HELSINKI UNIVERSITY OF TECHNOLOGY
Faculty of Electronics, Communications and Automation
Department of Signal Processing and Acoustics

Mikko Peltola

Augmented Reality Audio Applications in Outdoor Use

Master's Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Technology.

Espoo, Feb 23, 2009

Supervisor: Professor Matti Karjalainen
Instructor: Adjunct professor Tapio Lokki, D.Sc.(Tech.)

HELSINKI UNIVERSITY OF TECHNOLOGY

ABSTRACT OF THE MASTER'S THESIS

Author:	Mikko Peltola
Name of the thesis:	Augmented Reality Audio Applications in Outdoor Use
Date:	Feb 23, 2009 **Number of pages:** 66
Faculty:	Faculty of Electronics, Communications and Automation
Professorship:	S-89
Supervisor:	Prof. Matti Karjalainen
Instructor:	Adjunct professor Tapio Lokki, D.Sc.(Tech.)

Augmented Reality Audio (ARA) can be defined as the real audio environment augmented with virtual sound objects. Everything the user hears is recorded with miniature microphones integrated in the ARA headset earphones. The recorded audio is immediately played back directly to the ears of the user. Along with the recorded signal, artificial or previously recorded sounds can be added. Binaural processing is used to position the virtual sounds in certain directions related to the user. The virtual sound sources can be fixed to the real environment with binaural processing, if the position and orientation of the user are known.

Several ARA applications have been implemented before, but so far only to be used indoors, because of the limitations of applied tracking methods. The goal of the thesis was to design and implement a platform to be used outdoors. Inertial and magnetic tracking were used together to track the orientation of the user, and Global Positioning System (GPS) was used for position tracking.

As a proof of concept, the Audiomemo application was implemented on the platform. Audiomemo can be used to record everything the user hears, and to enable the user to browse through previously recorded audio memories. The orientation and position information is saved as metadata in the audio file to enable retrieval of the information afterwards.

The Audiomemo application was preliminarily tested and analyzed. The platform was found functional and a useful working base for ARA applications in outdoor use.

Keywords: Augmented Reality Audio (ARA), Orientation tracking, Position tracking, Binaural hearing, Wav format, Ara-Wav

TEKNILLINEN KORKEAKOULU DIPLOMITYÖN TIIVISTELMÄ

Tekijä:	Mikko Peltola
Työn nimi:	Lisätyn audiotodellisuuden sovellukset ulkokäytössä
Päivämäärä:	23.2.2009 **Sivuja:** 66
Osasto:	Elektroniikan, tietoliikenteen ja automaation tiedekunta
Professuuri:	S-89
Työn valvoja:	Prof. Matti Karjalainen
Työn ohjaaja:	Dosentti, TkT Tapio Lokki

Lisätyksi audiotodellisuudeksi (LAT) kutsutaan todellista äänimaisemaa, johon on lisätty virtuaalisia ääniobjekteja. Käyttäjän kuulema ääni tallenetaan LAT-kuulokkeisiin integroiduilla miniatyyrimikrofoneilla ja toistetaan tämän jälkeen suoraan käyttäjän korviin. Nauhoitetun signaalin päälle käyttäjälle voidaan toistaa keinotekoista tai aikaisemmin nauhoitettua signaalia. Binauraalisella prosessoinnilla lisätyt, virtuaaliset, äänilähteet voidaan sijoittaa haluttuun suuntaan. Virtuaaliset äänilähteet voidaan sitoa todelliseen äänikenttään binauraalisella prosessoinnilla, jos käyttäjän sijainti ja orientaatio tunnetaan.

Käyttäjän jäljittämiseen käytettyjen menetelmien rajoituksista johtuen aikaisemmin toteutetut LAT-sovellutukset on tarkoitettu vain sisäkäyttöön. Tämän diplomityön tavoitteena oli suunnitella ja toteuttaa alusta ulkona käytettäviä LAT-sovellutuksia varten. Sekä inertiaan että maan magneettikenttään perustuvia menetelmiä käytettiin käyttäjän orientaation ja satelliittipaikannusta (GPS) paikan määrittämiseen.

Alustan toimivuuden osoittamiseksi toteutettiin äänimuistisovellus suunnitellulla alustalla. Äänimuistia käytetään tallentamaan kaikki käyttäjän kuulema ääni, paikka- ja orientaatiotiedolla laajennettuna. Paikka- ja orientaatiotieto tallennettiin metatietona äänitiedostoon uudessa Ara-Wav formaatissa, jotta tietoa voidaan käsitellä myöhemminkin.

Äänimuistisovellusta testattiin ja analysoitiin, minkä pohjalta alusta todettiin toimivaksi lähtökohdaksi ulkokäyttöön tarkoitettujen LAT-sovellusten toteuttamiseen.

Avainsanat: Lisätty audiotodellisuus (LAT), suunnan määritys, paikannus, binauraalinen kuulo, Wav formaatti, Ara-Wav

Acknowledgements

This Master's thesis has been done for the Department of Signal Processing and Acoustics in the Department of Media Technology and funded by Nokia Research Center.

I want to thank D.Sc. Tapio Lokki for his patient and excellent guidance and assistance with the thesis. The same applies to Lauri Savioja, my official boss in the department.

I wish to thank all my co-workers in the department, especially Raine, Antti, Inger and Hanieh. Thanks for all the help and support. And for all that coffee.

I would also like to thank my collegues at Loiste for understanding my absence from 'the real work'. Especially Heidi for helping me with the figures in the thesis. And my dear friend Reetta for the proofreading.

My gratitude also goes to Julia Turku, Riitta Väänänen, Matti Hämäläinen and Johan Kildal from Nokia Research Center. And to Miikka Tikander and Matti Karjalainen from the Department of Signal Processing and Acoustics.

Finally, I would like to thank my dear family and loved ones for everything; I would not have been able to do this with out your support! Last spring I tried my best to understand Hannele and her strugles with her thesis. Now I finally do. Thank you for all the love and help!

Otaniemi, March 23, 2009

Mikko Peltola

Contents

Abbreviations

ARA	Augmented Reality Audio
DOF	Degree of Freedom
GPS	Global Positioning System
HRTF	Head Related Transfer Function
IC	Interaural Coherence
IID	Interaural Intensity Difference
ILD	Interaural Level Difference
ITD	Interaural Time Difference
KAMARA	Killer Applications for Mobile Augmented Reality Audio
NRC	Nokia Research Center
MARA	Mobile Augmented Reality Audio
PD	Pure Data
SHAKE	Sensing Hardware Accessory for Kinesthetic Expression
TML	Telecommunications and Software and Multimedia Laboratory
WARA	Wearable Augmented Reality Audio
WGS84	World Geodetic System 1984

Chapter 1

Introduction

This thesis has been created as a part of KAMARA++ (Killer Applications for Mobile Augmented Reality Audio) project. KAMARA++ is continuation to a series of KAMARA projects carried out as a joint effort between Nokia Research Center (NRC) and Helsinki University of Technology Departments of *Signal Processing and Acoustics* – formerly known as *Laboratory of Acoustics and Audio Signal Processing*, or *Acoustics Lab* – and *Media Technology* – formerly known as *Telecommunications and Software and Multimedia Laboratory* (*TML*).

1.1 KAMARA

A framework for all KAMARA projects have been introduced in [1]. The base of the studies is the idea of having real audio environment extended by virtual sounds creating augmented audio reality. KAMARA projects focus especially on Mobile Augmented Reality Audio (MARA), first introduced as WARA, Wearable Augmented Reality Audio in [2] with the concept and a prototype system; a headset system with integrated microphones was presented. The binaural audio recorded by the microphones could be immediatelly played back to the user creating a pseudo-acoustic audio environment. Also virtual sounds were added to create an augmented audio environment. The application ideas for the MARA framework have been presented for example in [2, 1] and [3].

The first stage of the KAMARA projects was carried out between 2001 and 2004, with research concentrated on ideas and concepts of MARA, but also on analysis of recorded binaural signals; i.e. reverberation time estimation [4] and binaural position and orientation tracking [5]. After the first stage the research has been continued by M. Tikander and S. Vesa as doctoral students with academic funding. In autumn 2007 KAMARA+ project started in cooperation with NRC, Acoustics Lab and TML with two subtasks; one was

to implement Augmented Reality Audio Mixer (ARA Mixer) and headset to be used with ARA applications [6], and the other one to implement Auditory Sticker application on video based tracking [7].

1.2 Objective and scope of the thesis

All the previous implementations of applications and research topics in KAMARA projects have been limited to be used indoors and within fixed location. The aim with this thesis is to design and implement a platform for ARA applications to be used outdoors. As a proof of concept the Audiomemo application – an application to record and browse all the audio the user is hearing – has been implemented. The platform is intended to be a tool for future application prototyping with a possibility to easily extend and improve the platform. The platform is not strictly tied with the current technology, but when the tracking methods improve the used equipment can be updated accordingly.

1.3 Organization of the thesis

In Chapter 2 all the required theory to understand the decisions made and the implementation of the platform is presented; the concept of Augmented Reality Audio is explained in more detail (Section 2.1), fundamentals of binaural hearing are presented (Section 2.2), different kinds of orientation and position tracking methods are introduced (Section 2.4), and saving audio information in Wav files is explained (Section 2.5).

In Chapter 3 the setup for the platform is chosen, on which the platform is built in Chapter 4. Chapter 5 covers the Audiomemo application, which is tested and analyzed in Chapter 6. Conclusions with suggestions for future work are presented in Chapter 7.

Chapter 2

Theory

In this chapter basic concepts required to build a platform for augmented audio reality applications in outdoor use are presented.

2.1 Augmented audio reality

To understand the concept of augmented audio, real, virtual and pseudo-acoustic audio environments are defined first. The characteristics of binaural hearing are an essential part in generating virtual objects in virtual and augmented reality. Binaural hearing is described in more detail in Section 2.2.

2.1.1 Real vs. virtual audio environment

The difference between real and virtual audio environments is visualized in Fig. 2.1. A real audio environment is the default situation from everyday life when there are no additional equipment to affect the natural perception of audio. In a virtual audio environment all the sounds are originating from another environment or created artificially. Typically the virtual environment is auralized using binaural processing (explained in more detail in Section 2.2.3). [2, 1]

2.1.2 Pseudo-acoustic environment

In a pseudo-acoustic environment (see Fig. 2.2) the user is wearing a binaural headset with a small microphone element integrated into each earphone (illustrated in Fig. 2.9). All the sound recorded with microphones is immediately played back to the earphones. In many applications it is desired to have the pseudo-acoustic environment sound as identical to the real environment as possible [1]. So basically the user hears the real environment,

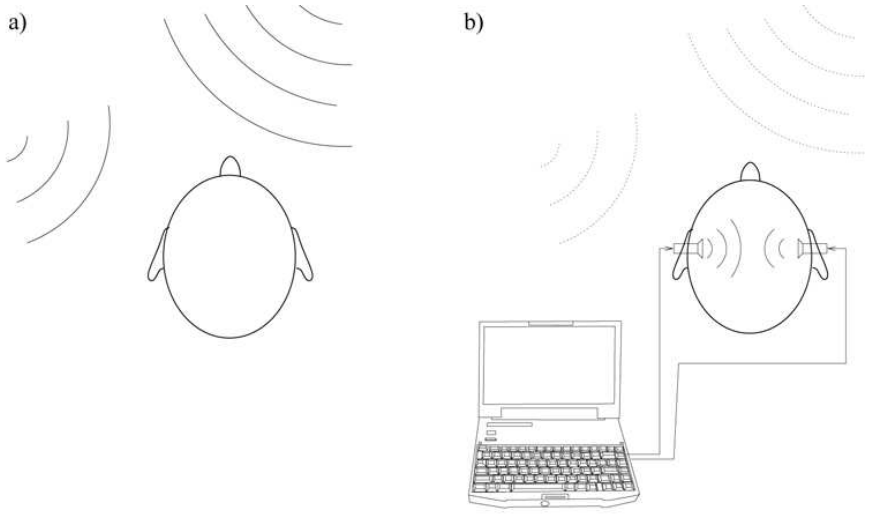

Figure 2.1: In a real audio environment audio is perceived naturally. A virtual environment consists only of recorded or artificial audio. [2]

but through the earphones and a bit colored. The reasons for the coloration, and ways to compensate it, are described in more detail in Section 2.3.2.

2.1.3 Augmented reality audio environment

An Augmented Reality Audio (ARA) environment can be defined as a real environment extended with virtual objects [8]. It is a combination of real and virtual sound scenes mixed so that the virtual elements are perceived as a part of the natural environment. In most cases the augmented audio environment is implemented by first extending the real audio environment to pseudo-acoustic environment and thereafter adding virtual components to create a real environment augmented with virtual contents. This is illustrated in Fig. 2.2. [2]

2.1.4 ARA applications

Various possible use cases and applications for augmented audio reality have been proposed. Lokki et al. [3] divide the applications by the way the virtual audio events are connected to the real environment into *freely floating* and *localized*.

Freely floating audio is not connected to any location in real environment, but the anchor point is the head of the user. An example of this kind of applications is auditory display for mobile calendar proposed by Walker et al. [9]. Lokki et al. propose that applications with freely floating audio events could be used for other information services also (e.g. news or

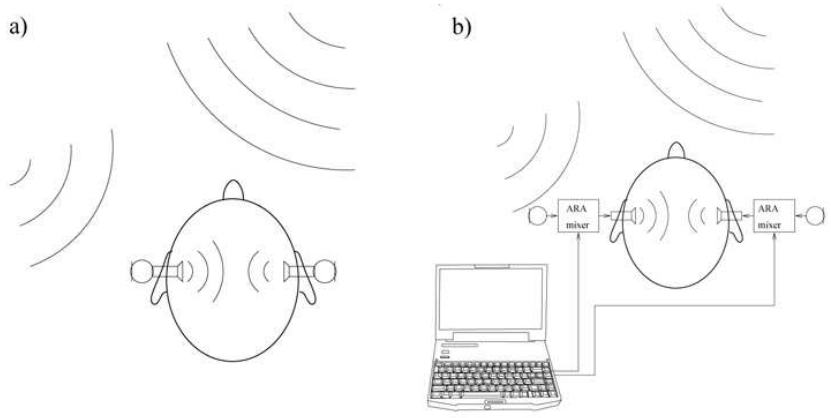

Figure 2.2: A pseudo-acoustic environment is a real-time replica of real environment (see Fig. 2.1 a). An augmented reality audio environment consists of real sounds heard through a pseudo-acoustic environment and completely virtual sound elements. [2]

announcements) and for entertainment purposes (e.g. listening to music) [3].

The purpose of *localized audio* events is to connect audio to objects or locations in a real environment. The Auditory Sticker application proposed by Lokki et al. [3] has been implemented by Seppänen [7]. When using the Auditory Sticker the user can record messages and leave them to be listened to by other users. The messages can be left to a certain location or attached to objects. It is kind of an auditory representation replacement for the yellow Post-it stickers. [3]

In some applications the virtual sounds are intentionally distinguishable from the real environment (for example audio display for mobile calendar proposed by Walker et al. [9]). But usually the augmented reality system should be able to accurately align virtual objects over the real perceived environment [10]. For that the global position and orientation of the user have to be known. Position and orientation tracking are described in more detail in Section 2.4.

2.2 Binaural hearing

Perceiving the environment is a learnt ability. The auditory system analyses acoustic environment from the complex mixture of direct sound, reflections and reverberation arriving from different directions with different delays. Information about the surrounding audio environment and directions of the sound events are processed from two signals - signals to the left and right ear. This is called binaural hearing. [11]

Localization of sound is mainly based on the following six cues:

1. Interaural Time Difference (ITD)
2. Interaural Level Difference (ILD)
3. Monaural spectral cues
4. Effect of head rotation
5. Interaural Coherence (IC)
6. Visual cues

Interaural time difference (see Section 2.2.1) and interaural level difference (see Section 2.2.2) are the most significant cues for determining in which cone of confusion the sound source is. Cone of confusion (Fig. 2.3) is defined as a cone which forms a constant angle with the line connecting both ear canal entrances of the listener. Within the cone of confusion the source generates the same ITD. Monaural spectral cues help determining the location within the cone of confusion. Rotating the head changes ITD, ILD and monaural spectrum. The change can be used as a cue of the source location. [12]

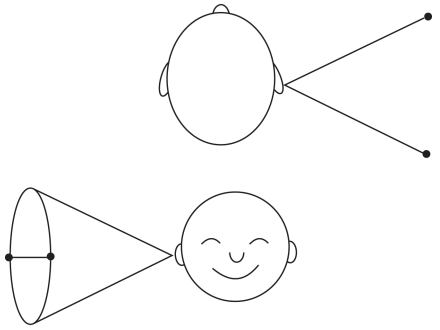

Figure 2.3: Cone of Confusion; the positions where the source creates the same ITD and ILD.

