majdak, Matthew J. Goupell, and Bernhard Laback. "3-D Localization of Virtual Sound Sources: Effects of Visual Environment, Pointing Method, and Training". In: *Attention, Perception, & Psychophysics* 72.2 (Feb. 2010), pp. 454–469. DOI: 10.3758/APP.72.2.454.

[50] Bruce Masterton, Henry Heffner, and Richard Ravizza. "The Evolution of Human Hearing". In: *The Journal of the Acoustical Society of America* 45.4 (Apr. 1, 1969), pp. 966–985. DOI: 10.1121/1.1911574.

[51] R. Miccini and S. Spagnol. "HRTF Individualization Using Deep Learning". In: *2020 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*. Mar. 2020, pp. 390–395. DOI: 10.1109/VRW50115.2020.00084.

[52] Riccardo Miccini and Simone Spagnol. "Estimation of Pinna Notch Frequency from Anthropometry: An Improved Linear Model Based on Principal Component Analysis and Feature Selection". In: *Combined Proceedings of the Nordic Sound and Music Computing Conference 2019 and the Interactive Sonification Workshop 2019*. Stockholm, Nov. 20, 2019, p. 4.

[53] John C. Middlebrooks. "Individual Differences in External-Ear Transfer Functions Reduced by Scaling in Frequency". In: *The Journal of the Acoustical Society of America* 106.3 (Sept. 1999), pp. 1480–1492. DOI: 10.1121/1.427176.

[54] John C. Middlebrooks and David M. Green. "Observations on a Principal Components Analysis of Head-related Transfer Functions". In: *The Journal of the Acoustical Society of America* 92.1 (July 1, 1992), pp. 597–599. DOI: 10.1121/1.404272.

[55] John C. Middlebrooks, Ewan A. Macpherson, and Zekiye A. Onsan. "Psychophysical Customization of Directional Transfer Functions for Virtual Sound Localization". In: *The Journal of the Acoustical Society of America* 108.6 (Dec. 2000), pp. 3088–3091. DOI: 10.1121/1.1322026.

[56] Parham Mokhtari et al. "Comparison of Simulated and Measured HRTFs: FDTD Simulation Using MRI Head Data". In: Audio Engineering Society, Oct. 1, 2007. URL: https://www.aes.org/e-lib/online/browse.cfm?elib=14298.

[57] Parham Mokhtari et al. "Frequency and Amplitude Estimation of the First Peak of Head-Related Transfer Functions from Individual Pinna Anthropometry". In: *The Journal of the Acoustical Society of America* 137.2 (Feb. 1, 2015), pp. 690–701. DOI: 10.1121/1.4906160.

[58] Parham Mokhtari et al. "Vertical Normal Modes of Human Ears: Individual Variation and Frequency Estimation from Pinna Anthropometry". In: *The Journal of the Acoustical Society of America* 140.2 (Aug. 2016), pp. 814–831. DOI: 10.1121/1.4960481.

[59] Masayuki Morimoto. "The Contribution of Two Ears to the Perception of Vertical Angle in Sagittal Planes". In: *The Journal of the Acoustical Society of America* 109.4 (Apr. 2001), pp. 1596–1603. DOI: 10.1121/1.1352084.

[60] Swen Müller and Paulo Massarani. "Transfer-Function Measurement with Sweeps". In: *J. Audio Eng. Soc* 49.6 (2001), pp. 443–471. URL: http://www.aes.org/e-lib/browse.cfm?elib=10189.

[61] Vikas C. Raykar, Ramani Duraiswami, and B. Yegnanarayana. "Extracting the Frequencies of the Pinna Spectral Notches in Measured Head Related Impulse Responses". In: *The Journal of the Acoustical Society of America* 118.1 (June 28, 2005), pp. 364–374. DOI: 10.1121/1.1923368.

[62] Lord Rayleigh. "On Our Perception of Sound Direction". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 13.74 (Feb. 1907), pp. 214–232. DOI: 10.1080/14786440709463595.

