# An efficient 3-D sound experience for mobile applications

Group 3 Aalborg University
Copenhagen, Denmark

***Abstract*—We developed an application for mobile devices that uses an efficient HRTF model based on to create 3D soundscapes. The HRTF model is based on a combination of filters and delays provided on theoretical basis by Brown et. al. [1]. The device orientation is used to set the angle between the sound and the user. This model has many possibilities on the limited hardware capabilities of the mobile devices in comparison to HRTF database based models. The prototype will present the user several sounds scattered in a limited area where the user can interact with**

***Index Terms*—HTRF, mobile devices, soundscape, Pure Data, OpenFrameworks**

## I. Introduction

The auditory system is used by the humans for several purposes in the daily life. One of this purposes is to provide the necessary information to localize sound sources in various dimensions, (width, height, depth) and it is even possible to guess the size of the source.

When a sound event occurs, the waves travel in all directions and , when they reach us, our brain compares the signals received by the left and right ears to localize it in the horizontal plane. The spectrum of the signal reaching each ear is different, since the amplitude and phase information differs. These binaural cues are called interaural intensity difference (IID) and interaural time difference (ITD). However, these cues are not enough to localize accurately the source since with this information the listener can not determine if the sound is in front, above or behind. This region of positions where all sounds yield the same ITD and IID is called cone of confusion. This ambiguity can be solved with the information provided by the filter effect caused by the pinnae, head, shoulders and torso, which modify the spectrum of the sound that reach the listener's ears. The sum of all these features are characterized by the Head Related Transfer Functions (HTRF) which are not only frequency and direction-dependent but also differ from person to person. It is therefore hard to generalize the spectral features among individuals. It is well known that using a HTRF from one person in another can significantly impair the perception due to the individual differences in the anatomy, but it has also been shown that some people localize better the sounds than others, and their HTRFs are suitable for a large group of listeners.

Thus, having a database with own measured HRTF for each user would be the ideal case. However, because of the large amount of data that the customized HTRFs required to generate externalized virtual sources would entail, and the expensive real-time computation of them every time the sound source moves, it is desirable to find a more efficient HTRF model.

A lot of research on modeling 3-D sound has been done with fairly good solutions (references). The best results can be obtained by those models implementing 3-D sound using personalized HRTF functions which needs tedious measuring of impulse responses etc (reference). Since mobile applications as well as multimedia productions are usually aimed at a big number of users, implementing personalized HRTF functions would require measurements of the individual user which is almost impossible. Additionally, using huge databases with HRTF's for subjects having similar head diameters and pinnae structures (dunno, how those HRTF's work) are not feasible for mobile applications as memory space is limited. Therefore, the aim of this project was to develop an computationally efficient and general HRTF model, yet, keeping the quality of the 3-D sound as good as possible. We therefore used the model provided on a theoretical basis in [**?**] by implementing a combination of filters and delays in pure data and C. This audio engine was then embedded in a mobile application. Sensor data provided by the phone or tablet was used to compute from which direction the sound should appear to be coming from. With the application users should be able to discover a virtual sound space situated in Copenhagen. This means that our HRTF model should satisfy two major requirements.
1. It should provide direction cues that are good enough to guide the user to the position where a specific sound is situated.
2. It should give the user a feeling of how far the sound is away from the users current location.
The first aim of course is bound to accuracy of the GPS signal and orientation data provided by the individual mobile phone. We hypothized that using a general HRTF model should be of sufficient quality to satisfy the first requirement since sensory data can be rather noisy anyways which would be overfitted by a very precise HRTF model.
The second hypothesis was that providing some interaction (button press on sound, then distance cues slides from farthest to current location etc.) could habituate the user to be able after some training to approximate the distance of the sound from the users current location.

Taken both hypothesises together the user should experience a soundscape in at least 2-dimensions, azimuth and distance.

The purpose of this paper is...*(it should be short, specific and immediate)*

First, some state of the art research on audio augmented reality will be provided. That section will be followed by the presentation of our implementation of the audio engine as well as the main application. In the last part we will present the results obtained by testing our application and discuss those in light of the two hypotheses stated above.