The coherence between the signals to the left and right ear is denoted as interaural coherence. In reverberant environments and with multiple sources IC is an important cue. It has been shown by Faller & Merimaa [13] that in time instances with IC over a certain treshold, ITD and ILD are likely to represent the direction of one of the sources.

All the cues for localization are individual depending on the shape and size of the head, pinnae and torso of the listener. The cues can be measured, or extracted from the transfer-functions called Head-Related Transfer Functions (HRTFs, see section 2.2.3). [12]

To specify the location of a sound source relative to the listener, a proper coordinate system should be used. An intuitive way to go is to use the middle point of the line connecting the two ears as an origo, i.e. approximately in the middle of the head of the listener. The

median plane is defined to be the vertical plane that divides the head symmetrically into the left and right halfs. The frontal plane is defined as a vertical plane – normal to the median plane – which divides the head into front and back parts through the ears. The horizontal plane is the plane at the ear level that is perpendicular to both the median and frontal planes. The direction of the sound source can be unambiguously described with azimuth (φ, horizontal angle) and elevation (δ, angle). Also the distance (r) has to be known to specify the source location. Azimuth (φ) can be also referred as yaw, and elevation (δ) as pitch, see Fig. 2.11. [11]

2.2.1 ITD - Interaural Time Difference

Sound signals propagate with a constant speed from the source towards both ears. Since the ears are on different sides of the head, the signal has longer distance to travel to reach the ear on the other side of the head (illustrated in Fig. 2.4). This time difference in arrival of the sound signal is first of the two most important cues to localize the sound source [11]. ITD is the main cue in low frequencies, with cut-off frequency of 1,3-2,0 kHz depending on the source [14, 15].

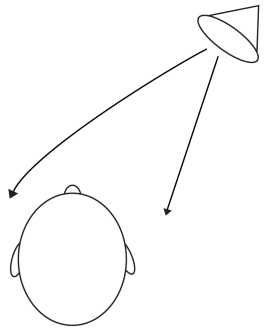

Figure 2.4: Interaural Time Difference – ITD.

2.2.2 ILD - Interaural Level Difference

The second basic cue for spatial hearing is interaural level difference – also known as interaural intensity difference (IID). The head acoustically "shadows" the ear located on the further side from the sound source, which results in different signal levels in each ear (illustrated in Fig. 2.5). The effect takes place in whole audio frequency range, but is the dominant one at high frequencies where ITD loses its importance (f > 1.3-2.0kHz, depending on the source [14, 15]). [11, 16]

Figure 2.5: Interaural Level Difference – ILD

2.2.3 HRTF - Head Related Transfer Function

If bone conduction is neglected, the binaural signal includes all the audio information the listener can perceive. The head, upper torso and pinnae reflect, colorize and shadow the signal differently depending on the direction of the source related to the listener. These effects can be measured, analyzed and simulated with Head-Related Transfer Functions. The HRTFs effectively consist of ITD and ILD, which can also be extracted from the measured HRTFs. Since the human body is individual with different sizes and shapes of the head, upper torso and pinnae, HRTFs are individual. [14, 16]

HRTFs can be measured in free field (e.g. in anechoic chamber) with good loudspeakers and miniatyre microphones placed in listeners ears. The measuring point can be at the entrance of the open ear canal, at the entrance of the closed ear canal, inside the ear canal or at the ear drum. Hammershøi & Møller have shown that the ear canal itself – including some millimeters outside of the ear canal – does not influence the directional information of the signal [17]. Hammershøi & Møller state that since the ear canal is highly individual, more general results are achived by avoiding the ear canal influence by performing the measurements with closed ear canal. Regardless of the microphone placement in the measurements the influences of measuring equipment have to be cancelled. Since the measurements with real listeners are laborious and the results are anyway somewhat individual, it is often beneficial to use dummy heads for measurements. [11, 16]

2.2.4 Resolution in source localization

When using HRTFs it is also important to know the limits in capabilities of the auditory system to notice differences in source direction. In psychophysics the term *just noticable difference* (JND) is used to define the smallest change needed in the localization cues – or in the location of the source – that results in a change of perceived source location [14]. Blauert uses the term *localization blur* of JND in a context of auditory source localization.

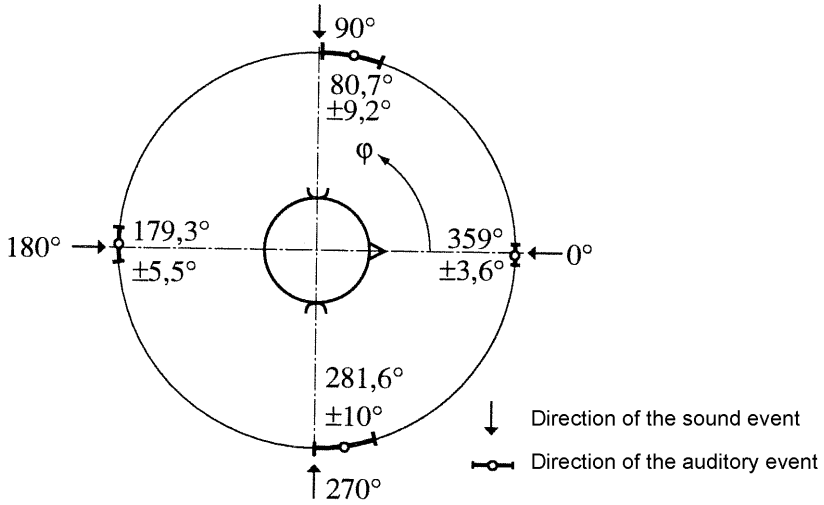

Figure 2.6: Horizontal resolution of human auditory system with stimulus of white noise with duration of 100ms [11, 18].

According to Blauert the absolute lower limit to localization blur is $1\,^{\circ}$, but the actual value varies greatly depending on stimulus and direction. [18]

The auditory system is most accurate on horizontal plane; with the sound source directly in front of the listener the accuracy is almost at the lower limit of the localization blur, $1\,^{\circ}$, whereas on the sides the localization blur is 3-10 times higher. On vertical plane the resolution is substantially lower. At best the elevation localization is in front of the listener – about $4\,^{\circ}$ – and significantly worse with sound source residing on top of the listener. [11, 18].

The given numbers for angles are the best-case scenarios with only one sound source, stationary listener and wideband stimulus. In real environments there are usually several sound sources and room reflections present, which may result in localization cues that do not even correspond to any of the actual sound sources. Human auditory system is however remarkably able to distinguish sources from complex composites of sound. In headphone listening ambiguous – conflicting or nonrealistic – localization cues are usually localized inside the heads of the listeners. This is called *lateralization* [14].

Precedence effect is an important factor in sound localization. Human auditory system tends to localize the sound source to the direction of the first arriving sound, unless the reflection arriving later is much louder. The sounds arriving within the Haas window (30-40 ms) after the first sound, are perceived as one sound, whereas the sounds arriving after that are perceived as an echo or separate sounds. [18]

Visual resolution is two orders of magnitude higher than the auditory resolution [18].

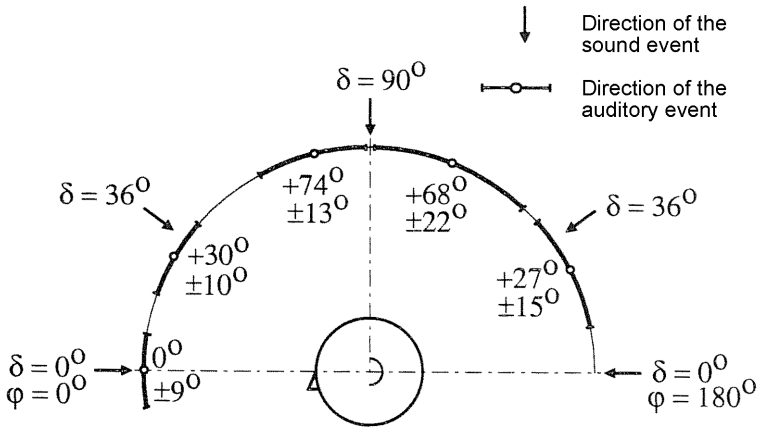

Figure 2.7: Localization and localization blur in the median plane with different elevation angles δ using the stimulus of continuous speech by familiar person [11, 18].

Often the visual cues are dominant over the auditory ones; this means that when the auditory cues point to the source location within a certain range, the visual cues can fine-tune and fix the perceived direction.

2.3 Spatial sound reproduction

To properly add virtual elements on top of a real audio environment, a device to produce the sound to the ears of the user is needed. Loudspeakers could be placed on the shoulders of the user, or an earphone-based headset could be used. The earphone-based system also needs a mixing device to handle the signal equalization.

2.3.1 Loudspeakers on shoulders

Loudspeakers could be placed on the shoulders of the user, which enables the user to hear the natural environment freely and unaltered. However, the shoulder placement of the speakers reduces the quality of the sound and localization cues. It may also rise privacy issues, and cause disturbance since all the audio is audible to other people around the user. [20]

Figure 2.8 presents an example of a shoulder-mounted device, Nortel Soundbeam Neckset, which has been used in Sawhneys Nomadic Radio [20, 19]. The speakers are placed on each shoulder and microphone on top of the users chest.

2.3.2 In-ear headset

Another option is to use miniature microphones integrated into each earplug and play the recorded sound without any delay through the earphones. In many applications the intention is to make the pseudoacoustic environment sound as much like the real environment as possible.

The problem with in-ear earphones is that when placing the earplugs into the ear canal, the entrance of the ear canal is blocked. When in normal situation (i.e. with open – unblocked – ear canal) the ear canal acts as a quarter-wavelength resonator, it now acts as a half-wave resonator, effectively doubling the main resonance frequency. This results in a loss of normal quarter-wave resonance and emergence of the new half-wave resonance. These both have to be taken care of with two separate filters; one to artificially boost the missing resonance frequency and one to attenuate the resonance caused by the blocking of the ear canal. [6]

With earphones one of the most important factors influencing on quality in sound reproduction is leakage through and around the earphone. With loose fitting of the heador earphones the pressure chamber effect, which denotes the cavity with pressure in phase with the volume displacement of the transducer membrane and an amplitude proportional to it [21], does not occur. This is due to improper sealing of the system. So the fitting must be as tight as possible, but some leakage will always occur. [6]

2.3.3 ARA Headset + Mixer

With a headset consisting of earphones and microphones integrated to them, a mixing device is required to reduce the effects mentioned in the previous section. Fig. 2.2 b illustrates

Figure 2.8: An example of a shoulder mounted device; Nortel SoundBeam Neckset [19].

Figure 2.9: ARA Headset inserted into the ear canal entrance [6].

the use of the ARA mixer.

Since no commercial ARA headsets are available at the moment, V. Riikonen has implemented an in-ear ARA headset within his masters thesis [6], see Fig. 2.9. The presented system consists of Philips SHN2500 earphones (Fig. 2.10) with integrated microphones, and a custom built mixer.

Cancelling the leakage of the earphone with a compensation filter would typically call for digital implementation, but since the leaking sound is anyway passing in real time, the filtering has to be done without any delays. Otherwise the leakage would arrive to the ear of the users before the compensated pseudoacoustic signal. At low frequencies the latency is somewhat acceptable, so digital filters could be used to some extent. However, Riikonen has taken an approach to use only analog circuits to build the filters. This approach provides us efficient filters with no delays. [1, 6]

A first-order highpass filter is used to compensate for the leaking low frequency sound. There are also two parametric equalisation controls. The first is a biquad peaking filter to make sure that the natural quarter-wave resonance occurs even though the ear canal is occluded. The range of the peak control is between 700 Hz and 3200 Hz. The other one is a notch filter designed to compensate for the peak posed by the occlusion of the ear canal. The notch can be within the range of 1.8-8.5 kHz. [6]

2.4 Tracking

In order to create a truly immersive virtual – or augmented – environment, the movement of the user should have an influence on the audio environment the user is hearing. The

Figure 2.10: Philips SHN2500 earphones with integrated microphones and a control unit [22].

virtual objects should sound like they have fixed positions; i.e. if the user turns his head, the panning of the virtual sound should be compensated to make the user feel like the virtual object is stationary. This requires tracking of both position and orientation of the user. [23]

Various position and orientation tracking methods have been presented. According to Sherman [23] there are three factors playing against each other in position tracking systems:

- Accuracy and speed
- Interfering media (e.g. metals, opaque objects)
- Encumbrance (wires, mechanical linkage)

Currently there is no technology available, which provides optimal conditions in all three areas, regardless of the price – which is often an important factor too. However, a reasonable systems can be implemented by taking into account the limits and making optimal tradeoffs. [23]

The terms ’tracking’ and ’positioning’ can both be used in this context. To note the difference, more specific terms ’position tracking’ and ’orientation tracking’ are used, along with the term ’positioning’.

2.4.1 Degrees of freedom

A *Degree of Freedom* (DOF) is a particular way an object can move in space. The different ways of movements can be divided into *translational* and *rotational movement*, where translational movement represents sliding the object along straight line and rotational movement means rotating the object in question about its axes. The translational and rotational movement are visualized in Fig. 2.11. [23]

Object’s movement without any mechanical linkages can be stated in terms of six degrees of freedom. Translational degrees are usually expressed simply as location along the x, y

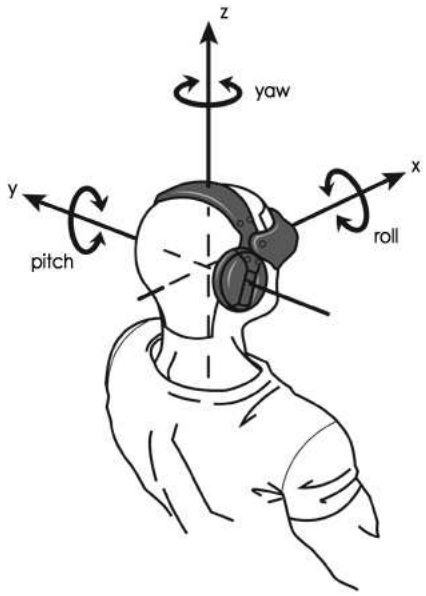

Figure 2.11: Translational (x,y,z), and rotational (yaw, pitch, roll) movement. These six degrees can define object's movement – location and orientation – freely in space. [24]

and z axes, i.e. position. The x, y and z axis are defined to be orthogonal to each other and all movements are considered from the user's point of view. Orientation, or rotational degrees, are expressed as *roll*, *pitch* and *yaw*, which represent the rotation about the x, y and z axes, respectively.

Some of the tracking systems report the complete 6-DOF position, but some are capable to report only a set of degrees. These subsets are for example 3-DOF orientation (for only rotational movement) and 3-DOF location (for only translational movement). [23]

2.4.2 Mechanical tracking

Mechanical tracking is probably conceptually the simplest method to track the movement of the user. The arm of the tracking device is physically attached to the head of the user. The arm consists of two or more rigid mechanical pieces interconnected with electromagnetical transducers (like potentiometers). When the user moves, the transducers located in the joints of the arm measure the movement and with the prior knowledge of the physical properties of the arm the exact location and orientation can be calculated. Both rotational and linear movements can be measured and calculated quickly, accurately and precisely. [25, 23]

On the pros side the mechanical arm can help carry the weight of the tracking device, and the display if one exists. The mechanical arm also enables force-feedback to the user, since the same electromagnetic devices that are used to sense the position can also be used

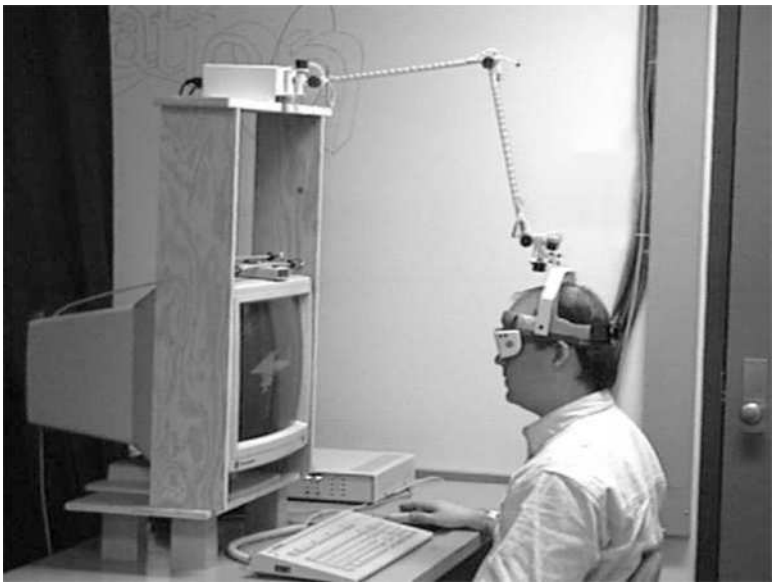

Figure 2.12: Mechanical tracking presented in [27]. The head movement of the user is tracked with the mechanical arm. 3D glasses are used to provide visual stereo image accordingly.

to provide forces. An example of such a device is Phantom by SensAble Technologies [25, 26].