[63] L. A. J. Reiss. "Spectral Edge Sensitivity in Neural Circuits of the Dorsal Cochlear Nucleus". In: *Journal of Neuroscience* 25.14 (Apr. 6, 2005), pp. 3680–3691. DOI: 10.1523/JNEUROSCI.4963-04.2005.

[64] Sebastian Ruder. "An Overview of Gradient Descent Optimization Algorithms". In: (Sept. 15, 2016). URL: https://arxiv.org/abs/1609.04747v2.

[65] B. McA. Sayers. "Acoustic-Image Lateralization Judgments with Binaural Tones". In: *The Journal of the Acoustical Society of America* 36.5 (May 1, 1964), pp. 923–926. DOI: 10.1121/1.1919121.

[66] E. A. G. SHAW. "The Acoustics of the External Ear". In: *Acoustical Factors Affecting Hearing Aid Performance* (1980). URL: https://ci.nii.ac.jp/naid/10029359840/.

[67] Edgar AG Shaw. "Acoustical Features of the Human External Ear". In: *Binaural and spatial hearing in real and virtual environments* 25 (1997), p. 47.

[68] Ki Hoon Shin and Youngjin Park. "Enhanced Vertical Perception through Head-Related Impulse Response Customization Based on Pinna Response Tuning in the Median Plane". In: *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* E91.A.1 (2008), pp. 345–356. DOI: 10.1093/ietfec/e91-a.1.345.

[69] Kihyuk Sohn, Xinchen Yan, and Honglak Lee. "Learning Structured Output Representation Using Deep Conditional Generative Models". In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'15. Cambridge, MA, USA: MIT Press, Dec. 7, 2015, pp. 3483–3491.

[70] S. Spagnol. "HRTF Selection by Anthropometric Regression for Improving Horizontal Localization Accuracy". In: *IEEE Signal Processing Letters* 27 (2020), pp. 590–594. DOI: 10.1109/LSP.2020.2983633.

[71] S. Spagnol, M. Geronazzo, and F. Avanzini. "On the Relation Between Pinna Reflection Patterns and Head-Related Transfer Function Features". In: *IEEE Transactions on Audio, Speech, and Language Processing* 21.3 (Mar. 2013), pp. 508–519. DOI: 10.1109/TASL.2012.2227730.

[72] Simone Spagnol and Federico Avanzini. "Frequency Estimation of the First Pinna Notch in Head-Related Transfer Functions with a Linear Anthropometric Model". In: (2015), p. 6.

[73] Simone Spagnol, Riccardo Miccini, and Runar Unnthorsson. *The Viking HRTF Dataset V2*. Oct. 30, 2020. DOI: 10.5281/zenodo.4160401.

[74] Simone Spagnol et al. *Current Use and Future Perspectives of Spatial Audio Technologies in Electronic Travel Aids*. Wireless Communications and Mobile Computing. Mar. 21, 2018. DOI: 10.1155/2018/3918284.

[75] Simone Spagnol et al. "The Viking Hrtf Dataset". In: (2019), p. 6.

[76] C. Szegedy et al. "Going Deeper with Convolutions". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015, pp. 1–9. DOI: 10.1109/CVPR.2015.7298594.

[77] Hironori Takemoto et al. "Mechanism for Generating Peaks and Notches of Head-Related Transfer Functions in the Median Plane". In: *The Journal of the Acoustical Society of America* 132.6 (Dec. 1, 2012), pp. 3832–3841. DOI: 10.1121/1.4765083.

[78] Lei Wang, Xiangyang Zeng, and Xiyue Ma. "Advancement of Individualized Head-Related Transfer Functions (HRTFs) in Perceiving the Spatialization Cues: Case Study for an Integrated HRTF Individualization Method". In: *Applied Sciences* 9.9 (May 7, 2019), p. 1867. DOI: 10.3390/app9091867.

[79] Anthony J. Watkins. "Psychoacoustical Aspects of Synthesized Vertical Locale Cues". In: *The Journal of the Acoustical Society of America* 63.4 (Apr. 1, 1978), pp. 1152–1165. DOI: 10.1121/1.381823.