December, 2015

## II. State of the art

The concept of augmented audio reality (AAR) deals with techniques where a real sound environment is extended with virtual auditory environments. Nowadays, the progress in audio technology and computing predicts the introduction of completely new type of interactive audio applications []. Advances in mobile technologies have made it possible to create audio augmented spaces almost anywhere. For instance, spatial auditory displays that can provide the user with landmarks and are capable to attract the user's attention have been tested and introduced []. Many experiments, qualitative and quantitative researches have been designed so far to better understand the way in which people usually perceive multiple simultaneous sources differently placed and to increase the level of immersion in the experience.

### A. Audio reality vs Virtual reality

First of all, the possibility to hear the natural acoustic environment around a user differentiates the concept of augmented reality audio from the traditional concept of a virtual reality audio environment []. In virtual reality, generally participants are abstracted from the natural environment and are surrounded only by a completely synthetic one (acoustic and/or visual). On the opposite, in augmented reality a virtual environment is *superimposed* on a real one. To be more specific, in a mobile audio augmented environment participants are able to interact with the virtual audio mixed with real vision and soundscape.

### B. Augmented audio reality

The aim of AAR is to combine both real and virtual sound scenes so that virtual sounds are perceived as an extension to the natural ones. Virtual sound, as expressed in [], could be originated from another environment or artificially created. Two extreme possible augmentations are possible. The former is a perfect augmentation of the listener's auditory environment and is achieved when the listener is unable to predict whether a sound source is part of the real or the virtual audio environment. The latter is a high-quality virtual auditory scene with some characteristic that are not possible in acoustic environments.

### C. Previous works

Several work has been done in this field although this area is relatively new. *Audio Aura* (Mynatt 1995) was one of the first project to deal exclusively with audio in augmented reality system. It basically consisted in providing information to users as they travelled through their workspace. These information was triggered by particular locations in the workspace. The use of audio in *Audio Aura* is particularly interesting because most of its cues were associated to sounds from nature rather then recorded vocal, speech or synthetic sounds. Similar in approach to *Audio Aura* was the *Automated Tour Guide* (Bederson 1995). In both cases, triggers are readily identifiable, still and rarely changing. Those augmented sounds were associated to pre-determined locations in space and there was no need to determine the precise location of an individual.

A successive work, *Hear&There*, was able to determine the location and head position of the user using the information from GPS and a digital compass []. A user could listen to these 'audio imprints' by walking into the area that a specific imprint occupied, which was triggered by proximity. The essential premise of *Hear&There* was that a physical environment has been augmented with audio. All the sounds and the data were gathered inside that system. Since a 'Field Use' has been developed, in which the user wear a hardware portable system, *Hear&There* has undoubtedly contributed to an improved definition of mobile augmented reality environment.

### D. Localization/Lateralization

Headphones are much related to the concept of mobile and wearable application and have been used successfully in many virtual reality applications. Headphones, and earphones as well, make the ear receive the sound separately. That is advantageous because the whole scale of differences between the signals coming to ears, i.e. *interaural time differences* (ITD) and *interaural intensity differences* (IID), can be manipulated separately and individually [].

Generally speaking, when listening to sounds coming from headphones, the sound sources seem to be located inside the subject's head []. This is usually called *lateralization*, or intracranial, or 'inside-head-localization' (IHL). On the opposite, there is the effect of having the sound outside the heed, according to specific direction and distance, i.e. *localization* or 'outside-head-localization' (OHL). It has been necessary distinguish localization inside and outside the head. This difference in terminology has reinforced the assumption that there is a basic difference between perception of sound conveyed by headphones and sound that reaches the ear from an external and real source []. It has also demonstrated that a listener can make a clear distinction in headphones listening between localized (that is, sound inside the head) and lateralized sounds sources and that both these type can coexist in the listener's experience [].

### E. Issues in headphones-conveyed sound

Headphones auralization often produces an incorrect localization of virtual sound sources [] and many issues could be experienced.