The tradeoff with the accurate and precise estimates on location and orientation is the very limited range of usage. The device is fixed on the location and typically the area of motion is only around one cubic meter [25]. An example of mechanical tracking device can be seen in Fig 2.12, which is a system called fishtank. The idea is to provide the user with a small window to virtual environment by displaying 3D content in a regular workstation display and make it more realistic by following the user's head position and orientation and draw the virtual image on the display accordingly [27].

2.4.3 Acoustic tracking

Acoustic tracking is based on sending and receiving a sound signal. The first implementations of acoustic tracking were based on continuous signal and measuring the phase shift between the transmitted and the received signal. Measuring the phase enables only detecting the relative distance changes within the cycle. There are problems also with continuous signals; because walls and objects reflect the acoustic signal within the room, summing the direct and reflected signal, the amplitude and the phase vary drastically depending on the receiver's position. Probably for this reason there have been no succesful implementations

with continuous signal. [25]

All commercial acoustic tracking systems are based on timing of the flight duration of a brief ultrasonic pulse. With short pulses the multipath problem can be circumvented since the first arriving pulse can be guaranteed to be the direct signal – unless the direct path is blocked [25]. Three transmitters and three receivers provide enough data to triangulate the full 6-DOF position of the user [23]. If the transmitters are positioned on the user, and receivers are located in the space, the system is called outside-in tracking. The downside with outside-in tracking is that the information is processed at the receiver, but it is often needed at the user end. [5]

Another approach is to use binaural recording – i.e. microphones integrated in earplugs – and use the same cues humans normally do (specifically ITD, but also ILD. See Section 2.2). Surrounding reference sound sources, *anchors*, are used for positioning. Since the positioning is done binaurally – and therefore the result is the position of the ears, not some other part of the body – the resulting position information can be directly utilized in augmented reality applications. [5]

Multiple anchors are needed for practical positioning. Wider areas can be covered and 3D-positioning is possible. The anchor signals can be adjusted to avoid overlap of each other. The anchor signals can be transmitted imperceivably and efficiently by using a high-frequency carrier signal to modulate the low-pass reference signal. [5]

The disadvantage in using acoustical tracking is the requirement of line of sight; all the

Figure 2.13: Logitech 3D Head Tracker. The triangle-shaped transmitter with three ultrasonic speakers at the corners is positioned on top of the monitor. Three microphones to receive the signals are located in this case in the 3D-glasses worn by the user. [28]

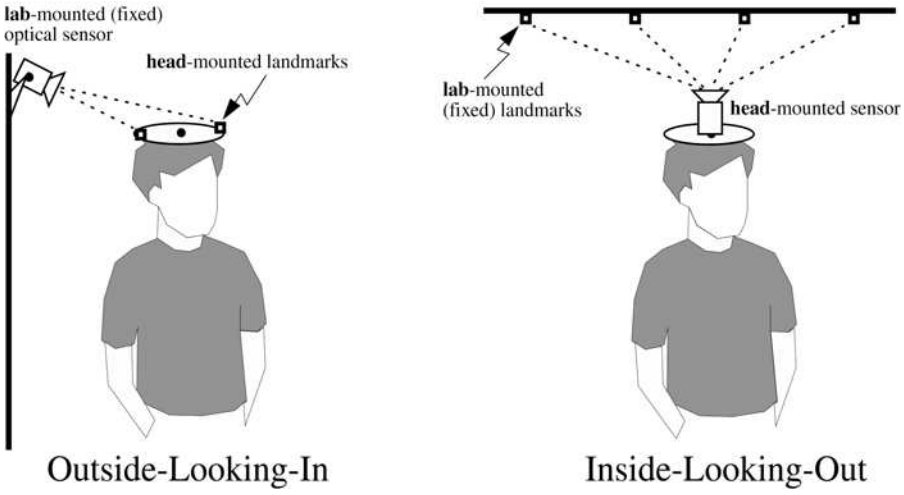

Figure 2.14: Visual tracking can be done Outside-Looking-In point of view (i.e. the moving user is tracked with fixed camera, Fig a.) or Inside-Looking-Out (i.e. from users point of view, Fig b.) [29]

objects in the line between the transmitter and the receiver distract or block the sound affecting it's measured travel time. The resulting resolution of the tracking system depends on the frequency used; the higher the frequency used, the shorter wavelengths, and therefore higher resolution. But the higher the frequency is, the more frequency-dependent attenuation of the sound in the air will happen. [25]

Other problems include unpredictability of the speed of the sound, which is dependent on the temperature, humidity and currents of the air [25], and the need for a fixed set of sources, loudspeakers. There are locations, which may already contain loudspeakers in known, fixed positions, like stores, vehicles and museums. These could be used as anchors, but the requirement of fixed sources limits the range of usage to fixed locations [5]. An example of a smaller scale ultrasonic tracking system can be seen in Fig. 2.13.

2.4.4 Optical / visual tracking

Visual tracking can be performed with two approaches; Outside-Looking-In or Inside-Looking-Out, visualized in Fig. 2.14. The difference is in the positioning of the landmarks to be tracked and sensors to perform the tracking. When using Outside-Looking-In tracking, the sensors have fixed position and the moving user is tracked. With Inside-Looking-Out the camera – or other optical sensor – is located on head of the user, and the environment is tracked from the user's point of view. [29]

Seppänen used a video camera attached to the ceiling to track the movements of the

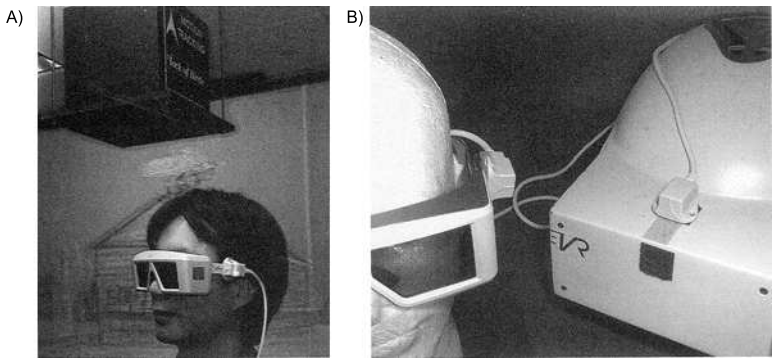

Figure 2.15: Electromagnetic transmitter has to have a fixed location (Fig. A), but the receiver is rather small and can easily be connected to 3D-glasses or on top of helmet. [23]

user at floor level [7]. Retroreflective material was used to mark the user; the users were wearing caps covered with retroreflective material of different colours. Control commands could be sent to the system by showing cards of certain colour. With a camera attached to the height of 3 meters, the tracked area was about 20 m^2, and the tracking error was 4-40 mm. Elevating the camera did widen the area to be tracked, but with the cost of degraded resolution and increased error. [7]

Another approach is to place a video camera on the head of the user and to analyze recorded images of surrounding areas to locate landmarks and recognizable shapes, like corners of the room. The position of the camera, and therefore also the position of the user, can be calculated from this information. Computationally this is quite demanding, but it can be eased by placing distinct landmarks at known locations. The landmarks can be made distinct with colour and shape so that the computer vision algorithm can easily distinguish them from surrounding objects. [23]

With visual tracking the downside is the requirement for a line of sight. Blocking objects prevent tracking to happen. With analog photosensors the situation is even worse; partial occlusion results in plausible but incorrect positioning [29]. The visual systems also have high demands for constant lighting; there has to be enough light, and it has to light the tracked area equally well. [7]

2.4.5 Electromagnetic tracking

One method commonly used in virtual reality applications is electromagnetic tracking, presented in Fig. 2.15. A transmitter is used to generate a low-level magnetic field from three orthogonally positioned coils. The receiver unit is placed on the head of the user and the signal in each coil in the receiver is measured. The strength of the signal in each receiver coil determines full 6-DOF position with respect to the position of the transmitter. Since

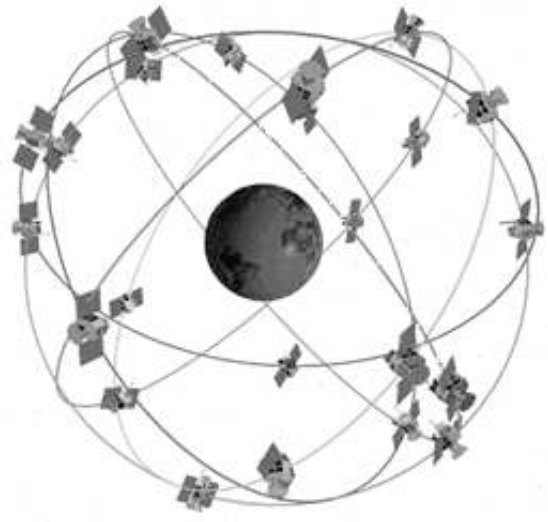

Figure 2.16: The 24 satellites orbiting the earth at the altitude of 11 000 nautical miles provide global position tracking around the globe. [30]

the position of the transmitter is known, the absolute position of the receiver, and thus the user, can be calculated. [23]

The magnetic field of the earth can also be used in magnetic orientation tracking, since it is already naturally existing and widely available. The earth is not a perfect sphere, and the shape of the magnetic field varies depending on the location. The anomalities depend on the area and look-up tables to correct the measurements are available [25]. One has to note that other magnetic fields and metals may interfere the measurement. [23]

The advantage with electromagnetic tracking over many other methods is the lack of line of sight restriction, and electromagnetical tracking devices can be implemented wirelessly. This reduces the restrictions to the area to be covered with the tracking. The disadvantages include the interference caused by metal in the tracking area. Also; with artificial field sources – i.e. not the earths magnetic field – the range of usage is rather small; reasonable accuracy is achieved within only 1-3 m from the transmitters. [23]

2.4.6 Radio wave tracking

Radio wave tracking is based on triangulating the location from at least three signals, which are originated from known locations. The *Global Positioning System (GPS)* uses satellites orbiting the earth as the signal sources. The satellites provide global position tracking, as the acronym GPS suggest, visualised in Fig. 2.16. Each satellite is equipped with an atomic clock to accurately keep the time. This time is sent along the signal and used to calculate the travelling time from satellite to the receiver. With the fact that electromagnetic waves

are travelling with the speed of light, the distance from the satellite can be calculated. If the distance from one satellite is known, the location is somewhere on the surface of the sphere with the radius of the distance to the satellite. With known distances to two satellites, the location is somewhere in the line of intersection of two spheres with separate radii of distance to each satellite. Finally with three known distances, exact location in the space can be calculated. [30]

Since radio wave methods are basically based on electromagnetic waves, the methods can be used in much wider ranges than tracking systems based on quasi-static magnetic fields (introduced in Section 2.4.5). The radiated energy attenuates in a field of radius r as $1/r^2$, compared to the dipole field with the strength gradient dropping as $1/r^4$. The absorption losses of the radio waves in the air, and the effects of wind and air temperature, are minimal and can be neglected. [25]

With a radio wave tracking system only the location can be calculated directly. The GPS devices do provide direction and speed, but that information is calculated from sequential data points. The proper orientation tracking has to be handled with some other methods. The GPS uses *World Geodetic System 1984 (WGS84)* as a system of coordinates [30], which has to be remembered if a different coordinate system for mapping is used.

Outdoors and in large open spaces the system works well. But there are limitations to the use of GPS tracking. Radio waves are attenuated rapidly in water, which means that human body blocks the signal. Since the method is based on measuring the travel time, the reflections from walls indoors and between high buildings cause problems due to multipath distortions. [25]

2.4.7 Inertial tracking

Inertial trackers operate the same way the vestibular organs in the inner ear perceives the head orientation; the fluid tries to stay stationary, while the structure around it moves. The difference and the changes between the fluid and the moving structure can be used to calculate the orientation and changes in orientation [25]. Gyroscopes and accelerometers can be used to complement each other's tracking results. The accelerometers and gyroscopes are both present in groups of three, orthogonally positioned, one for each axis, as visualized in Fig. 2.17 A.

Inertial tracking provides tracking without the requirement for line of sight and there is no need for complementary components fixed to known locations, and it is immune to all kinds of interferences. This means that the range of usage is unlimited. Inertial tracking devices work relatively quickly, and can be connected to the system wirelessly if needed. [32, 25, 31]

The problem with inertial tracking is the fact that it is not absolute, but measures relative

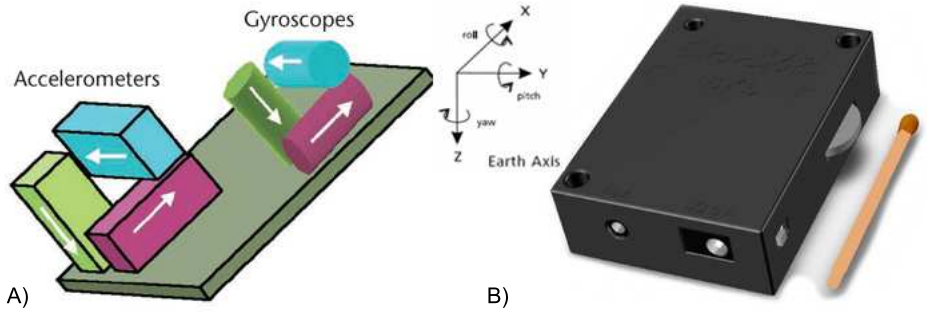

Figure 2.17: Schematic drawing of inertial tracker with tri-axis gyroscope and accelerometer (Fig. A, [25]). An image of actual inertial tracking device, SHAKE (Fig. B, [31])

change in orientation and position. Even small error per sample accumulates quite fast to quite big errors. Good trackers, which still have reasonable size for head tracking, drift about $15°/\mathrm{hour}$ without recalibration. The drifting can be compensated for example by filtering the data with Kalman filters. [32]

Inertial trackers can be used in combination with some other tracking method to recalibrate the orientation every now and then. SHAKE (Sensing Hardware Accessory for Kinesthetic Expression) inertial sensor pack is an example of devices which uses tri-axis accelerometers to measure quick changes in orientation, and tri-axis magnetometers to measure the absolute orientation and to recalibrate the possibly drifted value. (SHAKE-device is visualized in Fig. 2.17 B). A triple-axis gyroscopic angular rate module is optional. [33, 31]

2.5 Audio files

Sound from microphones is filtered, sampled and quantized to *Pulse Code Modulated* (PCM) signal (for more details e.g. [15]). The signal can be saved to a file in various formats – often lossless or lossy methods to compress audio are used (e.g. FLAC [34] or MP3, respectively) – but only saving the signal as raw PCM to WAV file is presented.

2.5.1 Wav files

One of the most common ways to save the sound signal is to use Microsoft WAVE audio file format. The base of the format is specified in the Microsoft *Resource Interchange File Format* (RIFF). WAVE audio is a specific subtype of RIFF to contain audio data. [35]

A RIFF file consists of *chunks* that contain fields for information about the file or the chunk itself. Each chunk has to have a 4 byte long unique identifier (e.g. 'fmt ' – space as a

last character), and chunk data size (in 4 bytes). Since one byte is 8 bits, the maximum size of a chunk is $(2^8)^4 = 4,294,967,296 = 4$ GB. Since the background of the format lies with Microsoft – and therefore also with Intel – all data values are stored in little-endian order (least significant byte first). There are no other limitations to the content. [36]

There are few required chunks, and unnumbered amount of other chunks. If the chunk identifier is unknown to the program opening the file, the RIFF standard tells to skip and ignore the chunk. Each file has one RIFF chunk, which includes all the other chunks as a subchunk. In WAVE audio files 'RIFF' (container for the contents of the file), 'fmt ' (for format information) and 'data' (for actual audio data) chunks are required. [35, 36]

Table 2.1: The structure of fmt -chunk (after [36]).

