

**Remote Collaborative Real-Time Multimedia Experience over  
the Future Internet**

**ROMEEO**

**Grant Agreement Number: 287896**

**D3.6**

**Final Report on 3D Audio - Video  
Rendering Algorithms**

<b>Document description</b>	
Name of document	Final Report on 3D Audio-Video Rendering Algorithms
Abstract	This document defines ROMEEO audio-visual rendering algorithms developed for the different terminals
Document identifier	D3.6
Document class	Deliverable
Version	1.0
Author(s)	D.Doyen, S. Thiébaud, C. Thébault (TEC), C. Kim, I. Elfitri, H. Lim (US), M. Urban, M. Weitnauer, C. Hartmann, M. Meier (IRT), X. Shi (MULSYS), P. Apostolov (MMS)
QAT team	Nicolas Tizon (VITEC), Orkun Tanik (ARCELIK), Tasos Dagiuklas (UP)
Date of creation	27-Jul-2013
Date of last modification	25-Sep-2013
Status	Final
Destination	European Commission
WP number	WP3
Dissemination Level	Public
Deliverable Nature	Report

## TABLE OF CONTENTS

<b>TABLE OF CONTENTS .....</b>	<b>3</b>
<b>LIST OF FIGURES.....</b>	<b>5</b>
<b>1 INTRODUCTION.....</b>	<b>7</b>
<b>1.1 Purpose of the Document .....</b>	<b>7</b>
<b>1.2 Scope of the Work .....</b>	<b>7</b>
<b>1.3 Objectives and Achievements.....</b>	<b>7</b>
<b>1.4 Structure of the Document .....</b>	<b>7</b>
<b>2 AUDIO RENDERING .....</b>	<b>8</b>
<b>2.1 Introduction .....</b>	<b>8</b>
<b>2.2 Mobile Terminal.....</b>	<b>8</b>
2.2.1 Binaural technology.....	8
2.2.2 Binaural Room Synthesis .....	9
2.2.3 BRS bottlenecks on mobile device.....	10
2.2.4 Status of current work.....	11
<b>2.3 Portable Terminal.....</b>	<b>19</b>
2.3.1 Object-based Rendering.....	19
2.3.2 Status of current work.....	21
<b>2.4 Fixed Terminal.....</b>	<b>21</b>
2.4.1 Spatial Audio Rendering Principles .....	21
2.4.1.1 Conventional Multi-channel Audio .....	21
2.4.1.2 Wave-Field-Synthesis (WFS) and Simplified WFS.....	22
2.4.2 Audio Rendering Implementation Framework.....	25
2.4.2.1 Conventional 5.1-channel rendering system.....	25
2.4.2.2 Simplified WFS system .....	25
2.4.3 Audio Rendering System Settings for the Fixed Terminal .....	27
2.4.3.1 5.1-channel rendering .....	27
2.4.3.2 Simplified WFS system setup .....	28
2.4.4 Virtual Source Localisation based on WFS.....	30
2.4.4.1 Focused source.....	30
2.4.4.2 Experimental validation of the source localisation .....	30
<b>3 VIDEO RENDERING.....</b>	<b>34</b>
<b>3.1 Introduction .....</b>	<b>34</b>
<b>3.2 Fixed Terminal.....</b>	<b>34</b>
3.2.1 Experience acquired during the MUSCADE project (range of disparity) .....	34
3.2.2 Range of disparity defined for ROMEO.....	35
3.2.3 Depth Image Based Rendering Algorithm improvements .....	37
3.2.3.1 Overview .....	37
3.2.3.2 Disparity map interpolation.....	37
3.2.3.3 Occlusions identification.....	38
3.2.3.4 Occlusion border management.....	38
3.2.3.5 De-aliasing .....	40
3.2.3.6 Interpolated view rendering.....	41
3.2.4 Algorithm adaptation for parallelization .....	41
3.2.4.1 CUDA programming model.....	41
3.2.4.2 View interpolation CUDA kernel.....	42
3.2.5 Implementation into a GP GPU system.....	44
<b>3.3 Portable Terminal.....</b>	<b>48</b>
3.3.1 Video rendering on portable terminal .....	48
<b>3.4 Mobile Terminal.....</b>	<b>49</b>
3.4.1.1 Introduction.....	49
3.4.1.2 Stereo rendering in Mobile Terminal Device.....	49
3.4.1.3 OMAP Display Subsystem Capabilities .....	50