[80] Francis M. Wiener and Douglas A. Ross. "The Pressure Distribution in the Auditory Canal in a Progressive Sound Field". In: *The Journal of the Acoustical Society of America* 18.2 (Oct. 1, 1946), pp. 401–408. DOI: 10.1121/1.1916378.

[81] Frederic L. Wightman and Doris J. Kistler. "Headphone Simulation of Free-field Listening. I: Stimulus Synthesis". In: *The Journal of the Acoustical Society of America* 85.2 (Feb. 1, 1989), pp. 858–867. DOI: 10.1121/1.397557.

[82] Kazuhiko Yamamoto and Takeo Igarashi. "Fully Perceptual-Based 3D Spatial Sound Individualization with an Adaptive Variational Autoencoder". In: *ACM Transactions on Graphics* 36.6 (Nov. 20, 2017), pp. 1–13. DOI: 10.1145/3130800.3130838.

[83] Zhou Wang et al. "Image Quality Assessment: From Error Visibility to Structural Similarity". In: *IEEE Transactions on Image Processing* 13.4 (Apr. 2004), pp. 600–612. DOI: 10.1109/TIP.2003.819861.

# Appendix A

# Commands

The following is a list of commands used during the training of the various models discussed herein. The list is not meant to be comprehensive, but it is provided to illustrate the usage of the training scripts implemented as part of the project.

## A.1 Pinna VAE

### A.1.1 Train

```
# models
python train_ears.py vae_conv ./configs/data/ears/hutubs256.json
    --model_cfg ./configs/models/vae_conv/big.json --batch_size 128
    --max_epoch 1000
python train_ears.py vae_resnet ./configs/data/ears/hutubs256_3.json
    --model_cfg ./configs/models/vae_resnet/18.json --batch_size 64
    --max_epoch 1000
python train_ears.py vae_incept ./configs/data/ears/hutubs256.json
    --model_cfg ./configs/models/vae_incept/small.json --batch_size
    32 --max_epoch 1000
# 2d features
python train_ears.py vae_incept ./configs/data/ears/combined256.json
    --model_cfg ./configs/models/vae_incept/small.json --batch_size
    32 --max_epoch 1000
python train_ears.py vae_incept
    ./configs/data/ears/hutubs_depth256.json --model_cfg
    ./configs/models/vae_incept/small.json --batch_size 32
    --max_epoch 1000
python train_ears.py vae_incept
    ./configs/data/ears/hutubs_edges256.json --model_cfg
    ./configs/models/vae_incept/small.json --batch_size 32
    --max_epoch 1000
```

## A.2 HRTF CVAE (Regular)

### A.2.1 Train

```
python train_hrtf.py cvae_dense configs/data/hrtf/hutubs_full.json
    --model_cfg_path configs/models/cvae_dense/medium_full.json
    --batch_size 256 --max_epochs 10000 --nfft 512
python train_hrtf.py cvae_dense configs/data/hrtf/hutubs_median.json
    --model_cfg_path configs/models/cvae_dense/medium_full.json
    --batch_size 256 --max_epochs 100000 --nfft 512
```

## A.3 HRTF CVAE (interaural)

### A.3.1 Train

```
python train_hrtf.py cvae_dense
    configs/data/hrtf/hutubs_full_interaural.json --model_cfg_path
    configs/models/cvae_dense/medium_full.json --batch_size 256
    --max_epochs 10000 --nfft 512
python train_hrtf.py cvae_dense
    configs/data/hrtf/hutubs_median_interaural.json --model_cfg_path
    configs/models/cvae_dense/medium_full.json --batch_size 256
    --max_epochs 100000 --nfft 512
```