Offset	Size [bytes]	Description	Value
0x00	4	Chunk ID	"fmt " (0x666D7420)
0x04	4	Chunk Data Size	16 + extra format bytes
0x08	2	Compression code	e.g. 1 = uncompressed PCM
0x0a	2	Number of channels	e.g. 2 = stereo
0x0c	4	Sample rate	e.g. 44100 (Hz)
0x10	4	Average bytes per second	$SampleRate * BlockAlign$
0x14	2	Block align	$SignificantBitsPerSample/8 * NumChannels$
0x16	2	Significant bits per sample	e.g. 16 (bit)
0x18	2	Extra format bytes	0 - 65,535

Wave Format chunk

Format chunk (ChunkID 'fmt ') is a required chunk since it holds all the basic information to play back the actual contents of the file; the audio data. Possible fields are listed in Table 2.1. If the compression method is any other than uncompressed PCM (i.e. Compression code is not 1), other chunks to define the compression details may be needed. Number of channels, sample rate and bit rate are required information to correctly play the contents of the file. 'Block align' and 'average bytes per second' describe the information for reading the data in playback buffer; the previous one tells the space requirement of one sample with all the channels, and the latter one is used to estimate the buffer size. [35, 36]

Wave Data chunk

Data chunk is the container for the actual audio data (see Table 2.2). All the information to interpret the audio data is given in the format chunk. The channels are interlaced so that all the samples related to one time instance are located next to each other. This way the file can

Table 2.2: The structure of data -chunk (after [36]).

Offset	Size [bytes]	Description	Value
0x00	4	Chunk ID	"data" (0x64617461)
0x04	4	Chunk Data Size	depends on sample length and compression
0x08	Audio data		

Table 2.3: The alignment of interlaced stereo wave samples (after [36]).

Sample	Channel	Value
0	1 (left)	0x0053
	2 (right)	0x0024
1	1 (left)	0x0057
	2 (right)	0x0029
2	1 (left)	0x0063
	2 (right)	0x003C

be streamed and the playback can be started without first loading the whole file in memory at once (see Table 2.3. The 'Value' column from top to bottom represents the actual data appearing in file). [36]

Chapter 3

Platform Setup

This chapter begins with the requirements for the platform developed in this work. The rest of the chapter deals with the chosen hardware and software. A general setup for mobile ARA applications is presented.

3.1 Requirements for the platform

Several mobile ARA applications have been implemented in the past [1, 3], but most of them have limitations which force the applications to be used only indoors. The goal of this thesis is to define a general platform for MARA applications for outdoor use. Existing devices – and software when possible – has been used.

Requirements for the platform are comprised of the following:

- The system should be able to record and play back binaural audio in real time.
- The system should be location independent, i.e. work in real environments.
- Recorded sound should be indexed with position and orientation information.
- Position and orientation tracking should be global.
- There should be some reasonable way for the user to communicate with the system.
- The platform should be OS independent (Windows, Mac & Linux supported).
- The platform should be extendable.
- The equipment should be light enough to carry with ease.

3.2 Hardware

3.2.1 Binaural recording and playback

For portability and location independence reasons headphones are selected over loudspeaker setups. Since both binaural playback and recording are required, the most suitable headphone selection is to use in-ear headphones with integrated microphones. This kind of combination is used in noise cancelling headphones but for ARA use there are no commercial headsets available in the market. The ARA mixer and headset introduced in [6] are used.

The more naturally the user hears real environment through headphones, the less he pays attention to devices he is wearing. The headphones in ear canals cause coloration in the sound. To compensate the coloration, Riikonen's ARA mixer equalizes the reproduced sound with analog filters. Compared to the natural listening experience, the difference with ARA headset and mixer is perceptible, but not annoying. [6]

3.2.2 Position tracking

Various methods for position tracking have been introduced. User-based methods – mechanical, acoustic, optical, electromagnetic, radio wave and inertial tracking – have been described in Section 2.4. All the methods mentioned do work indoors, but require a fixed location. For indoor use GPS would be unusable because it does not work inside buildings due to signal loss [25]. But since the platform is intended for outdoor use, there is no hindrance in using GPS.

The device should have documented protocol in order to enable communication with the platform. The National Marine Electronics Association (NMEA) has defined an interface for marine equipment to communicate with each other and computers. Various implementations have been presented for NMEA 0183 interface standard, including for Pure Data.

There are GPS devices with USB and Bluetooth connection available. Since wireless use is not necessary, and to avoid the need for batteries, a USB model was chosen. SiRFstar III GPS chip seems to offer good performance; it can receive 20 channels simultaneously, it has low power consumption and high sensitivity. GPS receivers based on this chipset have performed better than receivers based on other chipsets [37].

Chosen Globalsat BU-353 is a GPS device with SiRFstar III chip and USB cable, and it supports the NMEA 0183 protocol.

3.2.3 Orientation tracking

Orientation tracking is often handled with location tracking methods. Since GPS is used only for location tracking, separate tracking for orientation is needed. Requirement for location independent orientation tracking rules out acoustical, optical and mechanical methods. Combination of magnetic field tracking – for compass heading – and acceleration sensors – to compensate rapid head movements – is required to achieve head orientation tracking. Such devices have been introduced, but most of them are for industrial use and thus rather expensive.

SHAKE (Sensing Hardware Accessory for Kinesthetic Expression) inertial sensor pack is a matchbox-sized device, which features tri-axis magnetometer and tri-axis accelerometer [33]. It is lightweight (31 grams), can be connected to computer via Bluetooth and has rechargable batteries [31]. Because of the small size, light weight and cordless interface, it is almost unnoticeable for the user. The SHAKE device will be used since it is reasonably priced and offers the required features.

3.3 Software

Softwarewise the requirements for the platform are operating system independence and extendability. Since the signals will be both audio and metadata, but with emphasis on audio, a programming environment designed for audio use is prefered. Several patchable audio DSP environments exist.

- The most widely used is text-based open source environment Csound [38, 39].
- Explicitly for real-time use designed SuperCollider ([40]) is open source, but windows version is still on beta stage.
- The commercial alternative Max/MSP and it's open source counter part Pure Data ([41]) are both GUI-based environments designed for real-time use. Pure Data (PD) has simplified data structures in Max [42], and it is open source working on Linux, Windows and Mac OS.

Since real-time use of audio is required, PD is chosen over Csound. The requirement for OS independence rules out SuperCollider. Max/MSP supports only Mac OS and Windows. Pure Data is chosen as software environment since it meets all the requirements set in Chapter 3.1.

3.4 Chosen setup

The chosen setup consists of:

- **ARA headset** for binaural recording and playback
- **Globalsat BU-353 GPS device** for location tracking
- **SHAKE device** for orientation tracking
- **Pure Data** software environment
- **HP Pavilion tx 2000** Table PC

Chapter 4

Platform

In this chapter the components selected in Chapter 3 are combined to form a solid platform for MARA applications. The use of the platform is demonstrated in Chapter 5 with the Audiomemo application.

4.1 Platform in general

The goal of this Thesis is to implement a platform to be used for Mobile Augmented Reality Audio applications outdoors. The intention is to keep the platform general and usable in various situations. The universality of the platform is achieved by keeping the components replaceable; any physical component can be replaced with minor coding effort. The requirements for each component type are presented with introduction of the component.

The general layout of the platform is presented in Fig. 4.1. The heart of the platform is the ARA mixer with microphone-integrated headset. The mixer is connected bidirectionally to the Control Unit with stereo signals. The Control Unit also receives data from a Position Tracking Unit and an Orientation Tracking Unit. The User Interface can be implemented for example with a Nintendo Wii Remote.

Flext C++ development layer was used to implement the additional externals required for the platform in Pure Data. Flext was used because it enables the compiling for PD and Max/MSP on Windows, Linux and OSX without changes to source code; however so far the platform has been tested only with PD on Windows XP. [41]

The decision was made to implement pre- and postparsing of commands sent to the SHAKE device as externals, but there is no reason this could not be done purely on Pure Data also. Writing simple WAV files could have been done with existing Pure Data internals and externals, but since the intention was to save the metadata within the WAV file, SndObj library [43] was used as a basis of the implementation. SndWave class was extended to in-

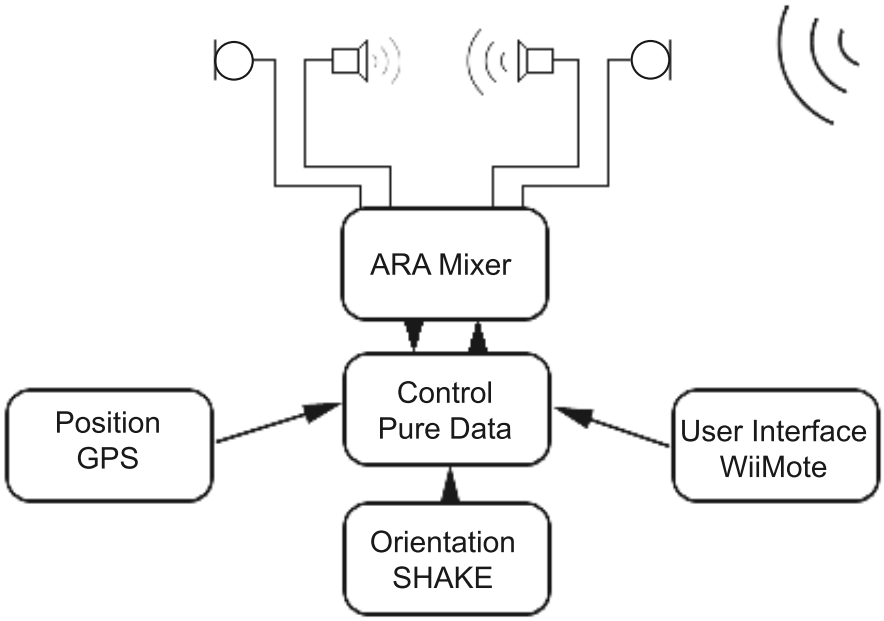

Figure 4.1: Structural layout of the presented MARA platform

clude writing and reading of required ARA chunks, and otherwise the existing functionality of SndObj and Flext was used.

Since the reading and writing of ARA chunks are implemented in SndObj, the extended SndObj can be used without Flext and PD to implement for example a standalone command line metadata extractor for ARA-Wav files.

4.2 ARA headset and mixer

The real audio environment is recorded with microphones, after which the audio signal is directed to both the Control Unit and immediately back to the user's ears via equalization creating the pseudo-acoustical audio environment described in Section 2.1.2. The equalization is done to compensate the effects of headset blocking the ear canal, as explained in Section 2.3.2. If there is a need to augment the pseudo-acoustical environment by adding virtual sounds, the additional audio is received from the Control Unit as an already processed and orientation-panned analog binaural signal. It is simply added to the equalized real time audio signal. The headset and microphones are connected to the mixer, and the mixer is connected to the Control Unit (i.e. computer) with two cords with 3.5 mm stereo-plugs.

4.3 Orientation tracking - SHAKE device

The orientation tracking in this platform is implemented with the SHAKE device (see Section 3.2.3). However the platform is not specifically relying on this particular device; if better devices appear on the market, they can be used as well. The only requirement for the device is the output of an absolute compass heading value. The orientation is expressed as a compass heading (Yaw) presented as degrees from 0.0 to 360.0. Additional orientation information pitch and roll are presented as degrees between $-90\,^\circ$ and $+90\,^\circ$, and $-180\,^\circ$ and $+180\,^\circ$, respectively. All angles default to $0\,^\circ$ (i.e. 'head not turned') if no information is available.

The SHAKE device is attached to the visor of a cap placed on the head of the user and it is connected to the Control Unit via Bluetooth. The device is seen by Pure Data as a regular COM port, and was used with a 'comport' external. Since the 'comport' external accepts lists of ascii characters as integer values as input, and outputs also integer values, pre- and postprocessors for SHAKE commands were implemented as externals (shakepre and shakepost, respectively). PD patches would have been possible too, but writing the external was chosen for simpler implementation.

4.3.1 PD external 'shakepre'

Fig 4.2 illustrates how the preprocessor takes in SHAKE command packets, as well as some predefined shortcuts to full SHAKE command packets – this way shortcuts to most used commands can be used easily while enabling the control of SHAKE device in all possible ways. In PD a string which includes commas is handled as a list. To avoid this behavior, all the commas in SHAKE command packet messages should be replaced with dots. The dots are replaced back to commas in preprocessor before sending the command packet to the SHAKE device.

4.3.2 PD external 'shakepost'

Fig 4.3 shows that the 'shakepost' external accepts input directly from a COM port. Arrays of integer values representing ascii characters are transformed into SHAKE response packet strings. The heading packets (HED) are directed to the first outlet, accelerometer packets (ACC) to the second outlet, angular rate packets (ARS) to the third outlet, and all the other response packets are directed to the fourth outlet.

For the first three outlets the output is a list consisting of a symbol for the packet name (i.e. HED for heading, ACC for acceleration and ARS for angular rate movement [31]) and corresponding data elements as float values (for example X = 0.031, Y = 0.581 and Z = 0.806 for accelerometer (ACC) values, as illustrated in Fig 4.3).

The fourth outlet outputs a list consisting of symbols corresponding to the SHAKE packet received, registry address the packet refers to and the value in given address in hexadecimals and in decimals (which can be expressed in bitwise representation as in Fig 4.3). For example a list of symbols `ACK, 0x0000, 0xFF, 256` tells that the register, which defines the states of the measuring instruments, has a value `0xFF` to define that all the instruments are switched on. As can be seen in Fig. 4.3 the register value can be transformed to bitwise presentation to be easier interpreted. Details of the command and response packets to be used with the SHAKE device are explained in the SHAKE manual [31].

4.4 Position tracking - GPS

Globalsat BU-353 was chosen as the GPS device for the platform (see Section 3.2.2). The platform is not depending on this particular device. In fact any other position tracking method could be used as long as the position data can be presented in the form of the NMEA Recommended Minimum sentence C (GPRMC) [44]. This sentence expresses the position in degrees of latitude and longitude. The sentence also includes the speed, bearing, satellite-derived time, fix status and magnetic variation, which can be calculated with GPS also. GPS-calculated bearing values could be used to verify the validity of the orientation data. The fix status is used in deciding whether the position data is to be trusted or ignored.

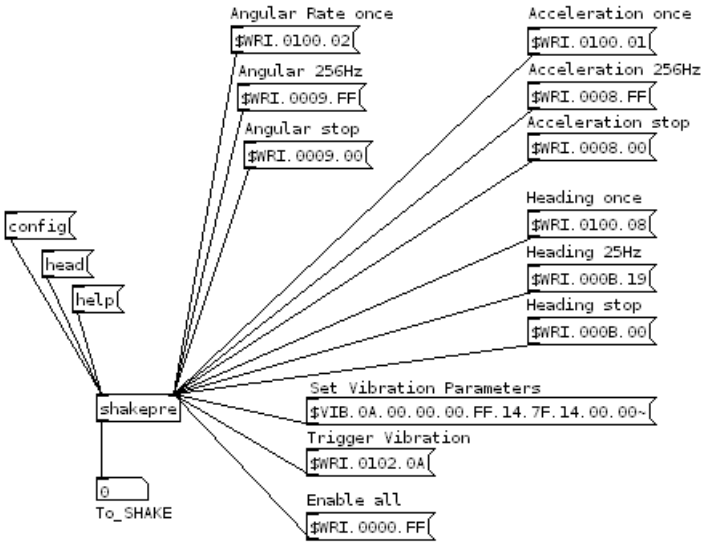

Figure 4.2: 'shakepre' PD External. Shortcuts connected to the left inlet, and full SHAKE command packets connected to the right inlet. Notice that all commas in command messages are replaced with dots.

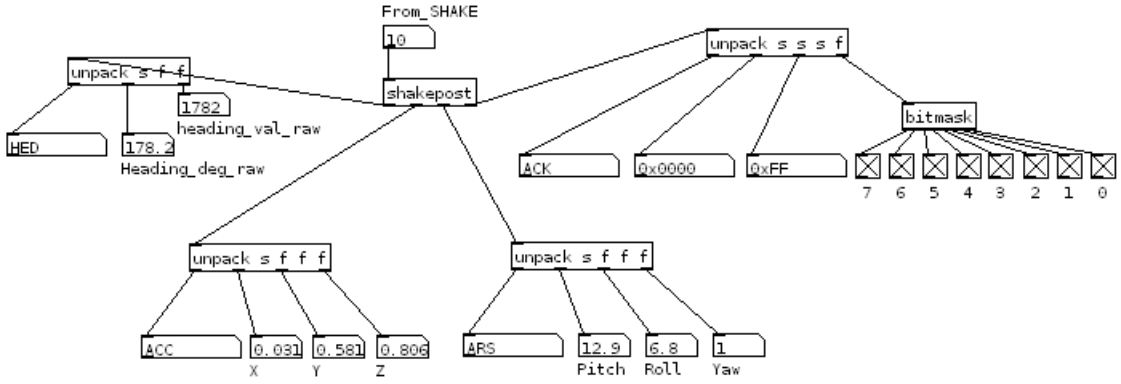

Figure 4.3: 'shakepost' PD External.

