An efficient 3-D sound experience for mobile applications

Group 3 Aalborg University
Copenhagen, Denmark

***Abstract*—The computational power of mobile devices has highly increased in the last few years and nowadays having a device with Global Positioning System (GPS) and compass sensor has become commonplace. These facilities opens up possibilities to enhance the user experience in daily life. In this paper an application for mobile devices that uses an efficient head related transfer function (HRTF) model to create 3-D soundscapes is presented. In a small experiment the developed 3-D audio engine is compared with a cosine panner model [?] in terms of quality and efficiency of the navigational cues. Although the experiment did not reveal significant differences between the two models a critical observation of this study supports that a more sophisticated 3-D audio engine can increase the user experience in audio navigation.**

***Index Terms*—Head related transfer function (HRTF), mobile devices, Global Positioning System (GPS), soundscape, Pure Data, OpenFrameworks**

I. Introduction

A lot of research on modeling 3-D sound has been done with fairly good results [**?**]. The best results can be obtained by those models implementing 3-D sound using personalized head related transfer functions (HRTF), which need tedious measuring of impulse responses for each individual and require large databases [**?**], [**?**]. Mobile applications as well as multimedia productions are usually aimed at a big number of users, thus implementing HRTFs requiring measurements of the individual user is highly unpractical. Furthermore the convolution needed for using databases with head related impulse responses (HRIR) is an unnecessary heavy technique in terms of computational power and memory, two requirements that are not abundant in mobile devices. Therefore, the aim of this project was to develop a computationally efficient and general HRTF model, yet, keeping the quality of a 3-D sound experience as good as possible. We 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. Compass and GPS data provided by the mobile device was filtered and then used to compute the apparent direction and distance to the sound source. The purpose of the application was to let the users discover a virtual sound space situated in Copenhagen. Thus, the 3-D audio engine should to satisfy two major requirements:

1) It should provide direction cues that are efficient in guiding the user to the position where a specific sound is situated.
2) It should provide an intuitive way of locating a sound source from a qualitative point of view.

The first aim refers to the speed at which the user can find a sound source receiving only the auditory cues, which optimally would not differ to a great extent of the time needed to navigate to the position of a natural sound source. The 3-D audio engine should yield better or at least as good result as a simple stereophonic panning as some studies already provided evidence of the efficiency of a simple panning to guide the user to a specific location **COMMENT:LARS[Here we have to put some citations][]**. Our second aim was that the model should give a qualitatively more natural and intuitive feeling of the sounds position in the space compared to the panning. While for the panning some level of abstraction as simple as hearing the sound louder from the left meaning that the sound is at the left has to be applied, with the 3-D audio engine we aimed to model a sound coming from a certain direction in a more natural way using a HRTF model which combines direction, distance and externalization cues.

In this paper, first, state of the art research on audio augmented reality will be elaborated to set the theoretical framework. In the second part, the implementation of an efficient and general HRTF model for mobile devices will be presented. Thereafter, to evaluate the implemented model, a comparative study between the 3-D audio engine and a simple stereophonic panning will be presented, and its results based on qualitative and quantitative measures will be discussed. In the last part, a thorough discussion of the results in light of the two demands to the implemented 3-D audio engine will be presented and conclusions will be drawn.

A. State of the art in audio augmented reality

Recently, the progress in audio technology and computing paved the way for the introduction of completely new types of interactive audio applications [**?**]. Advances in mobile technologies have made 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 [**?**]. *COMMENTS:LARS[Do we need this sentence? Because we say nothing about multiple sound sources except for the future works. So maybe we can put it there] ANDREA: I agree with Lars :) -¿* Many experiments assessing qualitative and quantitative measures 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.

Theoretically, audio augmented reality has to be distinguished 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 a real visual scene and/or auditory soundscape [**?**].

COMMENT:LARS [We have to reformulate this one once the citation is put in ;)] we wait for Mattia According to this definition, audio augmented reality (AAR) should be within the boundaries of 1) 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 and 2) a set of artificial sounds that are not possible in the real world superimposed and fitted to the visual perceived world like in described in Harma et. al. [**?**]. Any combination of these two fall within the boundaries of AAR.

Human sound localization:

The auditory system provides the necessary information to localize sound sources in various dimensions (width, height, depth), process which takes place in the brain [**?**].

When a sound event occurs, waves travel in all directions and when they reach us, our brain compares the signals received by the left and right ears in order to determine the sound source position. The signal spectrum reaching both ears is different, since the amplitude and phase information differ [**?**]. 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 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 pinna, head, shoulders and torso, which modify the spectrum of the sound that reach the listener's ears. Additionally, other cues as head movements and visual cues help to reduce these localization ambiguities [**?**]. 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 [**?**]. That dependency makes 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 individual differences in 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 [**?**]. Therefore there is some evidence that it is possible to generalize, and it is worth to allow the listener choose their own HRTF set [**?**], although the people localize better with their own HRTFs.

On the other hand, it is possible to render binaural audio using personalized HRTFs with some methods, as the one presented in [**?**], which is based on parametric representations of the HRTFs instead of convolution.

Localization/Lateralization:

In order to have a well binaural reproduction, the features of the HRTFs must be accurately simulated at the listener's ears. Fortunately, the use of headphones facilitates this work since the different signals played to the user's ears can be manipulated separately and individually [**?**].

However, one of the most critical problem when using headphones is the disability of the listener to hear the sound source placed in the physical space. Instead, it is often perceived inside the head [**?**], effect which is usually called *lateralization*, or intracranial, or 'inside-head-localization' (IHL). On the opposite, the effect of hearing the sound outside the head, according to specific direction and distance, is called *localization* or 'outside-head-localization' (OHL) [**?**]. This difference in terminology serves to emphasize the difference between a sound source conveyed directly by headphones and a real source, as underlined by Plenge in [**?**]. It has also been demonstrated that a listener can clearly distinguish when listening through headphones between localized and lateralized sounds sources and that both types can coexist in the listener's experience [**?**].

Issues in headphones-conveyed sound:

Reproduction of sound through headphones often produces an incorrect localization of virtual sound sources [**?**] and many issues could be experienced.

a) Externalization errors: Externalisation is related to the perception of distance between the head and the sound source [**?**]. One of the most relevant problems in AAR is the perceived effect of *lateralization*, i.e. the sounds appear to be inside the listener's head. 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 between two headphone signals. The main goal of virtual acoustic synthesis should be to produce sounds that seem *externalized*, that is, outside the listener's head [**?**]. In order to make a sound source externalized and let the user be capable of a correct judgment of the distance of the sound, more sophisticated binaural techniques are needed. In particular, spectrum differences in the HRTF between the two ear signals, play an important role, as also expressed in [**?**]. 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. Finally, also dynamic cues related to head turning and other movements of either a listener or a sound source should be taken into consideration when dealing with externalization. As described in Harma et. Al. [**?**] unnatural changes in the acoustic properties of the virtual sound sources should be avoided, which could be caused by some user intentionally or unintentionally movement.

b) 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) [**?**]. These errors might come from some location accuracy. Determining the accurate position of the user is 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 [**?**].

c) Reversal errors: Sometimes called front-back or back-front 'confusion', that error refers to the judgment of a sound source as located on the opposite side of the interaural axis than the target position [**?**] due to the cone of confusion. 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. (COMMENT - mattia: is it necessary the following sentence?) However in application where the listener can change its angle to the source, the front and back will soon be obvious through the movement of the sound. (MG: insert references for previuos sentences).