## A.4 DNN (Regular)

### A.4.1 Export

```
# z_ears
python export_z_ears.py vae_incept
    logs/VAE_incept_combined/version_0/checkpoints/epoch=478.ckpt
    --data_cfg_path configs/data/ears/hutubs256.json
python export_z_ears.py vae_incept
    logs/VAE_incept_hutubs_depth/version_1/checkpoints/epoch=221.ckpt
    --data_cfg_path configs/data/ears/hutubs_depth256.json
python export_z_ears.py vae_incept
    logs/VAE_incept_hutubs_edges/version_0/checkpoints/epoch=155.ckpt
    --data_cfg_path configs/data/ears/hutubs_edges256.json
# z_hrtf
python export_z_hrtf.py
    logs/CVAE_dense_hutubs_None/version_2/checkpoints/epoch=4626.ckpt
    --data_cfg_path configs/data/hrtf/hutubs_full.json --nfft 512
python export_z_hrtf.py
    logs/CVAE_dense_hutubs_None/version_4/checkpoints/epoch=4950.ckpt
    --data_cfg_path configs/data/hrtf/hutubs_median.json --nfft 512
```

### A.4.2 Train

```
# fullgrid
python train_latent.py dnn
    configs/data/z/regular/hutubs_edge_fullgrid.json --batch_size 256
    --model_cfg_path configs/models/dnn/medium2.json --max_epochs
    5000 --training_name edge_fullgrid
python train_latent.py dnn
    configs/data/z/regular/hutubs_depth_fullgrid.json --batch_size
    256 --model_cfg_path configs/models/dnn/medium2.json --max_epochs
    5000 --training_name depth_fullgrid
python train_latent.py dnn
    configs/data/z/regular/hutubs_image_fullgrid.json --batch_size
    256 --model_cfg_path configs/models/dnn/medium2.json --max_epochs
    5000 --training_name image_fullgrid
# fgmedian
python train_latent.py dnn
    configs/data/z/regular/hutubs_edge_fgmedian.json --batch_size 256
    --model_cfg_path configs/models/dnn/medium2_do.json --max_epochs
    10000 --training_name edge_fgmedian
python train_latent.py dnn
    configs/data/z/regular/hutubs_depth_fgmedian.json --batch_size
    256 --model_cfg_path configs/models/dnn/medium2_do.json
    --max_epochs 10000 --training_name depth_fgmedian
python train_latent.py dnn
    configs/data/z/regular/hutubs_image_fgmedian.json --batch_size
    256 --model_cfg_path configs/models/dnn/medium2_do.json
    --max_epochs 10000 --training_name image_fgmedian
# median
python train_latent.py dnn
    configs/data/z/regular/hutubs_edge_median.json --batch_size 256
    --model_cfg_path configs/models/dnn/medium2_do.json --max_epochs
    10000 --training_name edge_median
python train_latent.py dnn
    configs/data/z/regular/hutubs_depth_median.json --batch_size 256
    --model_cfg_path configs/models/dnn/medium2_do.json --max_epochs
    10000 --training_name depth_median
python train_latent.py dnn
    configs/data/z/regular/hutubs_image_median.json --batch_size 256
    --model_cfg_path configs/models/dnn/medium2_do.json --max_epochs
    10000 --training_name image_median
```

## A.5 DNN (Interaural)

### A.5.1 Export

```
python export_z_hrtf.py
    logs/CVAE_dense_hutubs_None/version_5/checkpoints/epoch=4300.ckpt
    --data_cfg_path configs/data/hrtf/hutubs_full_interaural.json
    --nfft 512 --output_path
    /data/riccardo_datasets/hrtfi/latent/hrtf_interaural_fullgrid
python export_z_hrtf.py
```

```
    logs/CVAE_dense_hutubs_None/version_10/checkpoints/epoch=7084.ckpt
    --data_cfg_path configs/data/hrtf/hutubs_median_interaural.json
    --nfft 512 --output_path
    /data/riccardo_datasets/hrtfi/latent/hrtf_interaural_median
```