*1) Externalization errors:* externalization is related to the perception of auditory distance such that there is a continuum in perceived locations of sources from inside the listener's head to any external position. One of the most severe problem in

AAR is a perceived effect of *lateralization* even in sounds that should be located in a real environment. To avoid such effect several techniques can be used. For instance, as expressed in [], the effect of a lateralized sound in headphone listening can be produced using amplitude and delay differences in two headphone channels corresponding to each source. The main goal of virtual acoustic synthesis should be to produce sounds that seem *externalized*, that is, outside the listener's body []. In order to make a sound source externalized and let the user be capable of a correct judgement of the distance of the sound more sophisticated binaural techniques are needed. In particular, spectrum differences in the two ear signal due to *head-related transfer functions*, HRTF's, play an important role. Moreover, acoustic cues such as the amount of reverberation and control of signal level are necessary for a successful auralization of a virtual sound source. These are explained into details in further sections.

*2) Localization errors: localization* error refers to the deviation of the reported position of a sound source from a measured 'target' location, i.e. the listener fails in matching the correct location of the sound. Localization errors can be divided in *azimuth* (deviations along the horizontal plane) and *elevation* errors (deviation from eye-level elevation) []. Dynamic cues related to head turning and other movements of either a listener or a sound source should be taken into consideration []. These errors might come from some location accuracy. Determining the accurate position of the user of one of the most important task in an AR system owing to that system always produces output to the used based on his or her location in space []. That said, any location inaccuracy should be avoided using GPS receiver with high sensitivity and reliability. Here, the implementation design takes a crucial role as noted in [].

*3) Reversal errors:* sometimes called front-back or back-front 'confusion', that error refers to the judgement of a sound source as located on the opposite side of the interaural axis than the target position []. An informal proposal has been made, which tries to help the user in front-back discrimination on the basis of the familiarity of the effects on timbre cues, e.g. unique patterns in *early reflections* depending on the virtual sound location. Indeed, this has not yet been verified experimentally.

### F. Conclusion

Although it is a recent field of research, several works and experiments in the context of audio augmented reality has been made so far. Most of research has been done mainly on the use of visual information with very little attention paid to the audio aspects.
The use of audio and its refinement in an augmented reality environment is a big deal. There are many issues and challenges that an audio environment presents []. For instance, providing a way for the user to orient himself or herself in an only-audio virtual space is even more difficult than providing visual cues. Another interesting aspects of audio, as also explained in [], is the fact that audio does not persist and it starts disappearing as soon as it has heard. That deals with a more attention due to the audio to capture every detail that may help the user retrieve a specific sound in a specific location.
Finally, with the advent of portable systems such as sensors-equipped smartphone and laptop a further development of *geolocated AR* has been made possible. That systems use locational sensing and user tracking technologies (e.g. Global Positioning System, or GPS) to let the user navigate through a audio-augmented space and to get a more defined listener's position, which is essential for a correct externalizaton. More information will be given in further sections.

## III. Design and implementation choiches

*also some argumentation why we used a HRTF model based on filtering rather than using a database (citations why database not efficient and why quality better with databases but good enough with filtering)*

## IV. The application

### A. openFrameworks

We decided to develop our application for Android platform. Nevertheless, we did not use the facilities and the built-in code provided by Android Studio, but instead we decided to use openFrameworks.

The reasons why we decided to develop using this programming environment was the need of a fast communication between the mobile platform and our program code. It works using C++ language, which implies that the code is compiled directly into assembly language and hence works very fast [] and does not need to be interpreted by a virtual machine as it is the case for a java-built environment such as Android Studio [].
Discussing all the steps involved in the building process is beyond the scope of this paper (and more information can be found in [] and []), but a short explanation of what is openFrameworks and the main operations used to achieve our application will be given.