1) Related works: Audio Aura [**?**] was one of the first projects to deal exclusively with audio in augmented reality systems. It basically consisted in providing auditory information to users as they traveled through their workspace. These information was triggered as soon as the subjects entered particular locations in the workspace. The auditory cues used in *Audio Aura* were particularly interesting because most of them 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* [**?**] COMMENT:LARS[add something more about that automated tour guide like: ... in which subjecta had to do ...]. In both cases, triggers are readily identifiable, still and rarely changing. The sounds were assigned to pre-determined locations in space and there was no need to determine the precise location of an individual except for knowing when the individual entered the area of interest. COMMENT:LARS[How did they know when to play the sound???]

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'COMMENT:LARS[is this a direct citation?] by walking into the area that a specific imprint occupied, which then was triggered. 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. COMMENT:LARS[I am not sure if it contributed to the definition, rather then just falling into the definition?]

More recent works in this field use mobile devices to render soundscapes and let the users interact with them. One example is the *In Situ Audio Services* (ISAS) application [**?**], addressed to blind users, which has been a significant reference for the work presented here. ISAS uses the GPS and orientation sensory data of the mobile device to feed an audio engine based on interaural time differences (ITD) and a low pass filter to reduce the back-front confusion.

II. Design, Implementation and Mobile Application

A. Introduction

Given the high computational and memory costs that using a spatial audio model based on a traditional HRTF implies, the existing mobile applications employ different approaches to localize the sound. In the development of ISAS, Blum and Bouchard discarded the usage of the Pure Data Extended object [earplug~][1], which is a real-time binaural filter based on KEMAR impulse measurement, due to its high CPU usage. Instead and as said before, they used simple panning techniques combined with ITD and filtering effects. For this work, [earplug~] was also considered, but it showed low accuracy. Hence, it was not been tested on a mobile device.

One of the aims of this project was to implement the HRTF model introduced by Brown and Duda in [**?**] on a mobile application. This model allows an efficient processing of multiple sounds in real time and it has been implemented in Pure Data and C. A more detailed description of the implementation follows.

Some of the challenges that this implementation has presented are: 1) Does an HRTF model improves the navigation in comparison to a simpler model? 2) How many sounds can be rendered using it?

This project focused on evaluating the improvements that this HRTF model presents in terms of navigation, and for this purpose two mobile applications were developed. The former used the HRTF model, and the latter used a cosine Panner model [**?**]. This is detailed in the experiment section of this paper.

Finally, Android is the platform that was used to develop the applications and the libpd[2] implementation of Pure Data was used to build the audio engines of these applications.

B. Audio engine

The audio engine is the core part of the mobile application. As mentioned before, it is based on the HRTF introduced [**?**], which is claimed to be both efficient and of reasonable quality in order to achieve a satisfying user experience and an efficient real time application. The block diagram given in [**?**] and depicted in Figure **??** was followed in order to implement it.

[1] https://puredata.info/downloads/earplug
[2] http://libpd.cc/

A general description of the block diagram and an individual of each of the blocks follow.

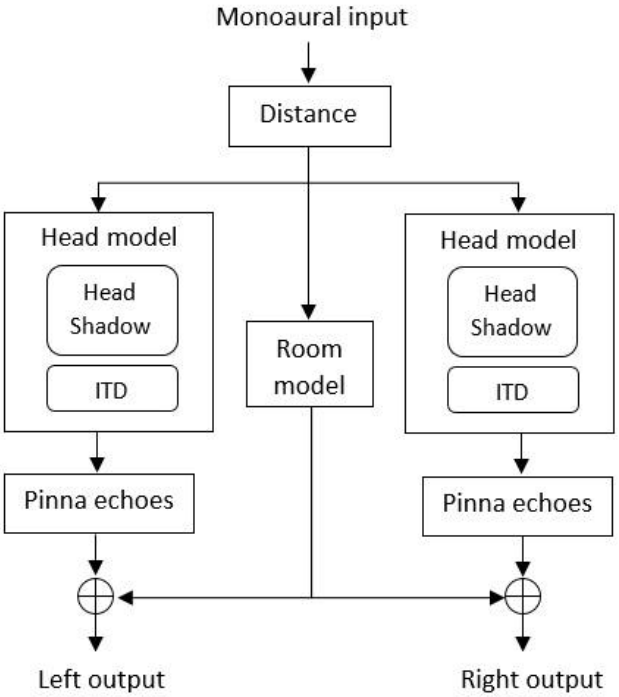

Fig. 1. Components of the model provided on a theoretical basis in [**?**]

First, a monoaural sound source is processed by a distance model, which will feed in parallel a head model and a room model. The head model stereo output is inputed to a pinna echoes model, whose output is added to the output of the room model to obtain the spatialized version of the sound.

As previously mentioned, to implement the head shadow block of the HRTF model, a Pure Data external called *headShadow* was written in the C programming language. So as to go through this source code development, it was necessary to get the correct difference equation of the head shadow block, which is carefully explained in *The head model* section. It was first implemented in MATLAB and then translated into C code. Except for the Pure Data external, the HRTF model was implemented completely using Pure Data.

1) The distance model:
In order to simulate the sound pressure in free field, the inverse distance law was applied.

> "Since sound intensity is proportional to the square of sound pressure, the inverse square law (for sound intensity) becomes the inverse distance law (for sound pressure). Therefore, sound pressure is inversely proportional to distance *r*. [**?**]"

Consequently:

$$P = \frac{k}{r} \tag{1}$$

where *P* is the sound pressure, *k* is a constant and *r* is the distance from source.