### A.5.2 Train

```
# fullgrid
python train_latent.py dnn
    configs/data/z/interaural/hutubs_edge_fullgrid.json --batch_size
    256 --model_cfg_path configs/models/dnn/medium2_do.json
    --max_epochs 5000 --training_name edge_fullgrid_interaural
python train_latent.py dnn
    configs/data/z/interaural/hutubs_depth_fullgrid.json --batch_size
    256 --model_cfg_path configs/models/dnn/medium2_do.json
    --max_epochs 5000 --training_name depth_fullgrid_interaural
python train_latent.py dnn
    configs/data/z/interaural/hutubs_image_fullgrid.json --batch_size
    256 --model_cfg_path configs/models/dnn/medium2_do.json
    --max_epochs 5000 --training_name image_fullgrid_interaural
# fgmedian
python train_latent.py dnn
    configs/data/z/interaural/hutubs_edge_fgmedian.json --batch_size
    256 --model_cfg_path configs/models/dnn/medium2_do.json
    --max_epochs 10000 --training_name edge_fgmedian_interaural
python train_latent.py dnn
    configs/data/z/interaural/hutubs_depth_fgmedian.json --batch_size
    256 --model_cfg_path configs/models/dnn/medium2_do.json
    --max_epochs 10000 --training_name depth_fgmedian_interaural
python train_latent.py dnn
    configs/data/z/interaural/hutubs_image_fgmedian.json --batch_size
    256 --model_cfg_path configs/models/dnn/medium2_do.json
    --max_epochs 10000 --training_name image_fgmedian_interaural
# median
python train_latent.py dnn
    configs/data/z/interaural/hutubs_edge_median.json --batch_size
    256 --model_cfg_path configs/models/dnn/medium2_do.json
    --max_epochs 10000 --training_name edge_median_interaural
python train_latent.py dnn
    configs/data/z/interaural/hutubs_depth_median.json --batch_size
    256 --model_cfg_path configs/models/dnn/medium2_do.json
    --max_epochs 10000 --training_name depth_median_interaural
python train_latent.py dnn
    configs/data/z/interaural/hutubs_image_median.json --batch_size
    256 --model_cfg_path configs/models/dnn/medium2_do.json
    --max_epochs 10000 --training_name image_median_interaural
```

## A.6 DNN (PCA)

### A.6.1 Train

```
# fullgrid
python train_latent.py dnn
    configs/data/z/pca/hutubs_edge_fullgrid.json --batch_size 256
    --model_cfg_path configs/models/dnn/medium2_do.json --max_epochs
    5000 --training_name edge_fullgrid_pca
python train_latent.py dnn
    configs/data/z/pca/hutubs_depth_fullgrid.json --batch_size 256
    --model_cfg_path configs/models/dnn/medium2_do.json --max_epochs
    5000 --training_name depth_fullgrid_pca
python train_latent.py dnn
    configs/data/z/pca/hutubs_image_fullgrid.json --batch_size 256
    --model_cfg_path configs/models/dnn/medium2_do.json --max_epochs
    5000 --training_name image_fullgrid_pca
# fgmedian
python train_latent.py dnn
    configs/data/z/pca/hutubs_edge_fgmedian.json --batch_size 256
    --model_cfg_path configs/models/dnn/medium2_do.json --max_epochs
    10000 --training_name edge_fgmedian_pca
python train_latent.py dnn
    configs/data/z/pca/hutubs_depth_fgmedian.json --batch_size 256
    --model_cfg_path configs/models/dnn/medium2_do.json --max_epochs
    10000 --training_name depth_fgmedian_pca
python train_latent.py dnn
    configs/data/z/pca/hutubs_image_fgmedian.json --batch_size 256
    --model_cfg_path configs/models/dnn/medium2_do.json --max_epochs
    10000 --training_name image_fgmedian_pca
# median
python train_latent.py dnn configs/data/z/pca/hutubs_edge_median.json
    --batch_size 256 --model_cfg_path
    configs/models/dnn/medium2_do.json --max_epochs 10000
    --training_name edge_median_pca
python train_latent.py dnn
    configs/data/z/pca/hutubs_depth_median.json --batch_size 256
    --model_cfg_path configs/models/dnn/medium2_do.json --max_epochs
    10000 --training_name depth_median_pca
python train_latent.py dnn
    configs/data/z/pca/hutubs_image_median.json --batch_size 256
    --model_cfg_path configs/models/dnn/medium2_do.json --max_epochs
    10000 --training_name image_median_pca
```