*1) Description:* OpenFrameworks, by definition given on its website, is:

> *an open source C++ toolkit for creative coding* []

Since it is entirely written in C++, distributed under MIT license and actually runs on five operative systems and four IDEs, it undoubtedly presents some kind of advantages. It gives the opportunity to several users to deal with code designed to be minimal and easy to grasp and use it on the favourite IDE. In that case, we use both OS X and Windows platform, and Xcode and Visual Studio as integrated development environment. That *simple and intuitive framework for experimentation* [] is designed to work as a general purpose glue and wraps together several commonly used libraries, such as *OpenGl, OpenCv, PortAudio* and many more. Nowadays, this is a popular platform for experiments in generative and sound art and creating interactive installations and audiovisual performances []. The current operative version is *0.9.0*.

*2) Addons:* its design philosophy claims for a collaborative environment. It thrives on the contributions of many people, and collaborate mainly on *addons* and projects. An *addons* is made of several snippets of code put together in order to extend openFrameworks functionality, bring some external framework and allow it to be integrated into openFrameworks project or make specific and complicated tasks easier and reusable in other contexts []. It also generally contains the library itself in a form that is ready to be linked to project binaries. In our case, as depicted in the next section, we used several *third-party* addons and built our own library called *ofxOrientation* to access sensors information about localization and orientation and provide the HTRF model with right values.

*3) Main issues:* openFrameworks requires such lots efforts to be mastered and its learning rate is quite steep. Its weakness is in fact due to the lack of documentation and people who don't have at least some programming background could feel quite uncomfortable. Hopefully, its community is continuously providing discussion and projects, while addons are released with some example to compile and have to be created using a bunch of accurate rules.

*4) The openFrameworks project:* all openFrameworks project have a similar structure of folders and files. The most important folder among then is for sure the *src* folder. It contains all the source codes and consists at least of `main.cpp` (containing the `main( )` function to let the operating system start the application), `ofApp.h` (containing declaration of the specific class) and `ofApp.cpp`, which contains definition of all functions declared in the previous file. All the methods in that class are *event-handling* methods, that is they are triggered in response to events that happens inside the application such as mouse scrolling and program quitting .

To create a new project, we used the Project Generator wizard located in the same directory and directly provided by the environment. Such a way is simple and it is especially useful when dealing with several addons, which are automatically linked. Once we got the structure, we could access every source file in the *src* folder. At its simplest, working with an openFrameworks project is adding new code to the appropriate method, or just create a new one and declare it in the `ofApp.cpp`. In [] a further explanation of the main methods and their workflow is provided.

*5) Compiling: soon...*

### B. Sensor data retrieving

Our application required two kind of sensors data: *GPS* and *orientation*.

- GPS is a way of determining location that has become amazingly common with the increasing of portable technologies. Any device that receives a GPS signal is called a GPS *receiver*. As explained in [] the receiver calculates its position by precisely timing the signals sent by the GPS satellites. We needed the GPS user position to calculate the distance to the sounds that were placed on fixed locations in the real environment.
- orientation usually is derived using a combination of field sensors, such as accelerometer and magnetometer. In this case, we needed orientation to define the angle the user is heading relative to the sound sources.

To access the GPS data we used *ofxMaps* and *ofxGeo* addons (reference to webpage/gitrepository). Reading the GPS position on an Android platform requires a listening mechanism and we implemented it in openFrameworks by calling the `ofRegisterGPSEvent()` in the `setup()` method of our main code. We also added a method to our application, in the `ofApp.h` precisely, to handle the updates from the Android OS system calls. It is named `locationChanged()` and adds an event handler for the dispatched event []. Moreover, the `startGPS()` method is called at the beginning and `stopPGS()` should call when quitting the application to avoid an overconsumption of phone battery.

*(Something about smoothing if we used it, and maybe we have to look into the code how they used smoothing).*

Since we are dealing with some kind of filtering depending on the sound sources position (e.g. the sound could appear in front of the user as well as behind), it is necessary to take into consideration the user's orientation. That is basically the angle created between the heading of the user and the geographic north pole. Initially, we assumed the user always faced to the north, i.e. the angle between the user and the geographic north pole[1] is equal to 0. If we can discard the deviation of the user's orientation, it is possible to define the angle between the user and the sound source as the *true bearing* [2]. This is, by definition

> *the angle measured in degrees in a clockwise direction* [3] *from the north line [].*

If we also assume a sound source as a point that radiates in every directions, we could easily calculate the bearing of the user from the sound source, as shown in (add figure). The bearing could be accessed via the *ofxGeo* addon, which contains a specific method called *GeoUtils::bearingHaversine* that uses the **'haversine'** formula:

$$\theta = atan2(sin(\delta\gamma)cos(\varphi_2), cos(\phi_1)sin(\varphi_2) - sin(\varphi_1)cos(\varphi_2)cos(\delta\gamma)) \quad (1)$$

where $\varphi isthelatitude, \delta isthelongitude$.