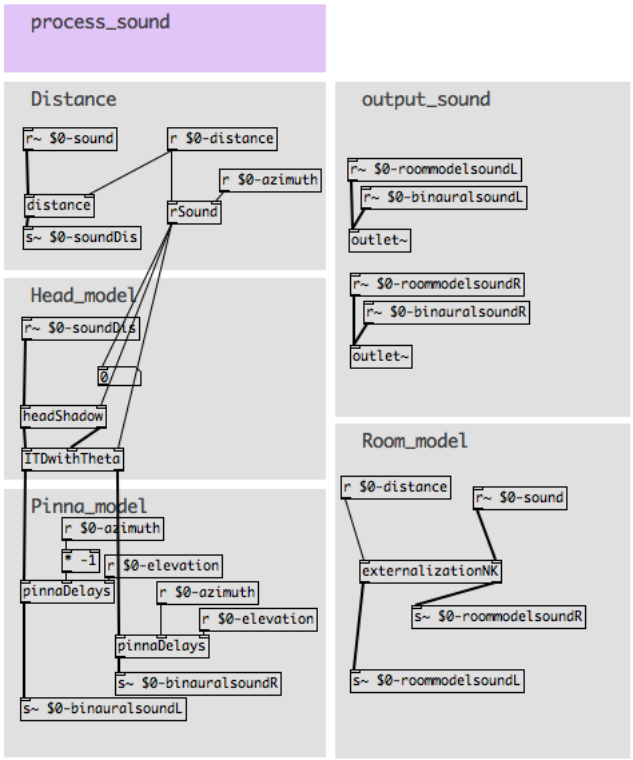

Fig. 2. The audio engine implemented in Pure Data

2) The head model:
The head model is composed of the Interaural Time Differences (ITD) applied to each channel and the and the head shadow effect.

The difference between the paths from the sound source to each of the two ears and the shadowing effect produced by the head at the far ear lead to a delay and a intensity difference in the sound arriving at the left and right ears. In order to estimate the time delay, the Woodworth's formulas were applied [**?**]:

$$ITD = (a/c)[\theta + sin(\theta)]\ [0 \leq \theta \leq \pi/2] \tag{2}$$

$$ITD = (a/c)[\pi - \theta + sin(\theta)]\ [\pi/2 \leq \theta \leq \pi] \tag{3}$$

where a is the approximated head radius in meters, θ is the azimuth angle in radians and c is the speed of sound in meters over seconds. The time differences of the audio signal reaching the head and the ears are therefore:

$$T_L(\theta) = \frac{a + a\theta}{c} \tag{4}$$

$$T_R(\theta) = \frac{a - asin(\theta)}{c} \tag{5}$$

where T_L and T_R are the time delays to reach the left and the right ear, respectively.

These formulas refer to a source placed in front of the head and on the right, with azimuth $0 \leq \theta \leq \pi/2$. If the source is placed on the left ($-\pi/2 \leq \theta \leq 0$), the expressions are reversed.

The head shadow effect is characterized in the following analog transfer function taken from [**?**]:

$$H(s,\theta) = \frac{\alpha(\theta)s+\beta}{s+\beta},\ where\ \beta = \frac{2c}{a} \quad (6)$$

Since it is an analog transfer function, it was derived to a digital version. Therefore, the following transfer function was obtained [3]:

$$H(z,\theta) = \frac{2\alpha(\theta)+T\beta+z^{-1}(-2\alpha(\theta)+T\beta)}{2+T\beta+z^{-1}(-2+T\beta)} = \frac{Y(z)}{X(z)} \quad (7)$$

And, hence, the following difference equation[4]:

$$Y[n] = \frac{a_0X[n]+a_1X[n-1]-b_1Y[n-1]}{b_0} \quad (8)$$

where $a_0 = 2\alpha(\theta)+T\beta$ and $a_1 = -2\alpha(\theta)+T\beta$ as well as $b_0 = 2+T\beta$ and $b_1 = -2+T\beta$ are the filter coefficients.

3) The pinna model:
High frequency components reaching the listener's ears are affected by the pinna surface, which provides elevation cues and some azimuth information [**?**]. However, in the mobile application the elevation is not considered since all the sound sources are placed in the horizontal plane.

The pinna model has the following form:

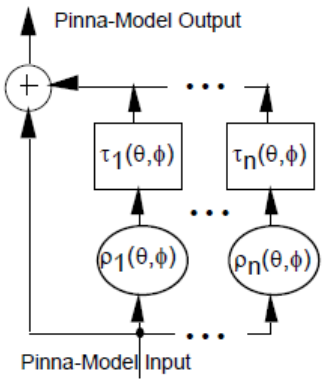

Fig. 3. Pinna model. Image taken from [**?**]

where the ρ_k are the reflection coefficients and the τ_k are the time delays of the *k*th event of a total of *n*. Informal listening tests showed that 5 events were enough to represent the pinna response and that it was convenient to use constant values for the amplitudes ρ_k, independent of azimuth, elevation and the subject [**?**].

The time delays seem to be properly approximated by the following formula:

$$\tau_k(\theta,\phi) = A_k cos(\theta/2) sin(D_k(90^\circ-\phi)) + B_k \quad (9)$$

In this equation, dependent on the azimuth and elevation, the A_k is an amplitude, B_k an offset and D_k is a scaling factor that should be adapted to individual listener. In the following table one can see the values for the parameters used in the pinna model. Only one set of values for D_k in the application have been used.

[3]For the derivation of Equation **??** see appendix A
[4]For the derivation of Equation **??** see appendix B

TABLE I
PINNA MODEL COEFFICIENTS

k	ρ	A_k	B_k	D_k
1	0.5	1	2	1
2	-1	5	4	0.5
3	0.5	5	7	0.5
4	-0.25	5	11	0.5
5	0.25	5	13	0.5

4) The room model: In [**?**], it is implied a very simple room model consisting of only one delay with variable delay time and level according to the distance. An attempt to make a similar model was made by setting the delay time to the difference between the direct path from the object to the listener and a path that bounces on the ground half way between the object and the listener. Finally a more complex model was used due to the poor experienced effect and the large alteration (comb filtering) of the sound quality.

The chosen model is based on the reverb algorithm *rev2* implemented in Pd. Compared to the original model, the amount of early reflections were reduced and spread more out. Through this, a very large but little reflective room was simulated, with a subtle tail, resembling the characteristics of an outdoor environment. **COMMENT:Llus does not understand the following two sentences, could you write them in another way? and use an alternative to "the 2/d"?** The amount of reverb was diminished according to the distance between the user and the virtual sound source. This happens significantly slower than the $2/d$ of the direct sound, namely $8/(15+d)$. In this way the ratio between the direct and the reflected signal changes even though the total level goes down. This model approximates the test data in [**?**]. The *rev2* has a direct through signal incorporated, but this was removed in order for an easier control for the dry and wet signal relation.

COMMENT: could this be in the section future improvements? A further improvement could implement variable early reflections delay-times. And separate control over reverb tail and early reflections level.

C. Sensor data retrieval

The implemented mobile application required two kind of input variables to encode the specific location of the sound source in the real world, namely the distance between the sound source and the user, as well as the direction from which the sound source was coming from. This information could be obtained by accessing the GPS[5] and orientation sensors of the mobile device.

