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*—IEEEtran, journal, LaTeX, paper, template.**

I. Introduction

A lot of research on modelling 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 impulseresponses 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.

First, some state of the art research on 3-D sound 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 hypothesises stated above.

December, 2015

II. State of the art in 3-D sound

Here it would be nice to put some state of the art research about HRTF's etc.

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)

III. The application

We decided to develop our application for android. Nevertheless, we did not use the API provided by Android Studio but instead used openFrameworks which is a ..... (reference). The reasons why we decided to develop using this programming environment was the need of a fast communication between the mobile device and our programm code. Since programming in C can be directly translated into machine code and does not need to be interpreted by a virtual machine as it is the case for java (reference) we hoped to increase the speed of our application to fulfill the needs of a realtime application. Additionally openFrameworks as open source project provides a multitude of addons enabling us to benefit from already solved problems. (include a list of addons we used maybe). As last but rather minor point as far as our motivation for this application was concerned it should be mentioned that programming independent from a specific API such as those provided for Android or iOS makes modifying our application to run on a different operating system easier as the main code is written platform independent.

A. Sensory data extraction

Our application required two main sensory information. We needed the GPS location of the user to calculate the distance to the sounds that were placed on fixed GPS locations

in addition to the angle the user is heading relative to the sound sources. To access the GPS data we used an already existing addon called ?ofxMaps and ofxGeo? (reference to webpage/gitrepository). (Something about smoothing if we used it, and maybe we have to look into the code how they used smoothing). To calculate the angle between the heading of the user and the sound source we had to extract the bearing between our location and the location of the sound (reference and some explanation about bearing maybe?) as well as the heading of our mobile phone relative to the magnetic south (geographic north pole). The bearing could be accessed via the ofxGeo addon. 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 explenation about how we calculatet the angle between our heading orientation and the sound). In order to smooth the sensory data (especially for the orientation) we filtered the sensory information by computing the median of a chunk of mediansize instances of the sensor output.

B. Audio engine

The audio engine was the most crucial part of our applications. As shortly mentioned in the introduction (and/or state of the art in 3-D sound) 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 ... .

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:

$$H\left(s,\theta\right) = \frac{\alpha(\theta)s+\beta}{s+\beta},\ where\ \beta = \frac{2c}{a} \tag{1}$$

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

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

Applying the substitutin in Equation 2 to Equation 1 we get the following filter function in the digital domain:

$$H\left(z,\theta\right) = \frac{\alpha(\theta)(\frac{2}{T}\frac{z-1}{z+1})+\beta}{(\frac{2}{T}\frac{z-1}{z+1})+\beta} \tag{3}$$

To identify the filter coefficients we transformed Equation 3 and obtained the following frequency response[1]:

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

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 $b_1 = -2+T\beta$.

[1]For the derivation of Equation 4 see appendix A

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 4 as well as going from the frequency to the time domain. The result is given in Equation 5[2]:

$$Y[n] = \frac{a_0X[n]+a_1X[n-1]-b_1Y[n-1]}{b_0} \tag{5}$$

C. User interface

include pictures of user interface etc.

IV. Discussion

V. Conclusion

Acknowledgment

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

[2]For the derivation of Equation 5 see appendix B

LIST OF FIGURES

APPENDIX A
DERIVATIONS

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

APPENDIX B
DERIVATIONS

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