The GPS device is attached to the computer with a USB cable. Since the signals from the satellites penetrate cloth, the device is placed in the backpack with the computer. Pure Data can see the GPS device as a regular COM port, and the device is mapped to PD with 'comport' external. There is no data going towards the satellites – the GPS device only listens to signals – so the only data is a list of integers representing ascii characters from the GPS device. This list has to be parsed to extract the required information.

`NMEA gps parser` Pure Data patch [41] was used as a starting point of implementing the GPS parser. An outline of the patch can be seen in Fig. 4.4. The array of integer values is first converted to characters and repacked to a full sentence. The `GPRMC` can be used as a full string or the sentence can be parsed to get the specific data included. To combine the list to form a string `list2symbol` with empty symbol as a separator is used. (Please note that all commas are presented here as hyphen characters, since commas are interpreted as a list separator in Pure Data). The `fix status` value (true or false) within the sentence is used to determine if the values should be used or completely ignored.

The result of the GPS data parsing is a full `GPRMC` sentence string, such as:
$\$GPRMC, 184611.000, A, 6011.2381, N, 02449.2088, E, 1.94, 204.12, 181108, , *07$
Giving the position in degrees, minutes and seconds of both latitude and lognitude
$N60\,°11'23.81", E24\,°49'20.88"$

The GPS device requires at least three signals from satellites to fix the position for a certainty (see Section 2.4.6). If there are too few signals available, the GPS device calculates the position according to the data available, but also gives the information that the result is probably inaccurate.

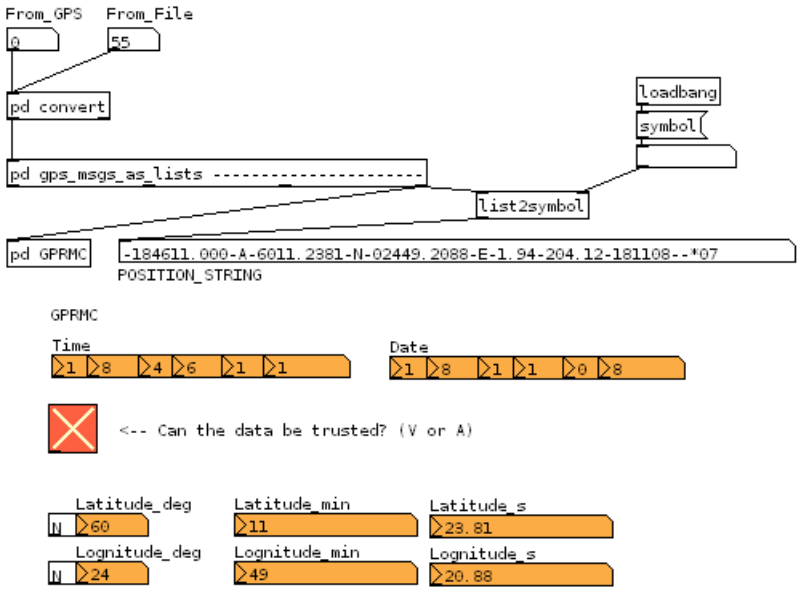

Figure 4.4: GPS sentence handling in Pure Data. The input is received either from GPS device, or extracted from ARA-Wav file. The array of integers is converted to an array of ascii characters. List recognized as `GPRMC` is output to a full string (with empty symbol as the character separator) and sent to be parsed in `GPRMC` patch. Values extracted from the string can be seen in named fields below.

4.5 User interface - Wii Remote

As the user is walking around outdoors, the interaction between the user and the system should be very straightforward. With current setup the GPS and the Control Unit (i.e. the computer) can be kept in a backpack if a Wii Remote is used as a controlling device. The Wii Remote operates over the Bluetooth, so there are no cables to disturb the user. Since it is a controlling device for very popular Nintendo Wii console, the layout of the buttons is already familiar to most of the users (see Fig. 4.5). It provides six buttons to be used easily without eye contact (left, right, up, down, A, and B buttons), accelerometers within the Wii Remote could also be easily used as a controlling mechanism. It is equipped with a wrist strap and it fits easily in the pocket.

A vibration motor within the Wii Remote (Rumble in Fig. 4.5) combined with the headset user is already wearing, provides us a possibility to create a non-visual user interface to be used on the road. For instance pushing buttons A and B simultaneously starts and stops the recording, left and right buttons rewind and fast forward the already recorded sound. All the actions are confirmed to the user by the system with auditory response. If the user tries

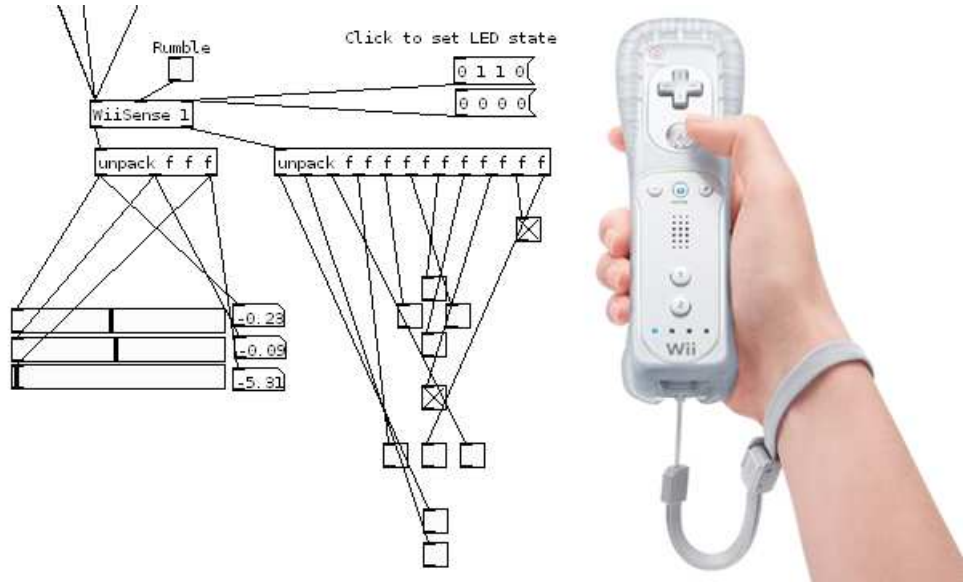

Figure 4.5: PD patch 'Wiisense' to receive controls from Wii Remote [45]. The information from the accelerometers of the Wii Remote can be seen on the left. In the middle the boxes representing the WiiRemote buttons are positioned accordingly, i.e. the user is holding down A and B buttons. The picture on the right is from [46]

to do something that can't be done – for example fast forward from current time instance to the future – tactile feedback is used to strengthen the audio feedback.

The implementation of the Wii Remote as a control device has not been implemented in platform level but left to application level. The implementation is quite straightforward with a Wii Remote interface for PD [45] (in Windows) and [47] (in Linux).

4.6 Control Unit - Pure Data

The core of the control unit was implemented with already existing Pure Data internals and externals. Basically the Control Unit is nothing but a collection of patches to implement the platform. Figure 4.6 visualizes the main view of the platform. On the left hand side all the measured data can be seen; `X`, `Y` and `Z` refer to the accelerometer values and roll, pitch and yaw to the values received from angular rate sensor. Heading value is absolute compass heading received in this implementation directly from the SHAKE device, but could be calculated from electromagnetic field and angular rate sensor output also. The position is presented in degrees, minutes and seconds of latitude and longitude and also as decimal interpretation of latitude and longitude in symbol `POS_FULL`. The big 'Fixed' checkbox tells whether the position value is to be trusted or not (see Section 4.4 for details).

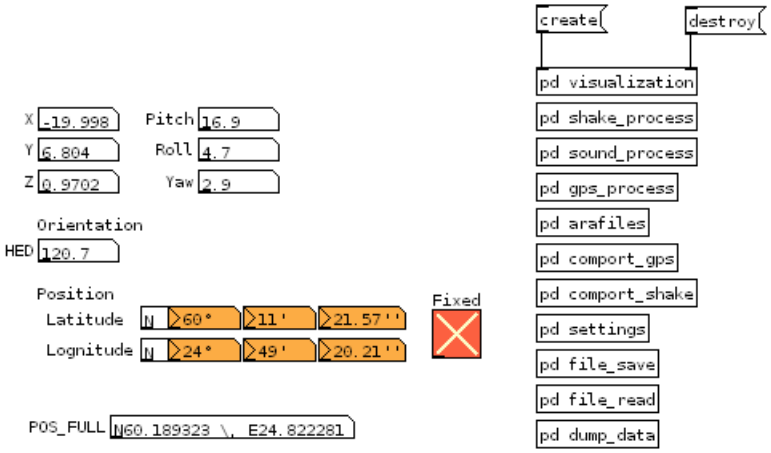

Figure 4.6: Control Unit

All the data gathered from different devices is sent as Pure Data symbols allowing the use of the data whereever needed; one can just mark symbol box to receive for example symbol `heading` and use the received value for whatever purpose. This enables the use of the information independent of receiving or calculating it; if the heading was to be calculated using electromagnetic field and angular rate received from the SHAKE device, the handling and calculation of the heading information would be changed accordingly, but the heading would still be sent as `heading` symbol. No changes to the use of heading information would be necessary.

There are settings the user may have to adjust according to the system used. The COM ports to connect the GPS and SHAKE devices vary on the system, and can be set to correspond to the system within the settings patch from the main view (Figure 4.6, '`pd settings`'). There are also links from the main view to both `comport_gps` and `comport_shake` which correspond to the comport settings of each device (Figure 4.6, '`pd comport_gps`' and '`pd comport_shake`').

The processing – i.e. calculating and parsing the data received from the devices – is performed in separate patches for each device. Patches to `shake_process` (Section 4.3), `gps_process` (Section 4.4) and `sound_process` can be accessed from the main view (Figure 4.6, '`pd shake_process`', '`pd gps_process`' and '`pd sound_process`'). PD patch `sound_process` is used in sound processing when adding HRTF-panned virtual audio to the pseudo-acoustic audio environment. The sound processing is greatly dependent on the application in question, and will not be explained here as a part of the platform but in Chapter 5 as a part of the application.

4.7 Data saving of Audio, position and orientation

There are various ways to save the gathered data with the audio. The obvious way is to write the audio in the file, and keep the orientation and position data in memory. But often there is a need to afterwards get the information where the user was, which direction the user was facing and what did the user hear at the certain moment.

The position and orientation data could be saved in a log file with timestamps. Audio and orientation and position data can be afterwards linked to each other using the timestamps. This approach requires accurate synchronization between the data and the audio. Two files have to be kept in memory at the same time, and have to be kept together in the file system also to make sure nothing goes missing. If either of the files gets lost, the missing data cannot be extracted anywhere.

Another approach is to save the orientation and position data within the audio file as metadata, which revokes the need for inter-file linking. One option is to reserve a separate channel for metadata. This would be good in streaming the audio with metadata since the metadata would be automatically read in right time instance and can be immediatelly combined with present audio data. However, there are two major drawbacks in this approach; this is not allowed in wav specification [48] since all the channels are supposed to contain audio. This will prevent the use of files created in this way to be played in any standard player. Even more severe drawback is the massive amount of space used. Normal sampling frequency is 44 100 Hz. If each sample is saved with 16 bits, 88 200 bytes have to be reserved for each channel for one second and 44 100 Hz * 16 bits/sample * (1/8 bits)/byte * 60s = 5,3 GB / minute. In Section 4.7.1 the chosen datapoint is described as a 90 bytes long structure. To save the metadata 25 times per second for one minute 90 bytes * 25 Hz * 60 s = 135 kB space is required. [1]

Another option to save the metadata in the same file with the audio is to save the metadata in a separate chunk and only supply the metadata with position of corresponding audio sample. In normal wav-files there can only be one data chunk, which has to be continuous, and therefore the chunk for metadata has to be located either before or after audio data. This prevents from streaming while still recording, unless sending the metadata afterwards is acceptable. In order to have the metadata available when playing the audio, the metadata should be sent before sending the audio data. And the same applies to recording the position and orientation data with the audio; the maximum time of the recording has to be known beforehand to reserve sufficient amount of space for the metadata.

If the duration of the audio – and therefore the space requirement for the audio and the

[1] 25 data samples / second was chosen since that is the current maximum limit of the SHAKE orientation tracker.

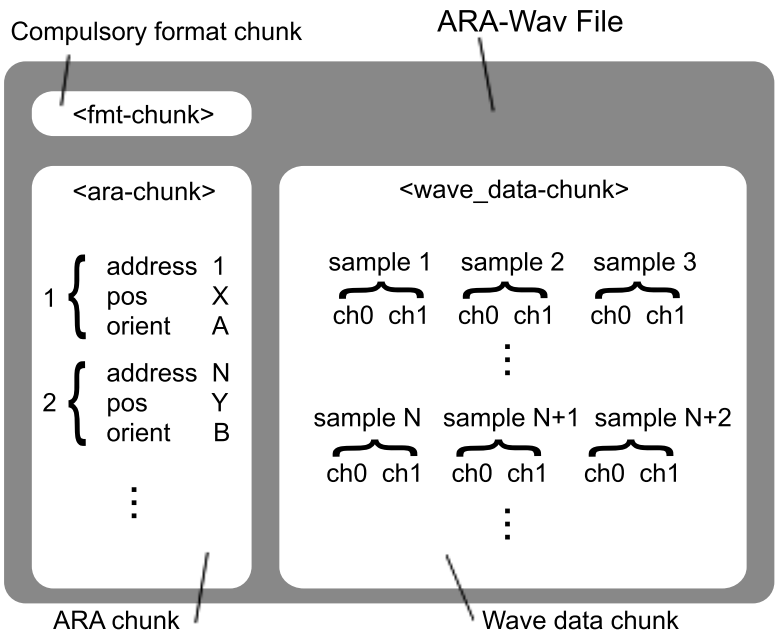

Figure 4.7: The Structure of an ARA-Wav file.

metadata – is not known, the metadata could also be saved after the data chunk. But if that is the case and if any errors occur, all the metadata can be lost since the metadata can only be written to the file very last after saving the full length of audio data.

To avoid synching problems, to save required file space and to keep the files playable with all standard players, saving the metadata in a separate chunk was chosen. The additional data will be saved within the WAV files in an ARA chunk. This chunk belongs to a wav-extension called ARA-Wav, which was first introduced in [49].

4.7.1 Saving position and orientation data in ARA-Wav files

The structure of an ARA-Wav file is shown in Fig. 4.7. Mandatory ’fmt ’ and ’data’ chunks are presented as required by the WAVE format specification ([48], see Section 2.5.1). The additional ’ara ’ chunk is located after ’fmt ’ chunk. The detailed contents of the ARA chunk can be seen in the following c-style presentation:

```
typedef struct ara_chunk {
  long chunk_id;
  long chunk_size;
  long point_amount;
  AraPoint* points;
}AraChunk;
```

The identifier for chunk (`chunk_id`) is 'ara ' (with a space at the end), and as required by the RIFF specification, chunk size is calculated excluding `chunk_id` and `chunk_size` itself. A `point_amount` holds the total amount of data points saved.

In `points`, array space is reserved for AraPoints. Since the 'ara ' chunk is located between 'fmt ' and 'data', the program has to reserve beforehand a sufficient amount of space to make sure the ARA data does not collide with the audio data. For one minute long files, $25 \times 60 \times sizeof($`AraPoint`$)$ of memory was reserved to save the orientation and position data 25 times per second.

The AraPoint presented in c-style is:

```
typedef struct ara_data_point {
  long identifier;
  long pointPos;
  float heading[3];
  char GPSPosition[70];
}AraPoint;
```

The `identifier` is used to differentiate the data points, and it has to be unique within the same file. In the current implementation the running numbering starting from '1' within each file was used. The connection between a data point and audio related to the same time instance is marked with a `pointPos`, which is the position of the sample within the file. The actual information saved consists of `heading`, saved as three floats (yaw, pitch and roll; in this order), and `GPSPosition`, saved as an array of characters. The GPS position is saved as a full NMEA Recommended Minimum sentence C (GPRMC) to make sure a standard string was used not to limit the possibilities of use later on.

The actual saving of the audio file and metadata is done in Pure Data external since the metadata is included – existing PD externals implement only audio saving. SndObj library [43] was used as a basis of the implementation. SndWave class from SndObj library was extended to include writing and reading of required ARA chunks. The existing functionality of SndObj and Flext was used to save the audio.

Since the reading and writing of ARA chunks are implemented in SndObj, for example a standalone command line extractor for extracting metadata from ARA-Wav files can be implemented. The need for saving the file depends on the application, therefore Pure Data implementation of saving and reading ARA-Wav files is presented in Chapter 5.

Chapter 5

AudioMemo

In this chapter an Audiomemo application is introduced, and it is built on the platform presented in Chapter 4. The implementation decisions are explained, followed by testing and analysis of the application.

