Automatic HRTF Individualisation Based on Localization Errors in 3D Space

Specification

Robin Yonge

Febuary 2017

Overview

Virtual reality has come to represent the dominant prediction for the near future of human-computer interaction[**?**]. If this prediction is to come to pass, then it is necessary that sufficient consideration is given toward the fidelity of both the visual and the auditory aspects of VR and AR technology. Currently, most implementations of spatial audio in VR and AR applications are based around the use of HRTFs. Mostof these implementations use one of a few publically available datasets, derived from measurements performed on either human subjects or a model such a the KEMAR[**?**][**?**]. For the use of VR/AR to become commonplace the experience must be universal. Because spatial audio relies on complex measurements of real-world subjects (be they flesh and blood or not) applications tend to use average or neutral HRTFs, ones that should work for the largest number of people. Of course, by dint of being average there are many people for whom these do not work. In order to achieve a universal experience, then, it is necessary to find a process for simple individualization of spatial audio that is sufficiently simple, and can be performed by the end user as a part of the standardsetup of any VR/AR device or application.

Goals and Research

The goal of this project is to produce a working prototype of an application that takes as input a neutral set of HRTF measurements, and returns an individualised set. For a process like this to achieve widespread use it must be simple enough that it can be performed by any given user with nothing more than a head-mounted display, and intuitive enough that it requires of the user no in-depth knowledge of spatial audio.

The output HRTF set should also provide a noticeable improvement in the ability of the user to locate sound sources in a virtual 3D environment.

Previous efforts to produce individualized HRTFs through either modification of an existing neutral set or total synthesis have focused primarily on a few different techniques. The most common method is based on a structural HRTF model [**?**, **?**], in which different characteristics of the HRTF are linked to physical characteristics of the pinna, torso, and head. Implementations based on this model have been varied [**?**, **?**, **?**], but the model's reliance on even limited sets of precise anthropometric measurements makes it impractical for my purposes - despite having the potential to be much more efficient than traditional HRIR measurement.

More recent studies have involved applying Principal Component Analysis (PCA) to HRTF data [**?**], in order to identify the spectral features that have the greatest effect upon the incoming sound signal. Once identified, these Principal Components (PCs) can potentially be adjusted in order to produce a more individualised HRTF [**?**, **?**]. Typically upwards of ninety percent of the information in a given HRTF is based on between 5 and 10 PCs[**?**], so through the adjustment of Principal Component Weights (PCWs) it should be possible to individualise HRTFs with a reasonably high degree of accuracy. The combined efficacy and efficiency of this method makes it well-suited to my proposed implementation, and as such this will be the model I will be using to perform the individualisation.

Once they have been identified, I will need data from the user on which to base any adjustment of PCWs. Because on my stated goal of producing an intuitive simple method, I will be performing the individualisation based on the user's own errors in localisation. This method will require me to identify, where possible, points of correlation between PCs and localisation, and as such this will be one of my primary research foci going forward. In order to gather this data from the user it is most practical to derive it from their own localisation errors in a virtual environment. Ideally, it will involve placing the user in virtual space, and asking them to identify the source of the audio cues that will be played to them. This helps to create an intuitive user experience, and giving the user a reticle to point at the perceived source gives precise data upon which to make adjustments.

notes:

Might still want to mention the incorporation of search algorithms/ML of a sort in deciding what PCWs to adjust.

Technologies

what technologies will I use and why?

The bulk of the implementation will be written in Python, particularly the core HRTF-adjusting module. Speed isn't too much of an issue the amount of processing to be done at one time will likely not exceed the capabilities of any device sufficiently powerful to run an HMD, and the access to the SciPy libraries[**?**] will make reading, visualising, and editing the files containing the HRTF data much simpler. The other option is to use MATLAB, which would require me to learn a whole new language for

very little benefit - it would allow me to skip one step in in processing the files, as most of the publicly-available HRTF datasets are stored in .mat files.

For the frontend UI component it is most likely that I will be building it using Unity[**?**] for quick prototyping and iteration. Other options, like other game engines and frameworks like OpenGL[**?**], would only make the process slower and offer no additional benefits aside from efficiency in the case of OpenGL It's very simple to build a small virtual space around a binocular camera object, adding a component to pass the required data back to the processing module as required. Using a game engine like Unity gives me more options, too, when it comes to processing whatever samples I choose to play to the user. Depending on my needs, I can either rely just on a plugin for Unity, or something using a dedicated piece of middleware such as FMod or Wwise. Being able to avoid processing the audio myself cuts down significantly on the workload, and potentially means that the final product, or one like it, could very easily be applied to any VR/AR application, provided the API I provide is comprehensive enough.

THIS BIT ABOUT THE DATASET I WILL USE

When it comes to testing, the best metric for measuring success is built into the program. The error rates of the user will be recorded in order to make modifications to the PCWs. If stored, this data should be all that is needed to ascertain the overall success or failure of the project. If the error rate gets significantly lower over time, then the algorithm can be considered a success. If there is little change or worse, an increase in localisation errors, then the algorithm is flawed. As part of the early investigation into PCA I may also generate a small number of HRIR-derived HRTFs covering a limited number of angles based on measurements of my own head. Though they would not be produced under as rigorous conditions as the neutral set, and would likely not be precise enough to use as proof of success or failure, they may be sufficient enough to serve as a comparison during earlier stages of development.

Timeline

Development Milestones

For the purpose of tracking progress on the implementation side of the project, I think it's helpful to define a set of development milestones.

Basic Functionality:

A set of classes and functions to parse the .mat files that contain the HRTF data and edit raw values as needed, storing the edited version in a copy.

Manual PCW-Adjustment:

Should produce a tool that, based on the functionality implemented already, allows me to manually adjust the weights of specific PCWs. This is essentially what was

produced alongside the papers by Josef Holzl[**?**], and should not be too difficult to develop based upon research by him and others.

Algorithm-Complete:

This stage requires me to have at least a preliminary version of the algorithm that I will be trying to use to automate this processing theoretically complete. At this stage I should be able to pass as input an HRTF from a particular angle, along with a vector describing the difference in position between the actual sound source and its perceived location, and receive as output an HRTF that has been modified to take that vector difference into account.

Minimum Viable Product:

The minimum that I could consider as fulfillment of my primary goals; the only difference between this and the complete version of the project is the UI. Requires all of the above features, along with the most basic possible frontend, likely a camera viewer floating in a void with a wire-frame sphere around it to give some visual context.

Finished Product:

The fulfillment of all my stated goals to the best of my ability. As mentioned, the UI difference between this and the above version would purely be the addition of an actual environment, turning the calibration process into almost a minigame. While this isn't strictly necessary to prove that the process is sound, it's important in the context of the project's role as a proof-of-concept, and insofar as I have given placed a degree of importance on the user experience aspects of the project.

Research Requirements

I don't know if this section is necessary, but all I'd outline is kinda what I need to learn about/nail down in order to do the implementation work above?

Principal Component Analysis:

Number of PCs required to get the best results, how I can link PCs to the user's specific errors, etc.

Search Algorithms:

I don't know if it's a case of now-that-I-have-a-basic-AI-shaped-hammer-everything-looks-like-that-kinda-nail, but it's looking at least a little like the search algorithms we're learning about could be useful in making decisions re what PCs to adjust when etc.

etc

I'll update this section properly when I'm sure it's necessary.

Gantt Chart/Timeline of Some Kind:

I think, with the extra time granted to me, I'd like to put together a nice visualisation rather than the original table thing I was trying to do!