To access the GPS data *ofxMaps*[6] and *ofxGeo*[7] addons were

[5]GPS is a navigation system which provides location information and most of the mobile devices nowadays include a GPS receiver. As explained in [**?**] **COMMENT:[ask Mattia where to find this reference]** the GPS receiver calculates its position by precisely timing the signals sent by the GPS satellites.
[6]https://github.com/bakercp/ofxMaps
[7]https://github.com/bakercp/ofxGeo

used.[8] With the GPS position of the user it was possible to estimate the distance between the user and the sound source. This computation was done using the *Haversine* formula[9], which expected as input arguments the GPS location of the user (provided by the GPS data of the phone) and the sound source:

$$a = \sin^2(\Delta\varphi/2) + \cos\varphi_1 + \cos\varphi_2 + \sin^2(\Delta\lambda/2) \quad (10)$$

$$c = 2 * atan2(\sqrt{a}, \sqrt{(1-a)}) \quad (11)$$

$$d = R * c \quad (12)$$

where φ is the *latitude*, λ is the *longitude*, R is earth's radius (mean radius = 6,371 km).

In order to compute the azimuth, which is the angle from which the sound source should appear to come from, first, the orientation of the mobile device to the magnetic north pole was extracted.[10]

Having the device's position relative to the magnetic North Pole and the sound location the azimuth could be computed using the following method. First of all, the cartesian plane is taken into account and its origin is defined as the device GPS coordinates. Additionaly, it is assumed that the sound location is in the first quadrant. Therefore the sound orientation on the horizonthal plane must be computed and refered to it as *beta*. In order to compute the angle *beta*, a triangle was defined using the mobile device coordinates, the sound coordinates and an immaginary point coordinates. The point's coordinates were defined as the latitude of the mobile device and the longitude of the sound. Second of all, the length of each side of the triangle was computed using the *Haversine* formula. Using the *asin* function the angle *gamma* can be computed which is then substracted by 90 degree and therefore defining *beta*. To compute the azimuth, *beta* is then substracted by the device's position relative to the magnetic North Pole (see appendix for pseudo-code).

As a last step before sending the data to the audio engine, simple mean filters were applied to smooth the changes in the raw data. Experimentation showed that a buffer size of 49 samples for the distance and averaging over 7 samples for the azimuth offered a good trade-off between delay of feedback to the users actions, and not having sudden jumps in distance or orientation (see figure **??**). Additionally, an outlier filter was applied to the raw GPS data. In case there was a jump in the GPS data resulting in a difference of two consecutive distance values of more than 30 meters, that outlier was ignored and the last known GPS location was used.

[8]The GPS position on an Android platform requires a listening mechanism, which has been implemented in openFrameworks by calling the `ofRegisterGPSEvent()` in the `setup()` method of the source code. In the `ofApp.h` a method was added in order 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 be called when quitting the application to avoid an over consumption of phone battery.

[9]http://www.movable-type.co.uk/scripts/latlong.html

[10]For that, the Android API provides access to the TYPE_ORIENTATION sensor which, apart from other things, returns the orientation of the mobile device to the magnetic North Pole [**?**].

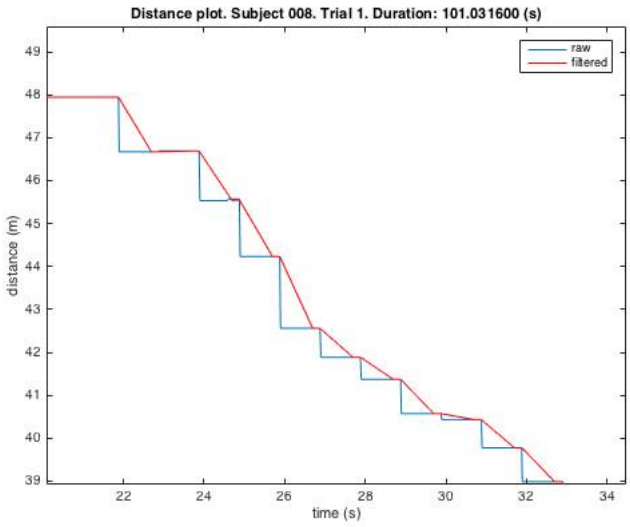

Fig. 4. Comparison between raw and averaged distance

D. User interface

The user interface was designed for the purpose of running the tests mentioned in the introduction. There was one button to start the training trials, and a second one to start the testing trials.

Since the interface was built only for testing purposes and the listener didn't need to interact with it, the interface was not developed in a user-friendly way (see figure 5). The user interface has not yet been specifically designed for the user.

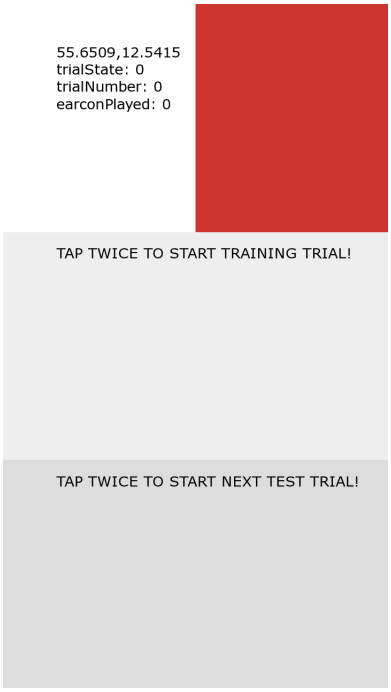

Fig. 5. Application GUI used for the experiment.

III. The experiment

One of the aims of our experiment was to investigate how fast users were able to locate and navigate to a single sound source using our 3-D audio engine compared to a cosine panner based on [**?**] and 2/d distance calculation which served as a baseline performance. Therefore the time needed from the beginning of a trial until the sound source was found was measured.

Additionally we assessed a qualitative analysis of how intuitive it was for subjects to locate the sound as well as investigating whether the sound seemed to be embedded in the real scenery. These results were again compared to the cosine panner model [**?**]. The data was obtained with a 10 point likertscale questionnaire going from ”don’t agree at all” to ”completely agree”.

Results showed no statistically significant results between both models, neither for the quantitative nor qualitative analyses.

A. Materials and methods

1) Participants: Qualitative data were collected from 11 participants (3 women and 8 men; mean age: 25, ranging from 22 to 30 years). For the quantitative data 6 more participants were measured summing up to in total 17 participants (4 women and 13 men; mean age: 25, ranging from 22 to 30 years). All participants were students coming from different backgrounds.[11] Six students were already familiar with 3-D audio sound and four of them had experience in audio navigation before conduction of the experiment. The other participants were naive. All participants reported normal or corrected to normal vision and no hearing deficits. Participants received no form of compensation other than gratitude.

2) Apparatus and stimuli: The experiment took place in a free field (grass) in ”Valby Parken” in Copenhagen, Denmark (Coordinates: 55°38’22.3”N, 12°31’27.4”E). The mobile device used was a Motorolla Moto G running with Android version 5.0.2. The device was connected to Sony MDR ZX600 stereo headphones. Only one person at a time was tested.

Fig. 6. Testing the application in Valby Parken, Copenhagen

[11]However, 14 subjects were master students in Sound and Music Computing at Aalborg University Copenhagen.