3.4.1.4	Stereo rendering pipeline implementation .....	50
<b>4</b>	<b>CONCLUSIONS .....</b>	<b>52</b>
<b>5</b>	<b>REFERENCES .....</b>	<b>53</b>
<b>APPENDIX A: GLOSSARY OF ABBREVIATIONS.....</b>		<b>55</b>

## LIST OF FIGURES

Figure 1: Typical BRIR for the left ear .....	9
Figure 2: Block diagram of the overlap-discard partitioned convolution algorithm .....	10
Figure 3: Components of the sound field over time [10].....	11
Figure 4: Coordinate systems relevant for mobile head-tracking applications .....	13
Figure 5: Schematic construction of the mobile head-tracking device .....	15
Figure 6: Artificial head with IMU head-tracking component mounted on a current headphone .....	16
Figure 7: Sensor fusion process based on DCM complementary filtering.....	17
Figure 8: Measurement diagram of the gyros output angle with and without drift compensation applied.	19
Figure 9: Basic building blocks to process and render object-based audio.....	20
Figure 10: First and second order reflection pattern generated by a sound source. Direct path (black), 1st order (blue) and 2nd order (yellow) .....	21
Figure 11: Loudspeaker layout for 5.1-channel system according to ITU-R BS. 775 standard, with a typical location for the LFE channel .....	22
Figure 12: Problem sketch .....	23
Figure 13: Sound field propagation generated by the algorithm.....	23
Figure 14: Sound field produced by a typical WFS system .....	24
Figure 15: Sound field produced by a simplified system .....	24
Figure 16: Typical multichannel audio rendering framework using external rendering hardware and ASIO protocol .....	25
Figure 17: Block diagram of main software algorithm .....	26
Figure 18: Block diagram of main hardware, <sup>‡</sup> Alesis Digital Audio Tape protocol, <sup>†</sup> Multi-Channel Audio Digital Interface, <sup>§</sup> Hammerfall DSP.....	27
Figure 19: Loudspeakers available for 5.1-channel rendering in ROMEEO .....	27
Figure 20: Multichannel audio rendering hardware in ROMEEO.....	28
Figure 21: Arrangement of WFS setup at I-lab.....	28
Figure 22: Loudspeaker panel .....	29
Figure 23: General configuration of the simplified WFS system.....	29
Figure 24: A focused virtual source at (2.8, 1), 500 Hz .....	30
Figure 25: Measuring setup including a) angle meter, b) B-format microphone, and c) a pair of half-inch free-field microphones.....	31
Figure 26: Virtual source positions on the horizontal cross-section of a space, +: originally intended, o: measured at different positions on the reference line. Points 'A' and 'B' denote positions where the monitoring microphones are mounted.....	32