5.1 Audiomemo application

The basic idea of the Audiomemo application is to provide the user with a browsable recording of everything the user has heard; all the audio the user is hearing is recorded, and position and orientation information is saved as metadata along with the audio. Afterwards the recorded audio memory can be browsed and the metadata can be extracted.

Possible use cases include: after being introduced to a new person the user can browse back in recording and re-listen what was the name of the new acquaintance. The user can see the path walked on the map and listen to the recorded sound environment at the specific location. The application can be used in research to help perform user tests in a real environment. The walked path, audio environment heard and the orientation of the head of the user can be saved with the application. The metadata can be extracted and analysed later.

A user of the Audiomemo application on the MARA outdoor platform can be seen in Fig. 5.1. The user is wearing a visor, which has the SHAKE device integrated to track the head orientation. In his ears he has the ARA headset and in the backpack the user is carrying the GPS device, the ARA mixer and the computer, which is running the application on Pure Data. On his hand the user has the Wii Remote to control the system.

Figure 5.1: A user of the Audiomemo application, which is implemented on the MARA outdoor platform.

5.1.1 Audiomemo architecture and design

The structure of the Audiomemo application is based on the platform presented in Section 4 (see Fig. 4.1). The ARA headset and mixer are used to record and play back the recorded audio. The GPS device is used to track the position of the user, and the SHAKE device to track the orientation of the user. The Wii Remote is used as a user interface to start and stop recording, and to browse the recorded and saved audio. The Control unit connects all the devices mentioned. The required functionality to extend the platform to implement the Audiomemo application is added to the Control Unit.

The added tasks to implement the Audiomemo application consist of:

- Start recording
- Stop recording
- Browse recorded audio
- Dump metadata into a text file

5.2 Saving recorded audio with metadata

After the recording is started, the audio is saved with metadata in ARA-Wav files. The files are named according to the starting time of the saving, using the format:
`ARA-save-[year]-[month]-[date]_[hour]-[minute]-[second].wav`
Maximum length of audio per file is set to one minute, after which new file with new time stamp in filename will be started automatically.

The operation of the system is described in three layers. The use of the external in Pure Data is shown first. The PD external is implemented with Flext C++ development layer; the operation of the external on Flext level is described next. Finally SndWaveAra class, the class called from Flext level to actually save the metadata and the audio in the file, is presented.

5.2.1 arasave~ external in Pure Data

The Pure Data external to save ARA data, `arasave~`, can be seen in Fig. 5.2. Microphone signals from the ARA headset are received from `adc~` object and directed to the first two inlets of `arasave~` after adjusting the gain.[1] The orientation value – `HED` – received from the SHAKE device representing yaw, and values for pitch and roll, are directed to the third, fourth and fifth inlets of `arasave~` object. The position string received from the GPS parser is directed to the sixth inlet and the seventh inlet is used to receive control commands, like 'start' to start recording and 'stop' to stop recording.

[1]The gain value can be set in `pd settings`, accessible from the main view of the platform, Fig. 4.6

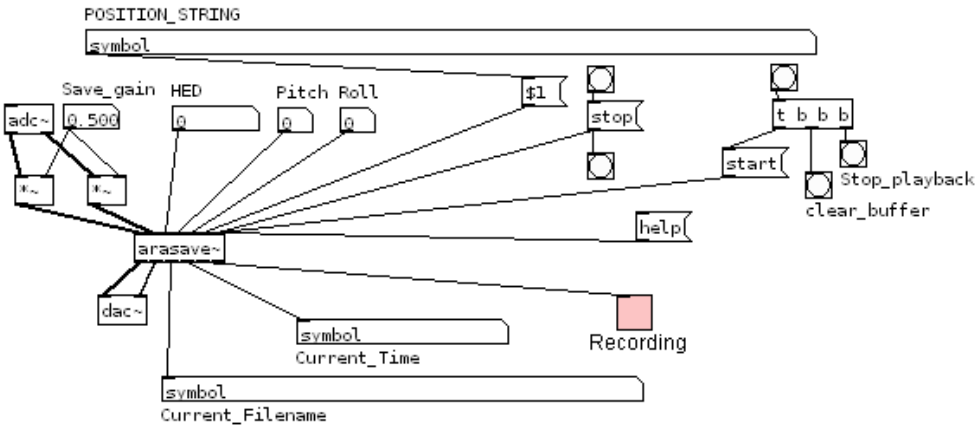

Figure 5.2: Saving ARA file in Pure Data

The bang buttons originating from 'start' and 'stop' messages in Fig. 5.2 are sent to metadata dump to clear the buffer on start and to dump the metadata to a textfile. The playback is stopped before starting the recording. See Section 5.5 for more details.

The first two outlets of `arasave~` are the outputs of the recorded sound. Since the ARA mixer is already feeding through the live sound to create the pseudo-acoustic audio environment, there is no need to connect the output of the external to `dac~` to be played again. The third and fourth outlets provide the current filename and current time as strings to be shown to the user to help find the saved files afterwards. The fifth outlet is used to give the user indication whether the system is currently recording or not.

The target file in recording will be automatically changed when the maximum length of the file is reached. This is handled in the external automatically, and the user only sees the difference in changed filename.

5.2.2 Flext implementation of arasave~ external

Flext C++ development layer is used to implement portable code for the external with it's built-in support for SndObj library, as explained in Section 4.1. The base class *flext_sndobj* is used in `arasave~` external. The base class provides the interface and basic functionality of the external and there are three virtual functions *NewObjs()*, *FreeObjs()* and *ProcessObjs()* which have to be overridden in a derived class to implement required functionality. [50]

As mentioned in the previous section, the maximum length of audio per file is set to one minute. This length was set to limit the losses in case an error occurs. The maximum length of the file has to be known beforehand, since sufficient amount of space has to be reserved for the metadata in the file. The space is reserved between 'fmt ' and 'data' chunks (for details, see Sections 2.5.1 and 4.7.1). The maximum length setting can be adjusted in *SndWaveAra* class, see Section 5.2.3.

When starting the recording, a new instance of SndWaveAra is created with parameters consisting of filename, permission to overwrite the file if one with the same name already exists, number of channels (2), bits/sample value (16), input list (0, none), starting position (0), and blocksize and sample rate from Pure Data. The filename is constructed with the current date and time to make it easy to differentiate between the files and to sort them.

The received signal is scaled from values in range $[-1, 1]$ to cover the range of integer short values. If this was not done, the conversion from float to integer would result in only three possible values; -1, 0 and 1. The scaling is done by setting the gain multipler to `SHRT_MAX` (32767 in 32 bit operating system. In theory the multiplier should be 32768 [2]

[2]The minimum value, `SHRT_MIN`, equals -32768 in 32bit system

on the negative side, but the difference is considered unnoticeable.

NewObjs(), *FreeObjs()* and *ProcessObjs()*

NewObjs() and *FreeObjs()* are used to create and destroy the objects when starting and shutting down Pure Data. The sound objects used in the external and the gain value are set in *NewObjs()* function. When the external is closed, all the sound objects are automatically destroyed within *FreeObjs()* function.

ProcessObjs() function contains the main signal processing functionality of the external. Pure Data handles the audio in blocks. On every block the *ProcessObjs()* function is called. Block size may vary depending on the system; on Windows XP and Pure Data version 0.39.3-extended the block size is 64 samples. This is very important to note since the default block size in SndObj library is 256. The block size is fixed in Pure Data, so SndObj blocksize has to be changed to correspond to PD (in SndObj library the block size is referred to with variable `vecsize`)

Every time the *ProcessObjs()* -function is called, i.e. once on every block, the signal processing is performed. The gain of samples is adjusted with the gain value set in *NewObjs()*. The block is sent to *SndWaveAra::Write()* function to be written to the open file. Since the update rate of measuring instruments is significantly lower than 689 Hz (which is the number of blocks in one second, since 44100 Hz / 64 (samples/block) = 689,0625 (blocks/second)), saving the measuring results with each block would be a waste of space. *SndWaveAra::PutAraPoint(orientation, position)* function with every 27th block (*floor*(44100 Hz / (25 Hz * 64)) = 27). Saving metadata 25 times per second was chosen since that is the maximum update rate of heading information from the SHAKE device.

When the maximum length of the file is reached, i.e. at least `DATA_POINTS_PER_FILE` blocks have been saved, the current file is closed and a new one is opened immediately with the name containing the current time. This check is performed in *ProcessObjs()* function. `DATA_POINTS_PER_FILE` is the number of data points the SndWaveAra class has prepared to save.

5.2.3 SndWaveAra class operations with arasave~ external

SndWaveAra is an improved version of *SndWave* class belonging to the *SndObj* library [43]. *SndWaveAra* implements *SndFIO* class, which is the File Input/Output class of *SndObj* library. *SndWaveAra* class is used for the file operations needed in `arasave~` external. *SndWaveAra* class could be used without Flext and Pure Data to implement e.g. a standalone metadata extractor.

When *SndWaveAra* class is first called from the `arasave` external the parameters are

used to adjust the settings. Given filename is opened and the space is reserved for the metadata before audio. Permission to overwrite the file is either granted or denied as a parameter. Other parameters include number of channels, number of bits used to save one sample, list of other SndObj inputs to be used, starting position of recording in file and blocksize and sample rate to be used.

The actual saving of the audio and metadata is performed in *SndWaveAra::Write()* function. The operation is identical to the one in *SndWave* class. The block is looped through and all the samples are written in the file.

The metadata is written in ARA-Wav file in *SndWaveAra::PutAraPoint* function. The parameters consist of orientation (as a `float [3]` value) and position (as `char [70]`). The exact definition of the AraPoint struct is presented in Section 4.7.1.

When the recording is stopped – either by the user or to change the file on the fly – the wave data header and ara chunk header are written again. The writing of Wave header is identical to the one in SndWave class. The idea of saving the wave data and ara chunk headers very last is to provide the headers with the information that could not have been known before; e.g. the actual audio length in the file, and the number of saved AraPoints. If the header of the data chunk is not saved very last, the players can't play the file since they can not interpret correctly the length of the data chunk.

Constants to define the metadata saving rate (`DATA_SAMPLES_PER_SECOND`, currently 25), file maximum length (`SECONDS_TO_RECORD`, currently 60), and block size in Pure Data (`PD_BLOCK_SIZE`, currently 64) can be changed in SndWaveAra.h file. Values for constants `DATA_POINTS_PER_FILE` and `BLOCKS_BETWEEN_DATASAVE` are calculated accordingly. These options most probably don't have to be changed, but if changed, the external has to be recompiled and Pure Data has to be restarted.

5.3 Browsing the recorded audio

In the current implementation simultaneous playback while recording is not possible; the recordings can be listened to only after recording has been stopped. The playback can be started from the first file in the directory, or by browsing backwards from the current position. Although the audio lenght of one file is limited to one minute, the recordings are played as one continuous recording by automatically opening the next file according to the filename. There is no separation between recording sessions, but all the recordings in the directory are simply sorted and played in order, unless stopped by the user. If no file is specified, the first file in the directory is opened.

Like with saving the audio and metadata, the operation of the system is described in three layers. The use of the external in Pure Data is shown first, this is followed by the operation

of the external on Flext level, and finally SndWaveAra class implementation is presented.

5.3.1 araread~ external in Pure Data

The ARA-Wav files are read in Pure Data with `araread~` external (illustrated in Fig. 5.3). Messages used to control the playback are directed to the first inlet of the `araread~` external. The messages are allowed to pass only if the application is not currently recording. This is checked in PD patch `ifNotRecording` which passes the message forward to the external if the playback is allowed, and blocks the messages and notifys the user if not. If playback is allowed both "start" and "stop" messages are routed also to send clear and dump messages to the metadata buffer, respectively. The metadata buffer is cleared everytime the playback is started, and dumped to a file everytime the playback is stopped. This is explained in more detail in Section 5.5.

When the end of the file is reached, the `araread~` external automatically changes the playback to the next file in the directory. No action from the user is required. When the end of the last file is reached the playback is stopped and the user is notified via `notify_user`.

Control messages

The possible messages to control the external are `start` and `stop` to start and stop the playback, `prev` and `next` to jump to the previous and next file, and `first` to directly start playing the first file found in the directory. A brief introduction of the external to the

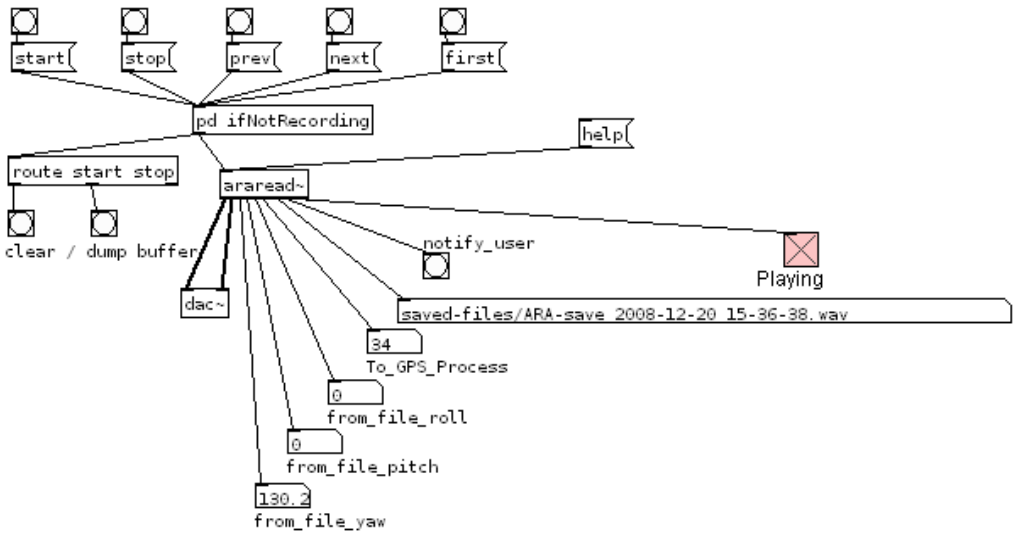

Figure 5.3: Playing back ARA files in Pure Data.

Pure Data main window is triggered with `help` message.

Sending `start` message to the `araread~` external starts the playback of the current file. If no file is defined as current yet, the first file in the directory is used. If the `araread~` external is already playing the file, the current file will be restarted from the beginning. The `stop` message stops the playback keeping in memory which file was played last, which means that sending `start` message again starts the playback of the same file. `prev`, `next` and `first` messages also start the stopped playback. If `next` message is sent while playing back the last file in the directory, the playback is stopped and `notify_user` is sent.

Outlets

The audio of the file is output via the two first outlets to the `dac~` to be played to the user via headphones. The three next outlets are used for orientation information; yaw, pitch and roll orientation of the user. The sixth outlet is used to pass the position string to the GPS parser (see Section 4.4) as a series of ascii characters. Since this is the same format the data is received from the GPS device, no modifications in parsing or using the GPS data are needed. The data received from the GPS device should be blocked while playing in order to show only the GPS data received from the file.

The seventh outlet sets the `current_filename` symbol according to the file currently played. The eight outlet is used to send the `notify_user` message in case of an error, and the last outlet represents the `is_playing` status.

5.3.2 Flext implementation of araread~ external

Similar to the implementation of `arasave~` external, the base class *flext_sndobj* is used in `araread~` external. As mentioned the base class provides the interface and basic functionality of the external and the three virtual functions *NewObjs()*, *FreeObjs()* and *ProcessObjs()* have to be overridden in a derived class to implement required functionality. [50]

NewObjs(), *FreeObjs()* and *ProcessObjs()*

NewObjs() function is called when the external is created, i.e. when the application patch is opened in Pure Data. There are no sound objects to be created in `araread~` external until the playback is started, so there is nothing to do in *NewObjs()* but to return with *true* value. *FreeObjs()* function takes care of all the opened sound objects when the external is closed, i.e. when Pure Data is shut down. Any open file and gain objects, opened in *m_start* function, are destroyed.

The main signal processing functionality of the external is performed in *ProcessObjs()* function. As explained in Section 5.2.2 the audio in Pure Data is handled in blocks. For each block *ProcessObjs()* function is called. The processing in *ProcessObjs()* function consists of gain adjustment and metadata extraction. The gain is adjusted according to the value set when opening the file in *m_start* function. Metadata is requested from the SndWaveAra file instance with *SndWaveAra::GetAraPoint* function (explained in 5.3.3). The metadata is forwarded to the outlets, as defined in Section 5.3.1.