To create the auditory stimulus, digital vocal recordings of a male voice (age: 38) were collected using a AudioTechnica AT4040 condenser microphone and Apogee Duet II digital audio interface. Recordings were done with 24 bits of resolution for amplitude at a sampling rate of 44100 kHz. Auditory stimuli have been compressed and equalized using a Maag Eq4 parametric equalizer, Teletronik LA-2A and Vertigo VSC-2 Quad Discrete Compressor plugins. They finally have been normalized in peak amplitude using the Izotope Ozone Maximizer (settings: IRC III algorithm, threshold: -7.0 dB, ceiling: -0.2 dB) so that at the minimal distance to the sound source (7 meter) using the maximal level of volume on the mobile device, the sound level of each sound source was at -4.5 dB SPL (see appendix for stimulus content).

3) Design: The experiment comprised one within subject factor at 2 levels (sound engine). The two levels of the sound engine were our 3-D audio engine (3-D) and a cosine panner [**?**] and distance amplitude modulation (panning).

The starting point was fixed and the distance of the sound source from that point was always 48 meters.[12] The sound could appear at any angle from the starting point.

For all trials the same soundfile was played. Each participant performed a total of 6 trials meaning 3 trials in each of the two conditions. Participants were encouraged to take a short break whenever they felt fatigued.

4) Procedure: The position (angle from the starting point) of the sound sources in the different trials were completely randomized within and between subjects independently from the sound engine. This ensured that participants could not predict the location of the sound in the next trial. The order of trials with the one or the other sound engine was not randomized within subjects. Nevertheless, half of the participants started the experiment with the 3-D audio engine while the other half started with the audio panning sound engine to counterbalance for effects related to training. After each block of trials (one block were 3 trials with one of the sound engines) participants had to fill out the qualitative questionnaire.

Each trial was initiated by the experimenter with a button press after which the sound source was played. No visual feedback about the location of the sound source or the already passed time to locate it was given. For localization, subjects had to rely solely on the auditory cues given via the headphones. /footnoteAdmittedly participants could have used the position of the starting point as some kind of fix point. However, participants did know nothing about the possible distances of the sound source. Additionally, questions after the experiment about applied strategies did not reveal that participants used visual cues for localizing the sound source. As soon as the participants reached a radius of 7 meter around the sound source, an earcon was played indicating the success of the localization. The participants went back to the initial starting point and the next trial was started.

Before the experiment, participants were familiarized with the target sound as well as the earcons that were played

[12]However, because of poor GPS accuracy, the location of the sound source was calculated with the momentary GPS coordinates provided by the phone sensor instead of the actual fixed position in the free field. This ensured that the initial distance to the sound source was always the same.

when having reached the location of the virtual sound source. Additionally, all participants performed at least two practice trials with the sound engine they were tested with first before the actual measurements with that sound engine began. The location in the practice trials were fixed at the same positions for all participants to allow us to guide the subject when they got lost.

Participants were instructed to find the sound source as fast as possible. No information about the possible locations (distances and angles) was given to the subjects. Trials which lasted longer than 5 minutes were aborted and labeled as "not found".

5) Analyses: For the quantitative analysis 26 trials were excluded because either analyses of GPS data for those trials revealed very poor precision (sudden jumps or no changes in the GPS signal for a period of time) or the orientation sensor got stuck which either increased the time needed to find the sound source to a great extent or made it even impossible for the subjects to locate the sound source within 5 minutes. One subject had to be discarded completely from the analyses because there was no single trial with reliable GPS for the panning engine. The other 16 subjects performed at least one trial in each condition. The remaining trials to be analyzed summed up to 41 trials in the 3-D condition and 42 trials in the panning condition.

For the qualitative analysis the questionaires from 11 participants were evaluated using parametric as well as non-parametric methods as there are arguments in favor [**?**], [**?**] and against [**?**] treating likert-scale data as ordinal and/or interval data.

The threshold chosen to correctly reject the null-hypothesis was always 5% (α = .05).

B. Results

1) Quantitative: On average subjects needed 104.6 seconds (SD = 44.8) to find the sounds for the 3-D conditions and 109.8 seconds (SD = 44.1) for the panning condition.

We computed a paired samples t-test to compare the results for both sound engines which revealed no significant difference between the means for both sound engines (t(15) =-.48, p =.64). However, checking whether our data is normally distributed using the Shapiro-Wilk test showed that for the 3-D data the assumption of normality was violated (Z=.81, p = .004) and close to significance in the panning condition (Z=.89, p =.054). Therefore, we additionally computed the non-parametric Wilcoxon signed rank test which does not assume normal distributed data. Nevertheless, we could not find a significant difference in scores for time needed to find the sound source between both conditions (median: 3-D=85.3, panning=95.9; Z=-.83, p = 0.41).

2) Qualitative:

The Shapiro-Wilk test showed that the data was normally distributed both for the 3D model (Z = 0.933, p = 0.44) and the Panning model (Z = 0.95, p = 0.67).

Evaluation of the questionnaires, Table **??**, showed that the mean and median values for questions 1, 2, 3 and 4 for the 3D model were only numerically different from the panning. Neither the paired samples t-test nor the Wilcoxon signed rank test or the sign test showed significant results comparing the two conditions (3-D vs. panning). However, performing a one sample t-test on means of the four questions for each model comparing it with a neutral response (5.5) showed that users rated the connection between their position and what was presented via the headphones better than neutral for the 3-D sound engine (t(10) = 4.32, p =.002; Z =2.74 , p = .006) but not for the panning (t(10) = 1.07, p = .31; Z = .90, p=.367). For all other questions there were no significant differences.[13]

Question	Mean (s)		Std (s)		Median (s)	
	3D model	Panning	3D model	Panning	3D model	Panning
1	6.36	5.91	1.75	1.70	6	6
2	6.91	5.82	2.51	2.27	8	6
3	7.09	6.00	1.22	1.55	7	6
4[14]	6.64	6.09	2.58	1.97	7	5
5	6.09	6.55	2.21	2.54	7	7

TABLE II
QUESTIONNAIRE RESULTS

Summing the results for the first four questions to form an overall category of quality of the two sound engines lead to the mean and median values depicted in Table **??**. Again, the difference between both sound engines were only numerically visible since performing a paired samples t-test and a Wilcoxon signed rank test showed no significant results (t(10) = 1.05, p = 0.32; Z = -1.07, p = .283). Furthermore, running a one sample t-test and wilcoxon signed rank test to compare the results of both sound engines to a mean (5.5) that represents being neutral towards a question showed no significant results for both conditions (HRTF: t(10) = 1.54, p = .154; Z = 1.54, p = .13; Panning: t(10) = -.22, p =.829; Z = -0.5, p = .96). However, summing only the first three questions, showed that the 3-D model ratings were significantly higher than the neutral mean (5.5) (t(10) = 2.83, p = .018; Z = 2.09, p = .036) which was not the case for the panning condition (t(10) = .86, p = .41; Z = 0.76, p = .447).