Once we got the bearing, we can start supposing that the angle between the heading of the user and the sound source should be the sum of the bearing and the direction the user is facing. Doing so, we can retrieve it using a compass. In order to access the compass orientation we built our own addon accessing the 'SENSOR TYPE ORIENTATION' provided by the android API. Having this information we now had to differentiate between four cases: (some images and explanation about how we calculated the angle between our heading orientation and the sound). In order to smooth the sensor data (especially for the orientation) we filtered the sensory information by computing the median of a chunk of mediansize instances of the sensor output.

[1] There's a conceptual difference between *geographic* and *magnetic* north pole.

[2] It is possible refer to it simply as *bearing*.

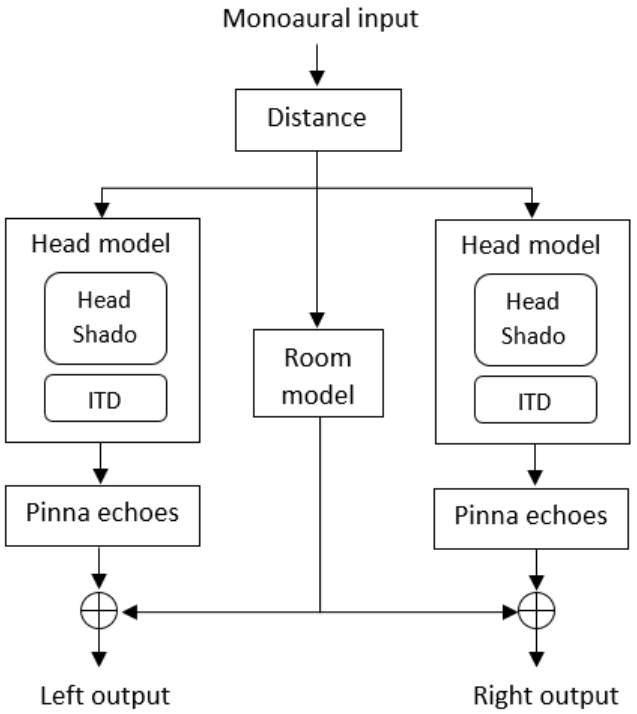
[3] All bearings are measured in a horizontal plane.

### C. Audio engine

The audio engine was the most crucial part of our applications. As shortly mentioned in the introduction we decided to use a filter based HRTF model which claimed to be both 'efficient' and of reasonable 'quality' [**?**] in order to achieve a satisfying user experience and an efficient real time application. We basicly followed the workflow (order of filter and delay lines) given in [**?**] and depicted in figure ... .

*Andrea*

First the sound had to be split up in .... Second ..... Third IID and ITD




We therefore used the analog filter models provided in [**?**] and further specified in [**?**]. The head shadow filter, implementing the ILD, was given by the following analog transfer function:

$$H(s,\theta) = \frac{\alpha(\theta)s + \beta}{s + \beta},\ where\ \beta = \frac{2c}{a} \tag{2}$$

Since this is an analog transfer function we had to derive the digital version by applying a bilinear transform applying the following substitution:

$$s = \frac{2}{T}\frac{z-1}{z+1},\ where\ T\ is\ ???\tag{3}$$

Applying the substitutin in Equation 3 to Equation 2 we get the following filter function in the digital domain:

$$H(z,\theta) = \frac{\alpha(\theta)(\frac{2}{T}\frac{z-1}{z+1}) + \beta}{(\frac{2}{T}\frac{z-1}{z+1}) + \beta} \tag{4}$$