At the end of *ProcessObjs()* function a check is performed to find out whether all the datapoints stored in the file have been processed. This is considered as an implication of reaching the end of the file. The playback of the current file is automatically stopped and the playback of the next file in directory is automatically started if possible. If current file was already the last in the directory, the playback is stopped and the user is notified via `notify_user`.

m_start(), *updateToNextFilename()* and *updateToPrevFilename()*

When the user wants to start the playback of saved ARA-Wav files `start` message is sent to the external. The message triggers *m_start* function, in which the file to be opened is found out, the SndWaveAra instance is created and the playback is started. The Flext class, *araread*, holds the `current_filename` as a protected variable. If the variable is not yet set *updateToNextFilename* function is called to set the `current_filename` if possible. If not, the playback is stopped and the user is notified via `notify_user`. New instance of SndWaveAra is created with parameters consisting of filename, mode (`READ`), number of channels (2), bits/sample value (16), input list (0, none), starting position (0), and blocksize and sample rate from Pure Data.

The gain adjustment is set in *m_start* function. As described in Section 5.2.2 the gain of the written audio was adjusted to scale the values from $[-1, 1]$ to cover the range of integer short values. When reading the integer values from the file, the samples have to be rescaled back to float values between $[-1, 1]$. This is done by adjusting the gain of the signal by -90.3 dB. [3]

Boost Filesystem library [51] has been used for *updateToNextFilename()* and *updateToPrevFilename()* functions to keep the codebase portable between different operating systems. Boost Filesystem *directory iterator* is used to browse through all the .wav files in `saved-files/` directory. In both *updateToNextFilename()* and *updateToPrevFilename()* functions the first file encountered is used if `current_filename` has not been set pre-

[3] Multiplier 1/SHRT_MAX ($= 1/32767$) can not be used, since as a float value it is rounded to zero, and adjusting the gain with the multiplier of zero effectively mutes the signal completely. Instead, the multiplier 1/SHRT_MAX can be replaced by attenuating the signal by -90.3 dB (since $20 \log_{10}(1/32767) = -90.3$)

viously. If the previously set `current_filename` is the last file in the directory the function will return 0 to indicate error. With successful update of `current_filename`, the return value 1 is used to represent successful operation.

5.3.3 SndWaveAra class operations with araread~ external

SndWaveAra class is called first from *m_start* function in `araread~` external to create SndWaveAra instance. The parameters received with creation call are used to adjust the settings and to open the file with given filename in `READ` mode. Other parameters include number of channels, value of bits used to save one sample, list of other SndObj inputs to be used, starting position of reading of the file and blocksize and sample rate to be used. These additional parameters are overriden if values are found in the opened file.

The reading of the audio is very straightforward and is performed in *SndWaveAra::Read()* function once for each block when called by *ProcessObjs()* function (as presented in Section 5.3.2). The values of the data at the current file position are read and put in the output array.

The reading of the AraPoint is requested with each block from *ProcessObjs()* function, but the values are returned only after each passing of the AraPoint. The AraPoints are saved in the file in the order of appearance and with the information of the file position of the audio related to that specific data point. The position of the next AraPoint, `next_point_position`, is compared to the current position in the file. When the current position equals or exceeds the anticipated position of the next AraPoint, contents of the next AraPoint is returned, and `next_point_position` is updated to correspond with the following AraPoint. This approach ensures that the data of each AraPoint is returned within `block_size` / `samplerate` ($= 64/44100 = 0,001$) seconds of the exact time even if the `block_size` used in saving the file has been different than the `block_size` when reading the file. If the `block_size` differs between the reading and writing of the file, exact matching with the audio position and position of next AraPoint can not be used since there is no guarantee that the *ProcessObjs()* function calls the *SndWaveAra::GetAraPoint()* function at the exactly same time instance.

5.3.4 User notify

In the Audiomemo application, a common Pure Data symbol is used to mark the need to notify the user. The notification to the user is sent as `notify_user` bang -message, which can be received anywhere in the application patch. `notify_user` is used to send the user feedback via Wii Remote and SHAKE vibration motor. This tactile feedback is used in the current implementation, but auditory feedback can be used instead. Notifying the user is discussed in more detail in Section 5.4.

5.4 User interface

Currently there are two possible ways to control the platform and the application; directly from the computer screen using mouse to control the Pure Data, or by using Wii Remote. Not everything can be done via Wii Remote, but after the application has been started all the functions needed on the road can be accessed with Wii Remote buttons.

Connecting the functionality of Wii Remote to the application is very straightforward in Pure Data. All the pressings of the buttons are received as states of the buttons; 1 for pushed button, 0 for released (the toggle boxes in Fig. 4.5). All the toggle boxes have been defined to send the value of the box as symbols like `wii_a` for A-button, `wii_left` for left button or `wii_home` for the home button. Setting a toggle box to receive the corresponding symbols enables the connecting of the button press to a certain action. This can be seen in Fig. 5.4 where left button is tied to "jump to the previous file" functionality and right button to "jump to the next file". Home button is connected with "jump to the first file" and `notify_user` to the tactile feedback[4] of the Wii Remote.

The starting and stopping of recording is implemented as a simple state machine, which starts and stops the playback in turns every time the A and B buttons of the Wii Remote are pushed simultaneously. When the recording is started, the playback is stopped first and the recording is started right after.

The restrictions for example in starting the playback do not have to be taken care of here.

[4]refered to as "rumble" with the context of Wii Remote

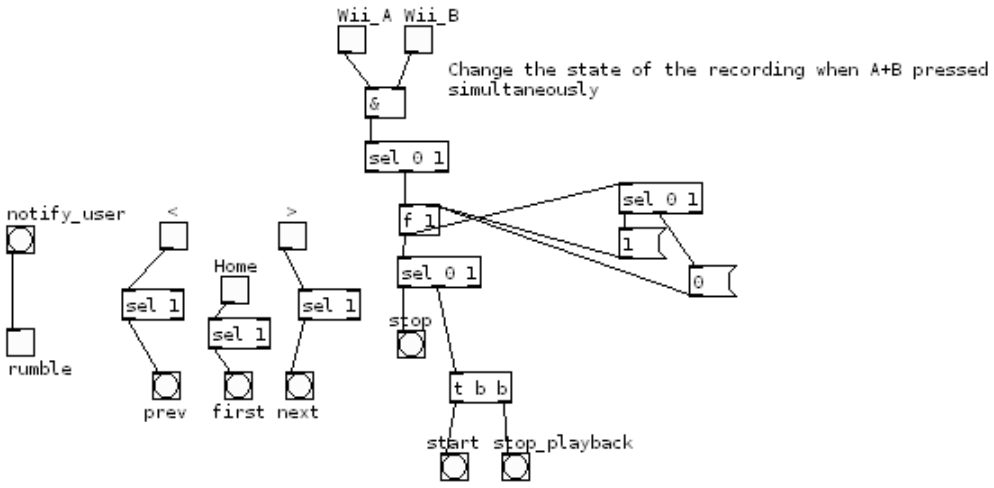

Figure 5.4: Pure Data patch to define the connections between Wii Remote buttons and functions in the application.

Restrictions are implemented in `pd file_read` PD patch and checked in `ifNotRecording` patch (see Section 5.3.1). This ensures that the permission to start playback is always checked right before starting the playback where it is needed, not everytime before sending the command. In other words, if recording has already been started and the user pushes home button, which is tied to the "jump to the next file" function, the command is sent, but before the command is executed the permissibility of the action is verified.

Another alternative to control the platform would be with a computer equipped with a touchscreen. The PD patch could be modified to contain bigger controls and all the commands could be done by pointing the screen. This would enable more possibilites for functions of the application and more detailed feedback and interaction with the user compared with the use of Wii Remote, but the downside is that the platform is intended to be used outdoors, where the light can be too bright for the screen. Furthermore, the user would have to carry the computer in hands, which is not very convenient for any longer period of time. For these reasons, a touchscreen option is not taken into account in this version, but the Wii Remote controls are favored.

5.5 Metadata dump to a textfile

When starting or stopping the recording or the playback, the metadata is automatically saved as a textfile in `saved-files` directory. The textfile can be modified as needed and imported for example into Matlab to analyze the recorded data, or to map the path walked in Google Maps (see Section 6.3 for an example).

PD patch `pd dump_data` (presented in Fig. 5.5) is used to handle the saving of the metadata in text format in separate files. This is done since the standalone extractor has not been implemented yet, and the current versions of Matlab or Google Maps do not contain the support for ARA-Wav files, nor does any other player yet. Therefore the buffers from both recording and playing back the files are dumped.

Everytime the recording or the playback is started, the metadata buffer is cleared to make sure only the current metadata is saved. New line in metadata buffer is added with each update of the heading information, i.e. with the frequency of 25 Hz. This functionality has to be noted if not all the measuring devices are used. The position information and the timestamp are currently directed into the "cold" inlets of the `sprintf` object, which creates the line to be saved. Only the update of the heading information triggers the creation of a new line, since that is the one directed to the "hot" inlet. The "cold" inlets save the data fed to them, but only the "hot" inlet also triggers the action. This is a basic functionality in Pure Data, and just has to be taken into account.

When recording or playback is stopped, the current contents of the buffer is saved in the

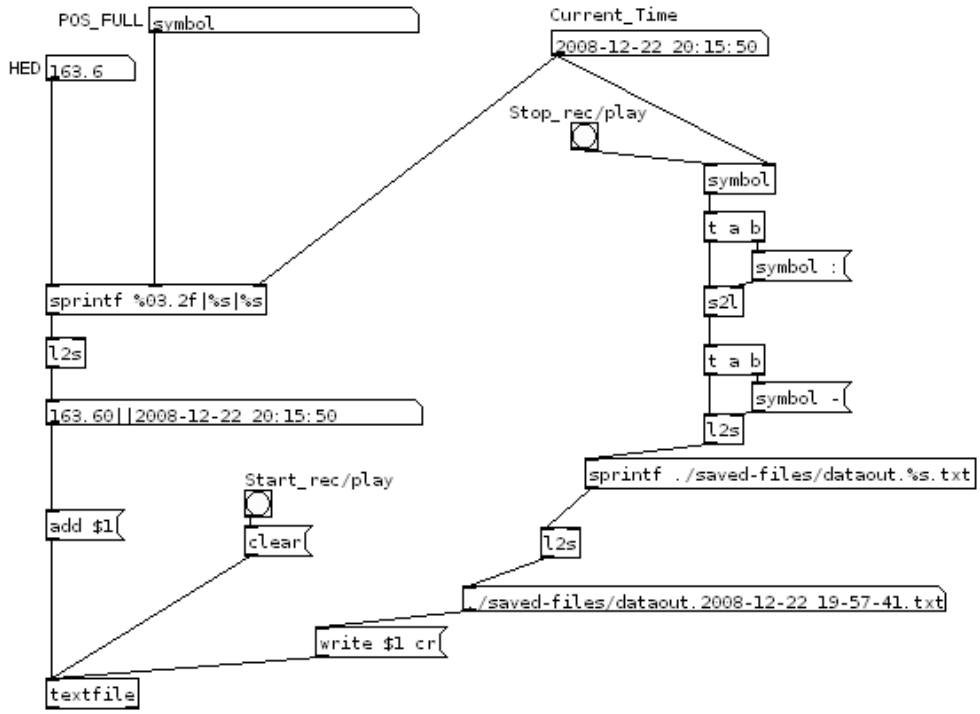

Figure 5.5: Dumping the metadata to a textfile. On the left hand side new lines to buffer are created everytime the heading information is updated. On the right hand side the filename is created using the timestamp when recording or playback is stopped.

form of prefix `dataout` concatenated with the current timestamp. Timestamps are used to help finding the correct datafile after the recording. One has to note that the timestamp is of the moment when the recording or playback was stopped, not when it was started.

Chapter 6

Testing and Analysis

In this Chapter the implemented Audiomemo application is tested and examined to find out whether the requirements set for the platform in Section 3.1 have been met. No comprehensive user studies have been performed, but the evaluation is done by comparing the implementation with the requirements, and by performing functional tests on the system. The functionality is evaluated by walking around the Otaniemi campus area recording the audio and the position and orientation of the user and comparing the recorded results with the actual path walked.

6.1 Requirements for the platform

6.1.1 Binaural recording with metadata

"The system should be able to record and playback binaural audio in real time."

The recording and the playback can be performed in realtime with no audible artefacts caused by the platform. Currently when listening to the recorded audio, the instant playback of the pseudo-acoustic audio environment is not muted. Considering safety reasons this is a good thing; e.g. cars driving by could be left unnoticed by the user if the surrounding environment was completely blocked. However, the quality of the audio played is reduced since the user hears effectively two audio environments on top of each other; the virtual audio from the file, and the pseudo-acoustic audio from the real life. The volume of the pseudo-acoustic environment should be adjustable from the control unit to reduce the mixing when needed. To clarify; the problem does not occur with pseudo-acoustic audio environment enhanced with a simple virtual audio, but when two complex and similar audio scenes are combined.

"Recorded sound should be indexed with position and orientation information."

The position is saved using maximum of 70 characters, and the orientation is saved using

three float values to represent the yaw, pitch and roll of the head of the user. This means that the 6-DOF position of the user can be saved within the audio file as metadata. Each datapoint is annotated with the exact file position of the corresponding audio. The Audiomemo application on MARA platform currently implements only 3+1-DOF position of the user, since orientation is only tracked according to the yaw angle of the user, but the same format of the datapoints is used, and this can be considered as a special case when the user only turns about the z-axis but does not tilt or roll the head. In other words, the system works as required and is already prepared for further improvements.

6.1.2 Independence

"The system should be location independent, i.e. work in real environment." and *"Position and orientation tracking should be global."*

The platform itself does not restrict the area of the usage in any way, since the chosen tracking method – GPS – can be changed to another tracking method if a better one is available. The GPS has severe restrictions with indoor coverage and some difficulties in the city environments with high buildings, but the platform can be considered usable anywhere in the world. Furthermore, the platform is not dependent on any other system or equipment.

6.1.3 Portability and extendability

"The platform should be OS independent (Windows, Mac & Linux supported)" and *"The platform should be extendable."*

The platform is implemented with software that is available to all mentioned operating systems, using libraries which are portable too. The binaries cannot be transferred directly, but the Flext based C++ code can be compiled without any code changes. The porting has not been done yet to any other operating system, so all minor difficulties are not promised to be non-existent.

The platform is not tied to any specific equipment. If better tracking devices or recording and playback equipment are developed, they can be taken into use with minor or no changes to the code.

6.1.4 User interaction

"The equipment should be light enough to carry with ease." and *"There should be a reasonable way for the user to communicate with the system."*

The biggest part of the platform is the computer, which is rather small and light to carry nowadays. With current devices the cords are a minor distraction, but since the computer, GPS device and the ARA mixer can be put in a backpack, and the application operated with

the Wii Remote, the inconvenience is easily tolerable. The Wii Remote as a control device proved to be even better than thought beforehand, since it revokes all the need to open the computer when on the road with the platform.

6.2 Setting up the system

The biggest problems with the devices of the platform seem to arise when setting up the system for the first time. Bluetooth connection works perfectly once the connection is set up, but in some cases adjusting the settings can be rather difficult. If the Bluetooth device is improperly disconnected – e.g. the computer is shut down – the reconnecting may be even more difficult.

Depending on the operation system used the pin code confirmation on pairing the SHAKE device with the computer may be optional (Windows XP) or required (Windows Vista). The latest firmware of the SHAKE device already includes the functionality to use Bluetooth security. The pin code is always the last four characters of the mac address of the device [31]. Based on email conversation with the representetative of the SHAKE manufacturer (Stephen Hughes, SAMH Engineering Services, Dublin, Ireland) Windows Vista probably keeps track of the paired devices by the mac address. If the connection is promptly shut down – e.g. by shutting down the computer – the operating system may still consider the device paired because it is still on the list. And for this reason the repairing does not happen even when the SHAKE device considers the connection closed. Operation on Windows XP was found working better.

There are some restrictions on the serial port numbering in Pure Data. The serial port could not be used if the port number was over ten. This limits the options in choosing the Bluetooth driver, since not all of them allow the user to choose the serial port number. There are also differences in connecting the Wii Remote with the Bluetooth driver. Some drivers (e.g. Toshiba) automatically identify the Wii Remote as a human interface device (HID), and with some drivers the connection has to be created every time specifically telling the driver to handle the Wii Remote as a HID (e.g. Widcomm drivers).

After the initial adjusting of the devices, the only problems were with reconnecting the Wii Remote if the connection was lost. When the computer is in the backpack, it is rather inconvenient to reconnect the Wii Remote since accessing the keyboard of the computer is required. Otherwise all the required actions can be performed with the Wii Remote, when walking outdoors. Processing and analyzing of the metadata still requires the computer screen and the mouse.