Question	Mean (s)		Std (s)		Median (s)	
	3D model	Panning	3D model	Panning	3D model	Panning
1 - 3	6.79	5.91	1.51	1.58	7.33	5.67
1 - 4	5.93	5.41	0.93	1.36	6	5.25

TABLE III
OVERALL RESULTS

IV. DISCUSSION

The results show that both sound engines did not differ in terms of performance. Although numerically subjects were slightly faster to locate the sound source when using the HRTF audio engine, this difference could not be verified statistically. This comfirms that the 3-D audio engine at least partly fulfilled

[13]It is worth noting that choosing mean and median as 5 instead of 5.5, showed that also for question two, a one sample t-test and the wilcoxon signed rank test were significant for the 3-D (t(10) = 2.52 , p = .03 ; Z = 2.19, p =.028) but not for the panning sound engine (t(10) = 1.19, p = .26 ; Z = 1.2, p =.23).

[14]It should be noted that for this question lower values represent higher quality of a 3-D sound because of the direction of the question.

the aims, namely perfoming at least as good as a simple panning. It is also possible that our sample size was so small that the difference in performance would have had to be bigger to show significant results. As noted by several methodologists Type II errors have a high probability to occur for low sample sizes [**?**]. On top of the small sample size, an unreliable GPS source (different number of GPS satellites fixes for each experiment, with mean of fixes: 11 ranging from 5 to 12) and different weather conditions (subjects complained about wind noises interfering with the auditory display of the soundsource) could have been responsible for the large variance in the data and thus reducing the power of our statistical test [**?**].

As for the qualitative results looking at the histograms of each questions shows that the differences between the ratings of each participants varied to a great extent. **COMMENT:Lars[Here we can discuss a little more the results of the hist? subjective?]**. Nevertheless, no statistical differences in user ratings between both audio engines could be found. An explanation for why the 3-D audio engine was not rated better than the panning might have been the choice of a unnatural high reverberation level for a free field in the HRTF audio engine. This is inspired by the fact that a lot of participants had background knowledge in acoustics and therefore made more predictions about a proper reverberation level for a free field. Some subjects even reported the reverberation to be the reason for low ratings of the question regarding whether the sound source was perceived to be embedded in the real scenery. This negative effect on the ratings might have nullified the difference between panning and HRTF in terms of perceiving the sound moving around and not inside the head.

Participants might have had a great top-down expectation of hearing the sounds coming from the headphones rather than being externalized meaning coming from a specific location in the real world. This might have negatively affected the experience of an externalized sound especially considering the lack of a visual representation of the soundsource.

Nevertheless, it should be noticed that the ratings of the first three questions about the quality of the audio engine taken together (as well as the third question alone which was asking about whether subjects felt the sound to come from outside of their head) were significantly higher than a neutral response for the HRTF but not for the Panning. Figure **??** shows the ratings for the first to third questions taken together, see the appendix **WHICH ONE?** to see the responses to each questionnaire question. At a first glance this might seem contradictory and has to be considered with caution since panning and HRTF ratings were not significantly different from each other. However, the finding that the HRTF was rated higher than the mean while panning was not could be interpreted as an indication of the superiority in quality of the HRTF over the panning audio engine, especially in rendering the perception of a connection between the users position and the feedback given via the headphones.

Although, GPS data was filtered the mapping from the actions of the participant to what was registered by the mobile device and given as input to the audio engine might not have been accurate enough for a convenient experience disregarding the audio engine used. Following this logic, even a perfect

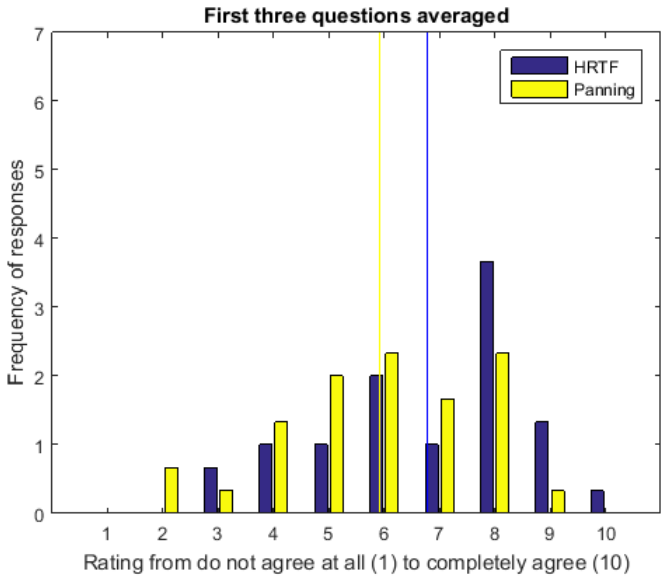

Fig. 7. Distribution of answers for the first three questions

3-D audio engine could not have resulted in higher ratings for questions about easyness and intuitiveness to locate the soundsource.

Another issue that should be addressed is the adequacy of the questionnaire used in this experiment. In the present experiment a 10-point likertscale instead of a 5- or 7- point likertscale was used to achieve a more fine grained rating (higher precision). Since we only had few questions, having so many answer possibilities should not have fatigued the subject too much so that accuracy of the responses should not have suffered due to too many answer possibilities. Though, it has to be noted that subjects were outside when filling out the questionnaire which due to cold temperatures might have driven the subjects to respond in a fast pace leading to less accurate responses. Also the fact that ratings of most of the questions were not significantly different from the mean of the scale itself (indicating being neutral towards the posed question) provides some evidence that subjects were either actually indifferent to the posed questions or had difficulties of knowing which number corresponds to which degree of agreement to the stated question. Additionally, it has been reported that 10-point likert scales produce significantly lower mean values than 5- or 7-point likertscales [**?**]. The just mentioned issues might sum up suggesting that it is rather the method used to assess the qualitative data than the quality of the sound engines themselves that lead to the qualitative results.

In light of all these considerations the attention should not be drawn away from the fact that the 3-D audio engine is far from being perfect. More thorough evaluation of reverberation levels and distance models can be considered to improve the user experience. Additionally more sophisticated and accurate sensors to track the position of the user relative to the virtual soundsource could enhance the responsiveness of the audio engine. Thereby attention should be payed to the tradeoff between the quality and accuracy of the audio engine itself

and the sensors i.e. which level of accuracy of the sensors is needed to meet up with the accuracy of the 3-D audio engine.

A. Future improvements

Undoubtedly, much refinement and improvement should be made to improve both the quality and the performance of our application. As a prototype, we concentrated mainly on the core, that is to say the programming and sound processing aspects. However, we are conscious that developing a clear and compact GUI is as much important in terms of accessibility to users. In this section several possible improvements will be given.

First of all, using multiple sound sources could allow for more interesting and immersive environments. We all are familiar to surroundings made of several items, whose combinations are almost endless. Let's take a crowded place as example. In such similar situation, the user would be given a bunch of different sounds placed in different locations. This is also a crucial point in developing applications such as *Events in the City* and *Audio Tour Guide*.