To identify the filter coefficients we transformed Equation 4 and obtained the following frequency response[4]:

$$H(z,\theta) = \frac{2\alpha(\theta) + T\beta + z^{-1}(-2\alpha(\theta) + T\beta)}{2 + T\beta + z^{-1}(-2 + T\beta)} = \frac{Y(z)}{X(z)} \tag{5}$$

This gives us the filter coefficients $a_0 = 2\alpha(\theta) + T\beta$ and $a_1 = -2\alpha(\theta) + T\beta$ as well as $b_0 = 2 + T\beta$ and $b_1 = -2 + T\beta$. Since we could not use the filter function in Matlab which has as input values the filter coefficientes we had to isolate the output $Y(z)$ in Equation 5 as well as going from the frequency to the time domain. The result is given in Equation 6[5]:

$$Y[n] = \frac{a_0 X[n] + a_1 X[n-1] - b_1 Y[n-1]}{b_0} \tag{6}$$

### D. User interface

include pictures of user interface etc.

## V. Testing

### A. Test group

### B. Methodology and procedure

### C. Results

## VI. Discussion

## VII. Conclusion

## Acknowledgment

The authors would like to thank...Smilen and Stefania :)

[4]For the derivation of Equation 5 see appendix A
[5]For the derivation of Equation 6 see appendix B

## LIST OF FIGURES

## APPENDIX A
## DERIVATIONS

$$
\begin{aligned}
H\left(z,\theta\right) &= \frac{\alpha(\theta)(\frac{2}{T}\frac{z-1}{z+1})+\beta}{(\frac{2}{T}\frac{z-1}{z+1})+\beta} \\
&= \frac{\frac{2\alpha(\theta)(z-1)}{T(z+1)}+\frac{T\beta(z+1)}{T(z+1)}}{\frac{2(z-1)}{T(z+1)}+\frac{T\beta(z+1)}{T(z+1)}} \\
&= \frac{\frac{2\alpha(\theta)(z-1)+T\beta(z+1)}{T(z+1)}}{\frac{2(z-1)+T\beta(z+1)}{T(z+1)}} \\
&= \frac{2\alpha(\theta)(z-1)+T\beta(z+1)}{2(z-1)+T\beta(z+1)} \\
&= \frac{z2\alpha(\theta)(1-z^{-1})+zT\beta(1+z^{-1})}{z2(1-z^{-1})+zT\beta(1+z^{-1})} \\
&= \frac{2\alpha(\theta)(1-z^{-1})+T\beta(1+z^{-1})}{2(1-z^{-1})+T\beta(1+z^{-1})} \\
&= \frac{2\alpha(\theta)-2\alpha(\theta)z^{-1}+T\beta+T\beta z^{-1}}{2-2z^{-1}+T\beta+T\beta z^{-1}} \\
&= \frac{(2\alpha(\theta)+T\beta)-(2\alpha(\theta)+T\beta)z^{-1}}{(2+T\beta)-(2+T\beta)z^{-1}} \\
&= \frac{(2\alpha(\theta)+T\beta)+(-2\alpha(\theta)+T\beta)z^{-1}}{(2+T\beta)+(-2+T\beta)z^{-1}}
\end{aligned}
$$

## APPENDIX B
## DERIVATIONS

$$
\begin{aligned}
H(z,\theta) = \frac{Y(z)}{X(z)} &= \frac{a_0+a_1z^{-1}}{b_0+b_1z^{-1}} \\
&\iff Y(z)(b_0+b_1z^{-1}) = X(z)(a_0+a_1z^{-1}) \\
&\iff Y(z)b_0+b_1Y(z)z^{-1} = a_0X(z)+a_1X(z)z^{-1} \\
&\iff Y(z) = \frac{a_0X(z)+a_1X(z)z^{-1}-b_1Y(z)z^{-1}}{b_0} \\
&\rightarrow Y[n] = \frac{a_0X[n]+a_1X[n-1]-b_1Y[n-1]}{b_0}
\end{aligned}
$$