Figure 27: Direction of the virtual source.....	33
Figure 28: Overall ROMEO demonstration scenario .....	34
Figure 29: MUSCADE camera and relative view interpolation positioning .....	35
Figure 30: ROMEO camera and relative view interpolation positioning .....	36
Figure 31: ROMEO content: ACTOR sequence.....	36
Figure 32: Disparity map of the ACTOR sequence .....	37
Figure 33: Projection of the left disparity map .....	38
Figure 34: Interpolation scheme without border management .....	39
Figure 35: Interpolation scheme with border management .....	39
Figure 36: Interpolation results: without (top) and with (bottom) border management .....	40
Figure 37: Disparity map and corresponding interpolated video without de-aliasing.....	40
Figure 38: Disparity map and corresponding interpolated video with de-aliasing.....	40
Figure 39: Interpolation results: without (top) and with (bottom) de-aliasing .....	41
Figure 40: Disparity projection per thread .....	43
Figure 41: Projected disparity by all the threads .....	43
Figure 42: Projected disparity after filtering.....	44
Figure 43: Functional description .....	46
Figure 44: Concurrent execution .....	48
Figure 45: View interpolation for portable terminal .....	49
Figure 46: Stereo rendering pipeline example .....	50
Figure 47: Stereo rendering pipeline operation .....	51

## **1 INTRODUCTION**

### **1.1 Purpose of the Document**

The purpose of document D3.6 is to present a status of the work done in the field of audio-visual rendering within the ROMEEO project. The document D3.3: “Interim Report on 3D Audio – Video Rendering Algorithms” has provided some technical descriptions of the work done in the field of the task 3.4 of the project during the first part of the project. The document D3.6 is the final version of the D3.3. It uses the interim version as a starting point and it is complemented by additional and complementary information. The structure of the document remains as the D3.3 one.

### **1.2 Scope of the Work**

This deliverable reports the work carried out towards the development of audio/video algorithms to improve the user experience for fixed, portable and mobile terminals. The different terminals are receiving audio and video content through the hybrid ROMEEO transmission chain. This is the purpose of the WP3 to work on the exploitation of these contents using respective capabilities of terminals.

### **1.3 Objectives and Achievements**

The task 3.4 is dedicated to review and implement appropriate audio and video rendering algorithms taking into account the complexity and buffering for real-time processing and displays/loudspeaker systems. Spatial audio rendering techniques will be based on Wave Field Synthesis WFS combined with Ambisonics for fixed users and binaural spatial audio rendering will be considered for the mobile users. For the 3D video, multi-view video content rendering is addressed by interpolating the in-between viewpoints.

### **1.4 Structure of the Document**

This deliverable is structured as follows: Section 2 presents the audio rendering processing and Section 3 the video rendering part for each terminal. Section 4 concludes this deliverable.

## 2 AUDIO RENDERING

### 2.1 Introduction

This section describes the principles behind the various audio rendering techniques used for the project. Moreover it provides an overview about the audio rendering work for the different terminals used in ROMEEO. The algorithms and principles (state-of-the-art and beyond) are described as well as the current status of the implementation.

### 2.2 Mobile Terminal

The so-called mobile terminal of the ROMEEO project targets mobile platforms such as smartphones and tablet devices. In the past years, these kind of devices evolved to constant companions of modern individuals. Due to the fact that the playback of audio with these devices is mainly via headphones, the audio rendering approach of ROMEEO for this kind of terminal will also use the benefits of headphone sound reproduction.

#### 2.2.1 Binaural technology

Binaural audio is a spatial audio technology, which is based on the principles of natural human hearing. Due to having two ears, humans can localise sound direction. Sound waves coming from a source reach the two ears at different times because they travel different distances. Shadowing effects by the human head also cause the sound pressure at each ear to have different levels. This means that human hearing system can simply detect the position of a sound source by processing mainly the inter-aural time differences (ITDs) and inter-aural level differences (ILDs) [1]. Shadowing by human head, reflection and diffraction by surrounding environments can also cause differences in the frequency spectrum of the signals received by the left and the right ear, which are used by the human auditory system to further improve the localisation of sound sources.

However, there are some positions such as in front or in the back of human head, which would produce identical ITDs and ILDs. If a sound comes from these locations, humans will get confused and cannot precisely determine the source position. This phenomenon is known as cone of confusion [3] and can be solved by head movements [4]. Using the changes of the ILDs and ITDs, which are referred to as dynamic binaural cues, the human auditory system is able to assign the correct position to the sound source. Another situation that makes human brain fail to detect proper positions of sound sources is when there are two or more sound sources, located in different locations, emitting the same sound with small time delays. These sounds will be perceived as a single sound source localised at the position of the sound source emitting the sound first. Such a situation is called the precedence effect [5].