As said in previous sections, in some experimental situation listeners could make *front-back* discrimination errors. Literature has already dealt with it and has provided numerous suggestions to solve that problem, as depicted in [**?**]and [**?**].

Use and choice of the right reverb is another crucial point. Even though its choice could be mostly affected by ease of implementation and computational cost, we would aim for an adaptive one, which could be capable of give back a more realistic environment model. Indeed, the choice of a convolution reverb would be the most appropriate since any kind of space could be precisely modelled using its frequency response. On the other hand, it has a huge computational cost, which maybe not all mobile devices couldn't afford. That said, an alternative model of this application should be thought, which all the computing part is shifted on a specific machine. Doing so, the mobile application could be fed directly on externalized sounds.

Finally, an improvement in graphics is encouraged in to let the application be more appealing and *user-friendly* as well as a code refinement operation to eliminate any superfluous tasks, let all the application be lighter and take advantage of the new technologies structure introducing multithreading.

V. Conclusion

Acknowledgments

Appendix A
Derivations

The analog transfer function has to be derived to the digital version by applying a bilinear transform:

$$s = \frac{2}{T}\frac{z-1}{z+1},\ where\, T\, is\, the\, sampling\, interval\, in\, seconds \tag{13}$$

Applying the substitution in equation **??** to equation **??**, following filter function in the digital domain is obtained:

$$H\left(z,\theta\right) = \frac{\alpha(\theta)(\frac{2}{T}\frac{z-1}{z+1}) + \beta}{(\frac{2}{T}\frac{z-1}{z+1}) + \beta} \tag{14}$$

The derivations followed to get equation **??** can be seen as follows

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

$$\begin{aligned}
H(z,\theta) = \frac{Y(z)}{X(z)} &= \frac{a_0 + a_1 z^{-1}}{b_0 + b_1 z^{-1}} \\
&\iff Y(z)(b_0 + b_1 z^{-1}) = X(z)(a_0 + a_1 z^{-1}) \\
&\iff Y(z)b_0 + b_1 Y(z)z^{-1} = a_0 X(z) + a_1 X(z)z^{-1} \\
&\iff Y(z) = \frac{a_0 X(z) + a_1 X(z)z^{-1} - b_1 Y(z)z^{-1}}{b_0} \\
&\to Y[n] = \frac{a_0 X[n] + a_1 X[n-1] - b_1 Y[n-1]}{b_0}
\end{aligned}$$

APPENDIX B
OPENFRAMEWORKS

OpenFrameworks[15] has been chosen as development framework rather than Android Studio[16]. This choice is due to cross-platform reasons. Even though the prototype application runs only on Android, future improvements will also inlcude iOS development. Discussing all the steps involved in the building process is beyond the scope of this paper, but a short explanation of openFrameworks and the main operations used to achieve the application will be given.

1) Description:
OpenFrameworks, is according to their website:

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 is massively cross-compatible. It gives the opportunity to deal with code designed to be minimal and easy to grasp [**?**]. 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. Recently this has become a popular platform for experiments and creation of generative art, sound art, interactive installations and audiovisual performances [**?**]. The current operative version is *0.9.0*.

2) Addons:
Its design philosophy aims for a collaborative environment. It thrives from the contributions of many people, that collaborate mainly on addons and projects. An *addon* is made of several snippets of code put together in order to extend the functionality of openFrameworks, allow for external framworks to be integrated into openFrameworks project or make specific and complicated tasks easier and reusable in other projects []*ask Mattia where to find this reference*. In this app, several *third-party* addons has been used such as *ofxGui, ofxXmlSettings, ofxAndroid, ofxGeo, ofxMaps, ofxTween, ofxPd*. Additionally, there has been implemented a specific addon called *ofxOrientation*, which accesses sensory data regarding orientation. This step was necessary in order to retrieve the required angle between the sound location and the user orientation, which is then use by the HRTF model.

3) The openFrameworks project:
All openFrameworks project have a similar structure of folders and files. The most important folder among them is 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, hence they are triggered in response to events that happens inside the application such as mouse scrolling and program quitting.To create a new project, the Project Generator wizard has been used which is 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. 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.

[15]http://openframeworks.cc/
[16]http://developer.android.com/sdk/index.html

APPENDIX C
DISCARDED DATA

Due to unreliability of the GPS and orientation sensors, some data was discarded. The process of deciding which trial data was not useful consisted, first, in identifying outliers in the time durations of each audio model, for each participant. And, second in evaluating if distance or orientation data of these trials was constant in a part of the trial. The following figures show the discarded trials and the reason for doing it.

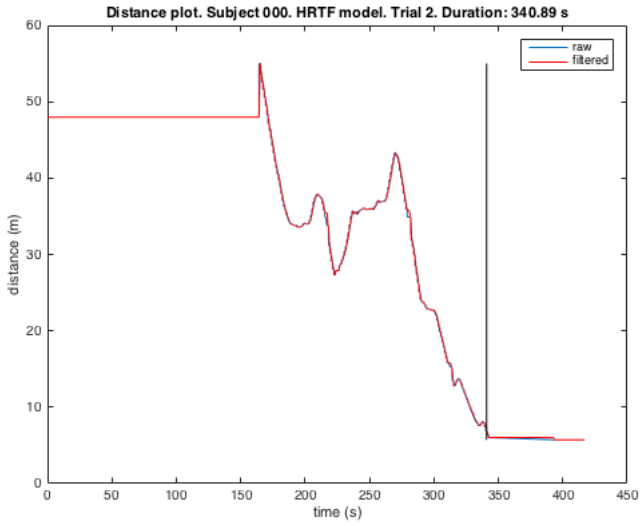

Fig. 8. Distance (related to the GPS sensor) remains constant the first 164 seconds. Vertical line indicates the time when the subject found the sound.

APPENDIX D
STIMULUS CONTENT

"Hello, I am your test guide. If you can listen to my voice you are not far from me but you haven't found me yet. So, your task is to find where I am speaking as fast as you can. Please use your hearing and let you be guided through. Enjoy your search."