6.3 Experiences

Testing of the platform was performed by setting up the system, wearing the equipment and walking around the Otaniemi campus area. Recording and playback were tested while walking, using the Wii Remote as the controlling device. Afterwards the recorded audio was listened to and metadata was plotted on the map. The results are compared with the actual path walked.

6.3.1 Arrangements before use

Wearing the platform on is simple, but care have to be taken with all the cords. Laptop, GPS device and ARA mixer are placed in the backpack, the visor with the SHAKE device attached is placed on the head and the Wii Remote is held in hand. A picture with the setup worn by the user can be seen in Fig. 5.1.

The laptop, which is set in the backpack has to be adjusted to not shut down when the lid is closed. The ventilation of the laptop has to be taken care of; the laptops tend to heat up and if proper ventilation is not arranged, the computer may be damaged.

6.3.2 Experiences in the field

As noted in [52] the pseudo-acoustic environment created with the ARA headset sounds a bit different than the real audio environment, but the adaptation is very fast. The friction of the microphone cords is audible, and somewhat annoying, as is the wind blowing to the microphones, and the sounds of the wind feel a bit amplified.

The Wii Remote fits well in the hand, and when not needed it can be put in the pocket. Led lights at the bottom of the Wii Remote give clear signal when the connection is still open, but only if the user knows to keep an eye on them. As there is no visual confirmation whether the commands have been received or not, the tactile feedback feels very important. Some auditory feedback would also be worth the effort when improving the application.

Some kind of indication and reminder when the application is recording would be advisable. It was noted that it is very easy to confuse oneself whether the recording was just started or stopped. Short vibration or quiet sound every 10-15 seconds would be sufficient to let the user know the recording is still on.

Also when listening to previously recorded audio the possibility to adjust the level difference of the real environment and the recorded audio was longed for. Neither need to be completely muted but possibility to adjust them would make it easier to differentiate the sources if wanted.

6.4 Analyzing the recorded information

The recorded audio was listened to afterwards. The quality varied quite a lot depending on the wind, and the gain settings. It was noted that the microphones, even the external ones, were set very sensitive in laptops with integrated microphones. If the gain was not set to attenuate the signal in plenty, the recorded audio would be overdriven. This varies a lot between computers, and the gain has to be set individually.

If the recordings were listened to shortly after recording it was easier to localize the audio. The same applies when the walked path was plotted on a map, and followed while listening to the recording. The auditory events were mostly cars and busses driving by, probably few clearly stationary sources could have been even easier to localize afterwards.

Currently there is no automation in plotting the walked path on the map. The dumped data in the textfile was modified with a word processing program to convert data lines from `248.90|N60.188911  E24.826063|2008-12-23_16:40:04` to Google Maps format `new GLatLng(60.188911,24.826063),` for each datapoint along the path. These lines can be put in a simple HTML file to map all the points as a part of the path, and for example once in a minute as a marker with time information and fan shaped icon to represent orientation. An example of plotting the walked path in the Google Maps can be seen in Fig. 6.1.

When plotting the walked path in Google Maps there seems to be an offset of approximately 100 meters to the north and 150 meters to the east (visualized in Fig. 6.2). This is caused by the Google Maps using WGS-84 datum and Mercator projection. Mercator projection is used to create square images of the spherical globe, which causes bigger offset towards the poles. In normal use this is acceptable, since the users need relative information from the maps more than absolute, but when plotting the points received from GPS the offset becomes visible. The trade-off has apparently been between absolute accuracy and the speed gained by using pre-calculated map images. [53]

The problem with the partially blocked GPS signal can be seen in Fig 6.3. The user was sitting in a bus with the GPS device in the backpack. The position could be calculated, but only some of the data points are correct. Probably too few signals were available to correctly calculate the position. The offset has not been corrected in the image.

Figure 6.1: Visualization of the walk path and starting points of recorded one minute long samples. The fan is the orientation in the end point of the last recording.

Figure 6.2: The offset error when directly plotting the positions on Google Maps visualised on the left. Corrected version on the right.

Figure 6.3: The positioning does not work properly unless the signal from at least three satellites is received. The recording was done while sitting in a bus. The red line represents the recorded path and the blue line the actual path.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

This thesis has proven that the described ARA outdoor platform can be implemented, as well as applications based on the platform. The system is by no mean perfect, but the defects are caused by the imperfect measuring equipment. Reasonable position and orientation tracking methods exists, but there is yet no silver bullet to offer fast and accurate tracking which works globally.

The platform enables binaural recording of everything the user hears, with no restriction on location. The resulting audio file is augmented with time, position, and orientation information. The information and the audio of a certain time or location can be retrieved using the metadata in the files. The files, and thus the audio environment, could also be sent to other users – if WLAN or GPRS network is available.

The platform is extendable, as required by the specification, and all the devices used are replaceable. As the technology advances, better equipment can be taken into use. Not all the presented devices have to be used if not needed. The platform operates well also with less equipment; the application is already known to been used in Nokia Research Center indoors without position tracking, but for tracking and saving only orientation data with the audio.

A conference paper presenting the platform and ARA-Wav format was written and has been presented in the AES 35th International Conference, Audio for Games, in London, February 2009 [49]. Though the platform was not specifically designed to be used in games, there are numerous possible ways to utilize the platform in that area.

Some mobile phones on the market already offer GPS positioning and include accelerometers and gyroscopes, which could be used for rough orientation tracking of the user. The orientation tracking should track the head of the user, and the binaural audio recording and

playback would be the ideal case. But even monaural recording with only shoulder orientation tracking and position tracking will probably open wide range of possibilities for new applications, especially if the user is wearing a hands-free device that would provide the in-ear sound production and that could be worn almost all the time.

7.2 Future work

There are various ways to improve both the platform and the Audiomemo application. The platform itself is rather usable and convertible for various applications and implementations. Enhancing the orientation and position tracking methods would improve the platform most significantly. Some observations on the ARA headset and mixer were noted also during the testing of the platform. For the possibility to use the platform as a basis of any sort of user experiments outdoors, improving the output of the recorded data would be beneficial.

The easiest way to improve the orientation tracking is to use the angular rate sensors, which already exist in the SHAKE device, and calculate the heading information with the angular rate with the electromagnetic fields instead of accelerometers. This is not implemented in the SHAKE firmware, but the measuring instruments already exists and the calculations can be done in the control unit.

The accelerometers could be used along with GPS to provide more accurate position tracking. The refresh rate of the GPS is rather sparse, and the accelerometers tend to drift over time, but using these two together would improve the situation. GPS should be used to define the absolute position and accelerometers to fine-tune the estimate.

Especially GPS, but also the inertial measuring devices, tend to have some noise in the measurements resulting in small random variation. This could be diminished by filtering the received data. For example Kalman filtering is used in many applications to estimate and predict the positions of objects [54].

As mentioned by Tikander & al. [52] the size of the ARA mixer is one of the most irritating feature of the mixing device, along with the cords that also cause disturbance to the audio. With the platform the need for two extra stereo cords the number of cords is now reaching total of four, which makes the handling of the device even more demanding. Reducing the mixer size and reducing the amount of cords needed would help the situation a lot.

Tikander & al. mentioned also the wind noise as a problem with the ARA headset. This is a important thing to improve, since the platform is intended to be used outdoors. The ears, and thus the microphones, cannot be covered without the quality of audio suffering, but perhaps some acoustically invisible material could be used.

An option to mute, or at least attenuate, the pseudo-acoustic environment signal would

probably be useful in many applications. Currently there is no way to alter the volume from the control unit. It would be easier for the user to make the adjustments for example with the Wii Remote.

The Audiomemo application should be developed further to allow the playback while still recording, and the browsing should be done in smaller steps – i.e. not only jumping from file to file, but also for example 10-15 seconds at the time. Both these improvements are dependent on some more logic to be implemented in the browsing algorithms. An option to attenuate the pseudo-acoustic audio would be beneficial with the "playback while still recording" functionality.

Bibliography

[1] A. Härmä, J. Jakka, M. Tikander, M. Karjalainen, T. Lokki, J. Hiipakka, and G. Lorho, "Augmented reality audio for mobile and wearable appliances," *Journal of the Audio Engineering Society*, vol. 52, pp. 618–639, June 2004.

[2] A. Härmä, J. Jakka, M. Tikander, M. Karjalainen, T. Lokki, H. Nironen, and S. Vesa, "Techniques and applications of wearable augmented reality audio, 5768," in *114th AES Convention, Amsterdam, The Netherlands*, March 2003.

[3] T. Lokki, H. Nironen, S. Vesa, L. Savioja, A. Härmä, and M. Karjalainen, "Application scenarios of wearable and mobile augmented reality audio, 6026," in *The 116th AES Convention, Berlin, Germany*, May 2004.

[4] S. Vesa, "Estimation of reverberation time from binaural signals without using controlled excitation," Master's thesis, Helsinki University of Technology, 2004.

[5] M. Tikander, A. Härmä, and M. Karjalainen, "Acoustic positioning and head tracking based on binaural signals, 6124," in *The 116th AES Convention, Berlin, Germany*, May 2004.

[6] V. Riikonen, "User-related acoustics in a two-way augmented reality audio system," Master's thesis, Helsinki University of Technology, 2008.

[7] L. Seppänen, "Development of an audible sticker application and a video-based tracking system," Master's thesis, Helsinki University of Technology, 2008.

[8] T. P. Caudell and D. W. Mizell, "Augmented reality: an application of heads-up display technology to manual manufacturing processes," *System Sciences, 1992. Proceedings of the Twenty-Fifth Hawaii International Conference on*, vol. II, pp. 659–669 vol.2, Jan 1992.

[9] A. Walker, S. Brewster, D. McGookin, and A. Ng, "Diary in the sky: A spatial audio display for a mobile calendar," in *Proceedings of IHM-HCI 2001, Lille, France*, pp. 531–540, 2001.

[10] B. Hoff and R. Azuma, “Autocalibration of an electronic compass in an outdoor augmented reality system,” *Augmented Reality, 2000. (ISAR 2000). Proceedings. IEEE and ACM International Symposium on, Munich, Germany*, pp. 159–164, 2000.

[11] M. Karjalainen, *Kommunikaatioakustiikka*. "Laboratory of Acoustics and Audio Signal Processing, TKK", 1999.

[12] J. Merimaa and V. Pulkki, “Spatial Impulse Response Rendering I: Analysis and Synthesis,” *Journal of the Audio Engineering Society*, vol. 53, no. 12, pp. 1115–1127, 2005.

[13] C. Faller and J. Merimaa, “Source localization in complex listening situations: Selection of binaural cues based on interaural coherence,” *Journal of the Acoustical Society of America*, vol. 116, no. 5, pp. 3075–3089, 2004.

[14] J. Merimaa, *Analysis, Synthesis, and Perception of Spatial Sound - Binaural Auditory Modeling and Multichannel Loudspeaker Reproduction*. PhD thesis, Laboratory of Acoustics and Audio Signal Processing, Department of Electrical and Communications Engineering, Helsinki University of Technology, 2006.

[15] T. D. Rossing, R. F. Moore, and P. A. Wheeler, *Science of Sound*. Addison Wesley, 3 ed., 2002.

[16] K. A. J. Riederer, *HRTF analysis: Objective and subjective evaluation of measured head-related transfer functions*. PhD thesis, Laboratory of Acoustics and Audio Signal Processing, Department of Electrical and Communications Engineering, Helsinki University of Technology, Espoo, 2005.

[17] D. Hammershøi and H. Møller, “Sound transmission to and within the human ear canal,” *Journal of the Acoustical Society of America*, vol. 100, no. 1, pp. 408–427, 1996.

[18] J. Blauert, *Spatial Hearing - Revised Edition: The Psychophysics of Human Sound Localization*. The MIT Press, 1997.

[19] N. Sawhney, “Contextual awareness, messaging and communication in nomadic audio environments,” Master’s thesis, Media Arts and Sciences, MIT Media Lab, 1998.

[20] N. Sawhney and C. Schmandt, “Nomadic radio: Speech and audio interaction for contextual messaging in nomadic environments,” *ACM Transactions on Computer-Human Interaction*, vol. 7, pp. 353–383, 2000.

[21] J. Borwick, *Loudspeaker and Headphone Handbook*. Focal Press, 2001.

[22] "Philips dealer support. philips shn2500 productpicture," June 2008. http://www.dealersupport.se/imagebank/images/shn2500.jpg.

[23] W. R. Sherman and A. B. Craig, *Understanding Virtual Reality: Interface, Application, and Design*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002.

[24] "HowStuffWorks "Virtual Reality Tracking Systems" by Jonathan Strickland," Dec. 2008. http://electronics.howstuffworks.com/VR-gear6.htm.

[25] G. Welch and E. Foxlin, "Motion tracking: No silver bullet, but a respectable arsenal," *IEEE Computer Graphics and Applications*, vol. 22, no. 6, pp. 24–38, 2002.

[26] "SensAble Technologies," Dec. 2008. http://www.sensable.com.

[27] K. W. Arthur, K. S. Booth, and C. Ware, "Evaluating 3d task performance for fish tank virtual worlds," *ACM Transactions on Information Systems*, vol. 11, no. 3, pp. 239–265, 1993.

[28] "Logitech – 3D Mouse & Head Tracker Techical Reference Manuel," Dec. 2008. http://www.inition.co.uk/inition/pdf/mocap_logitech_headtracker.pdf.

[29] G. Welch, G. Bishop, L. Vicci, S. Brumback, and K. Keller, "High-performance wide-area optical tracking: The hiball tracking system," *Presence: Teleoperators and Virtual Environments*, vol. 10, pp. 1–21, February 2001.

[30] "The aerospace corporation, gps primer," June 2008. http://www.aero.org/education/primers/gps/index.html.

[31] "Shake sk6 user manual," June 2008. http://www.dcs.gla.ac.uk/research/shake/shakedocs.html.

[32] E. Foxlin, "Inertial head-tracker sensor fusion by a complementary separate-bias kalman filter," *Virtual Reality Annual International Symposium, 1996., Proceedings of the IEEE 1996, Santa Clara, California*, pp. 185–194, 267, Mar-3 Apr 1996.

[33] J. Williamson, R. Murray-Smith, and S. Hughes, "Shoogle: excitatory multimodal interaction on mobile devices," in *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, (New York, NY, USA), pp. 121–124, ACM, 2007.

[34] "FLAC, Free Lossless Audio Codec," Nov. 2008. http://flac.sourceforge.net/.

[35] EBU - Technical Specification 3285, "BWF - A format for audio data files in broadcasting," tech. rep., European Broadcasting Union, 2001.

[36] "The Sonic Spot: Wave File Format," Nov. 2008. http://www.sonicspot.com/guide/wavefiles.html.

[37] "GPS Chipsets and Receivers Compared," Dec. 2008. http://www.gpspassion.com/fr/articles.asp?id=143.

[38] M. Puckette, *The theory and technique of electronic music*. World Scientific Press, 2007.

[39] "Csound," June 2008. http://www.csounds.com.

[40] "Supercollider," June 2008. http://supercollider.sourceforge.net/.

[41] "Pure data," June 2008. http://puredata.info/.

[42] M. Puckette, "Pure data: another integrated computer music environment," in *Proceedings, the Second Intercollege Computer Music Concerts, Tachikawa, Japan*, pp. 37–41, 1996.

[43] "Sndobj," November 2008. http://music.nuim.ie/musictec/SndObj/main.html.

[44] "Codepedia, The GPRMC Sentence," Nov. 2008. http://www.codepedia.com/1/The+GPRMC+Sentence.

[45] "A PureData external to access and control the Wiimote under Windows," Nov. 2008. http://wiisense.googlecode.com/.

[46] "Nintendo Wii Homepage," Nov. 2008. http://wii.com/.

[47] M. Wozniewski, "Wiimote external for pure data," 2008. http://mikewoz.om/index.php?page=pd-stuff.

[48] Microsoft, "Microsoft Software Developers Kit, Multimedia Standards Update, rev 3.0, April 15," 1994.

[49] M. Peltola, T. Lokki, and L. Savioja, "Augmented reality audio for location-based games," in *AES 35th International Conference, London, United Kingdom*, February 2009. Accepted for publication.

[50] "Flext, C++ programming layer for cross-platform development of PD and Max/MSP externals," Dec. 2008. http://puredata.info/Members/thomas/flext-intro.pdf.

[51] "Boost, Free peer-reviewed portable C++ source libraries," Dec. 2008. http://www.boost.org/doc/libs/1_36_0/index.html.

[52] M. Tikander, M. Karjalainen, and V. Riikonen, "An augmented reality audio headset," in *The International Conference on Digital Audio Effects*, September 2008.

[53] "cfis; Google Maps Deconstructed," Dec. 2008. http://cfis.savagexi.com/2006/05/03/google-maps-deconstructed.

[54] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.