The human head's size, the unusual shape of pinna, as well as the reflections off the shoulder and body will also affect the ability to localise a sound source. All of these cues can be represented as a head-related transfer function (HRTF) [2]. This function is a frequency response of the human outer ear. As HRTFs are also depending on the position of a sound source, they are typically measured in an anechoic chamber for every needed source position and stored in a database. If the transfer functions also include the directivity and characteristics of a sound source (e.g. a real loudspeaker), the acoustic properties of a room and the HRTF of a human head, these are commonly called Binaural Room Transfer Functions (BRTF). Converted into the time domain, impulse responses are generated. These are called Binaural Room Impulse Responses (BRIR) [2]. Figure 1 shows a typical BRIR of a loudspeaker within an acoustically optimized monitoring room.

There are basically two ways to acquire BRIRs. The first option is to use real measurements of real sound sources at various positions and for all possible head rotations. This so-called data-based approach generates high quality BRIRs which leads to a very good spatial audio quality. The downside of this approach is that only source positions that have actually been measured can be used during playback. Furthermore, the measurement process is relatively

time intensive, but only has to be done once. This approach is therefore superior for static playback situations, for example to simulate a 5.1 playback system on a mobile device.

The second option is the so-called object-based approach. For this, the transfer paths of a given sound source in a virtual room are calculated depending on the position and orientation of the listener. Based on these transfer paths, BRIRs can be synthesized for anechoic conditions using a HRTF database and directivity patterns of the sound sources. Furthermore, virtual listening rooms can be included in the simulation process to provide a realistic room impression. This approach is very flexible as sound sources can be moved freely and dynamically, but comes with a highly increased computational complexity compared to the data-based approach and is therefore not able to produce the same *high* quality BRIRs with reasonable resources.

Thus it is necessary to choose the right approach based on the requirements and system constraints of the targeted application. For the mobile system, a data-based approach based on *Binaural Room Synthesis* was chosen. For object-based audio playback on the portable device, obviously only the object-based approach is possible.

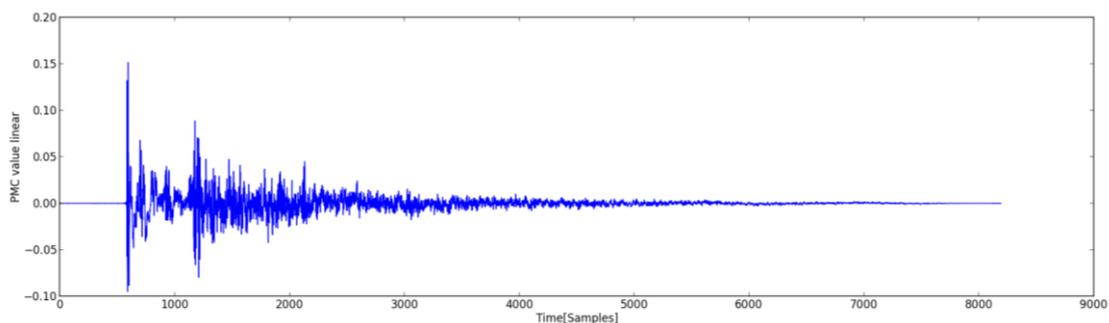


Figure 1: Typical BRIR for the left ear

### 2.2.2 Binaural Room Synthesis

The convolution of channel based audio signals with BRIRs generates a binaural signal. If this binaural signal is played back through headphones, it gives the listener a spatial impression, comparable to the one in this room at the recording position. This method is called binaural room synthesis (BRS) [6] and exists as some rare implementations for professional production, including a high-quality implementation of one of the ROMEEO partners. By employing a head-tracking solution, the binaural signal can be adapted to the rotation of the user's head. If this adaptation is conducted in real-time, the perception of the user is massively enhanced by dynamic cues. The localization of sound sources is much easier and front-back confusions are mostly avoided.