## A.7 PRTF generation

```
# regular_edges_fullgrid
python ear_to_prtf.py configs/eval/regular/edges_fullgrid.json
    /data/riccardo_datasets/hrtfi/hutubs_edges/left/0_0_0_0/pp2_3DheadMesh.png
    --nfft 512 --device cuda:2 --output_path
    /data/riccardo_datasets/hrtfi/prtfs/regular_edges_fullgrid/2.mat
python ear_to_prtf.py configs/eval/regular/edges_fullgrid.json
    /data/riccardo_datasets/hrtfi/hutubs_edges/left/0_0_0_0/pp3_3DheadMesh.png
    --nfft 512 --device cuda:2 --output_path
```

```
    /data/riccardo_datasets/hrtfi/prtfs/regular_edges_fullgrid/3.mat
4 python ear_to_prtf.py configs/eval/regular/edges_fullgrid.json
    /data/riccardo_datasets/hrtfi/hutubs_edges/left/0_0_0_0/pp4_3DheadMesh.png
    --nfft 512 --device cuda:2 --output_path
    /data/riccardo_datasets/hrtfi/prtfs/regular_edges_fullgrid/4.mat
5 python ear_to_prtf.py configs/eval/regular/edges_fullgrid.json
    /data/riccardo_datasets/hrtfi/hutubs_edges/left/0_0_0_0/pp5_3DheadMesh.png
    --nfft 512 --device cuda:2 --output_path
    /data/riccardo_datasets/hrtfi/prtfs/regular_edges_fullgrid/5.mat
6 # regular_image_fullgrid
7 python ear_to_prtf.py configs/eval/regular/image_fullgrid.json
    /data/riccardo_datasets/hrtfi/hutubs_edges/left/0_0_0_0/pp2_3DheadMesh.png
    --nfft 512 --device cuda:2 --output_path
    /data/riccardo_datasets/hrtfi/prtfs/regular_image_fullgrid/2.mat
8 python ear_to_prtf.py configs/eval/regular/image_fullgrid.json
    /data/riccardo_datasets/hrtfi/hutubs_edges/left/0_0_0_0/pp3_3DheadMesh.png
    --nfft 512 --device cuda:2 --output_path
    /data/riccardo_datasets/hrtfi/prtfs/regular_image_fullgrid/3.mat
9 python ear_to_prtf.py configs/eval/regular/image_fullgrid.json
    /data/riccardo_datasets/hrtfi/hutubs_edges/left/0_0_0_0/pp4_3DheadMesh.png
    --nfft 512 --device cuda:2 --output_path
    /data/riccardo_datasets/hrtfi/prtfs/regular_image_fullgrid/4.mat
10 python ear_to_prtf.py configs/eval/regular/image_fullgrid.json
    /data/riccardo_datasets/hrtfi/hutubs_edges/left/0_0_0_0/pp5_3DheadMesh.png
    --nfft 512 --device cuda:2 --output_path
    /data/riccardo_datasets/hrtfi/prtfs/regular_image_fullgrid/5.mat
11 # regular_edges_median
12 python ear_to_prtf.py configs/eval/regular/edges_median.json
    /data/riccardo_datasets/hrtfi/hutubs_edges/left/0_0_0_0/pp2_3DheadMesh.png
    --nfft 512 --device cuda:2 --output_path
    /data/riccardo_datasets/hrtfi/prtfs/regular_edges_median/2.mat
13 python ear_to_prtf.py configs/eval/regular/edges_median.json
    /data/riccardo_datasets/hrtfi/hutubs_edges/left/0_0_0_0/pp3_3DheadMesh.png
    --nfft 512 --device cuda:2 --output_path
    /data/riccardo_datasets/hrtfi/prtfs/regular_edges_median/3.mat
14 python ear_to_prtf.py configs/eval/regular/edges_median.json
    /data/riccardo_datasets/hrtfi/hutubs_edges/left/0_0_0_0/pp4_3DheadMesh.png
    --nfft 512 --device cuda:2 --output_path
    /data/riccardo_datasets/hrtfi/prtfs/regular_edges_median/4.mat
15 python ear_to_prtf.py configs/eval/regular/edges_median.json
    /data/riccardo_datasets/hrtfi/hutubs_edges/left/0_0_0_0/pp5_3DheadMesh.png
    --nfft 512 --device cuda:2 --output_path
    /data/riccardo_datasets/hrtfi/prtfs/regular_edges_median/5.mat
16 # regular_image_median
17 python ear_to_prtf.py configs/eval/regular/image_median.json
    /data/riccardo_datasets/hrtfi/hutubs_edges/left/0_0_0_0/pp2_3DheadMesh.png
    --nfft 512 --device cuda:2 --output_path
    /data/riccardo_datasets/hrtfi/prtfs/regular_image_median/2.mat
18 python ear_to_prtf.py configs/eval/regular/image_median.json
    /data/riccardo_datasets/hrtfi/hutubs_edges/left/0_0_0_0/pp3_3DheadMesh.png
    --nfft 512 --device cuda:2 --output_path
```

```
    /data/riccardo_datasets/hrtfi/prtfs/regular_image_median/3.mat
python ear_to_prtf.py configs/eval/regular/image_median.json
    /data/riccardo_datasets/hrtfi/hutubs_edges/left/0_0_0_0/pp4_3DheadMesh.png
    --nfft 512 --device cuda:2 --output_path
    /data/riccardo_datasets/hrtfi/prtfs/regular_image_median/4.mat
python ear_to_prtf.py configs/eval/regular/image_median.json
    /data/riccardo_datasets/hrtfi/hutubs_edges/left/0_0_0_0/pp5_3DheadMesh.png
    --nfft 512 --device cuda:2 --output_path
    /data/riccardo_datasets/hrtfi/prtfs/regular_image_median/5.mat
```