APPENDIX E
COMPUTING THE AZIMUTH

Data: C++ pseudo-code to compute the azimuth:
new Latitude = myGPS Latitude - myGPS Latitude;
new Longitude = myGPS Longitude - myGPS Longitude;
newCoord = Geo::Coordinate(new Latitude, new Longitude);
soundLat = soundGPS Latitude - myGPS Latitude;
soundLong = soundGPS Longitude - myGPS Longitude;
soundCoord = Geo::Coordinate(soundLat,soundLong);
pointLat = new Latitude;
pointLong = abs(soundLat);
pointCoord = Geo::Coordinate(pointLat,pointLong);
adjacent = Geo::GeoUtils::distanceHaversine(pointCoord,soundCoord);

opposite = Geo::GeoUtils::distanceHaversine(pointCoord,newCoord);

hypo = Geo::GeoUtils::distanceHaversine(newCoord,soundCoord);

gamma = asin(opposite/hypo);
if *the sound is in the first quadrant* **then**
  beta = 90 - gamma;
end
else if *if the sound is in the fourth quadrant* **then**
  beta = 180 - gamma;
end
else if *if the sound is in the third quadrant* **then**
  beta = 270 - gamma;
end
else if *if the sound is in the second quadrant* **then**
  beta = 360 - gamma;
end
if *beta is bigger than 180* **then**
  if *abs(beta - Angle to the North) is bigger than 180* **then**
    azimuth = abs(beta - Angle to the North) - 360;
  end
  else
    azimuth = beta - Angle to the North;
  end
  else
    if *abs(beta - Angle to the North) is bigger than 180* **then**
      azimuth = abs(abs(beta - Angle to the North)) - 360;
    end
    else
      azimuth = beta - Angle to the North;
    end
  end
  azimuth = round((azimuth * (PI / 180)));
  return azimuth;
end

Algorithm 1: algorithm that computes the azimuth

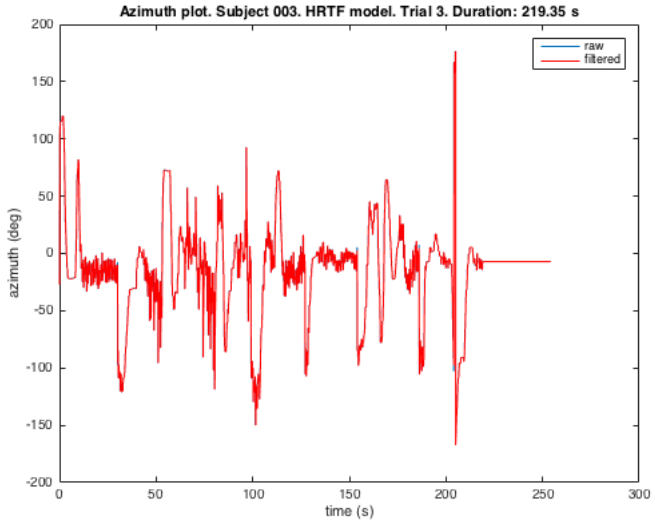

Fig. 9. Orientation is unreliable from 10 to 30 seconds

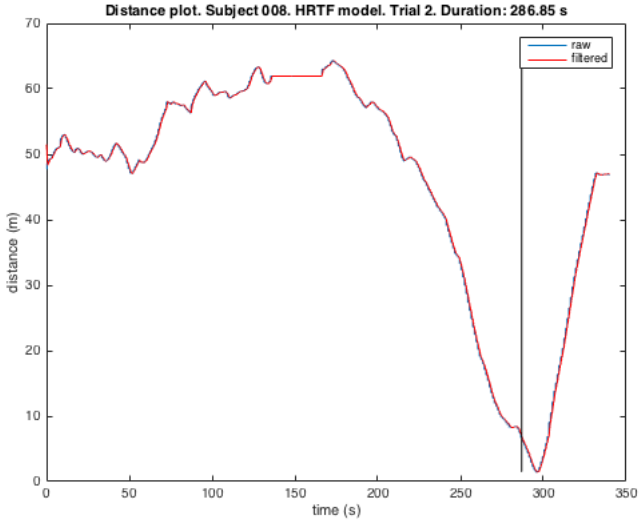

Fig. 10. Distance is constant for 30 seconds starting at mark 135 second

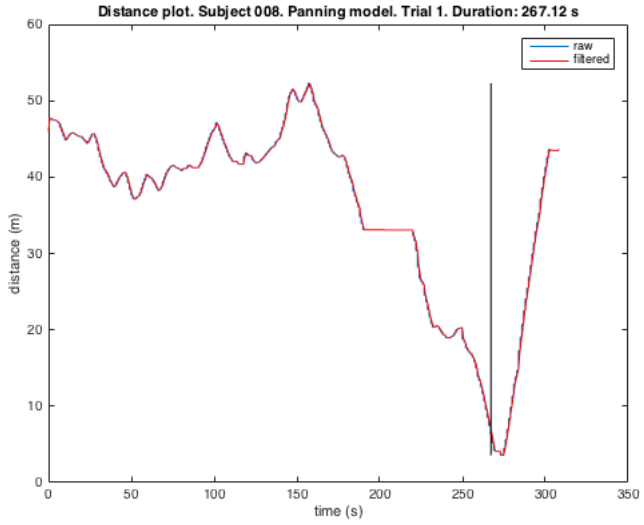

Fig. 11. Distance is constant for 30 seconds starting at mark 190 second

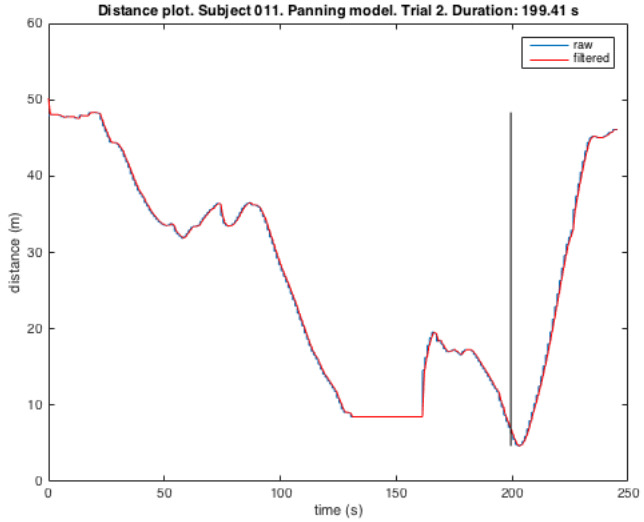

Fig. 12. Distance is constant for 30 seconds starting at mark 131 second

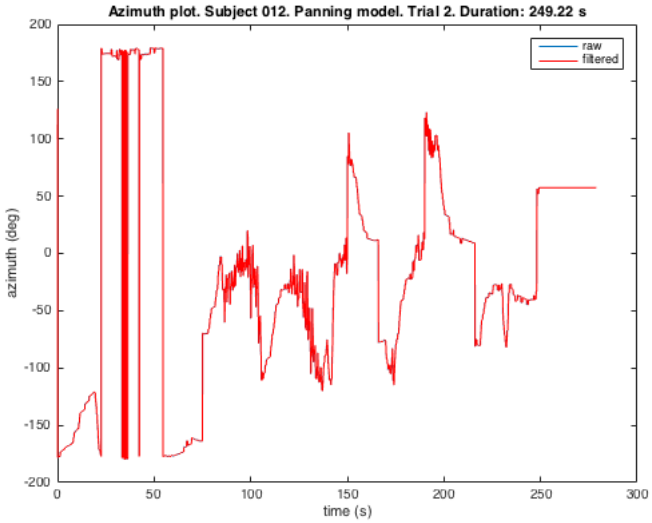

Fig. 13. Orientation is not reliable from 20 to 75 seconds