# Appendix B

# Attachments

This section describes the content of the attachments accompanying this document.

## B.1 `src_python/`: Source code for PRTF generation

This directory contains all the necessary source code and configuration files for reproducing the findings presented here, as well as for generating individualized PRTFs. Due to their prohibitive size, the pre-trained weights were not included. See Appendix A for example usages.

The code depends on a multitude of external libraries and packages. The recommended way of installing and managing these dependencies is through `conda`[1]. A virtual environment can be instantiated and populated with the necessary libraries by running:

```
conda create --name rm18_thesis
conda activate rm18_thesis
conda install pytorch torchvision torchaudio -c pytorch
conda install pytorch-lightning -c conda-forge
conda install pip seaborn tabulate debugpy matplotlib librosa pandas
    scikit-image scikit-learn tensorboard python-dotenv jupyterlab
    tqdm scipy
pip install python-sofa pytorch-lightning-bolts
```

## B.2 `src_matlab/`: Source code for HRTF individualization

This directory contains all the necessary MATLAB script for generating individualized HRTFs in SOFA format from the output of the deep learning sub-system. Furthermore, the source code for the evaluation conducted using the psychoacoustic vertical localization model is provided. The requirements for these scripts are

[1] `https://docs.conda.io/en/latest/miniconda.html`

the SOFA APIs and the Auditory Modeling Toolbox mentioned in chapter 7. For more information, consult the `test_nopinnae.m` script.

## B.3 `logs/`: Training logs

This directory contains logfiles storing training-time metrics and data such as loss function values, learning rate, model hyperparameters, reconstructed samples, and so forth. The data can be accessed on the browser using TensorBoard, by running the following command:

```
1 tensorboard --logdir=path/to/logs
```

## B.4 `report/`: Report code

This directory contains the source code used to generate this document, as well as high-resolution versions of all the figures shown here. The compilation pipeline depends on Pandoc[2] and on a LaTeXdistribution. To compile, simply issue a `make` command.

## B.5 `notebooks/`: Jupyter notebooks

This directory contains some of the Jupyter notebooks used during the early stages of prototyping, data analysis, and for generating most of the figures in this report. The notebooks can be accessed and visualized using Jupyter. Alternatively, they can be dragged into an online visualizer[3].

[2] `https://pandoc.org/MANUAL.html`

[3] `https://kokes.github.io/nbviewer.js/viewer.html`