Since the whole adaption process (including head-tracking, filter selection and convolution) has to be conducted within less than 85ms [7] and even the duration of a typical BRIR is longer than this value (e.g. the duration of the BRIR in Figure 1 is 169ms), the signal processing has to be realized with the so-called partitioned convolution, which is also known as fast convolution. A simple block diagram of the fast convolution using the popular overlap-discard method is depicted in Figure 2. The basic idea is to split both the continuous input signal as well as the BRIR into smaller partitions. Then each input signal block is convolved with each filter partition. The results of these operations can then be combined into the desired output signal. The benefit of using this method is that the latency is reduced to the length of the input block size. Furthermore, as the convolution is carried out in the frequency domain, the computational complexity is reduced significantly compared to time domain convolution.

### Partitioned Convolution (overlap-discard)

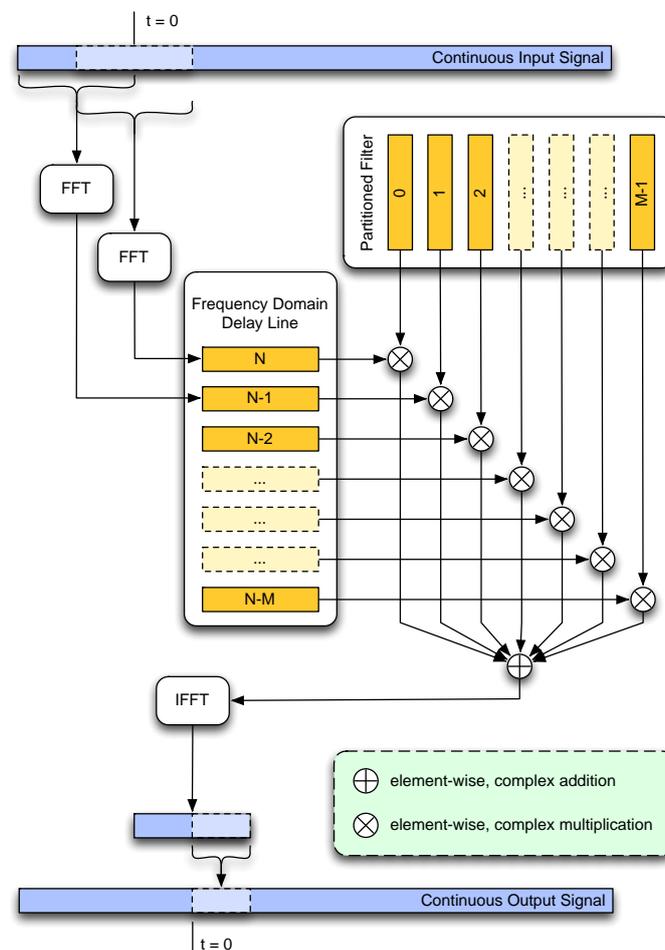


Figure 2: Block diagram of the overlap-discard partitioned convolution algorithm

For mobile applications, headphones are an interesting option to reproduce multichannel audio signals by transcoding multichannel audio into binaural audio. Using a convolution with BRIRs for the purpose of generating a binaural signal on mobile devices - especially not with a head-tracking solution - was not commercially available at the beginning of ROMEEO.

#### 2.2.3 BRS bottlenecks on mobile device

There are several major difficulties that occur, if a BRS system shall be implemented on a mobile platform.

The most obvious difficulty is the possibility to track the movements of the user with a head-tracking device suitable for mobile and outdoor applications. Therefore a small and lightweight tracking system, which provides resilient measured data even under highly diverse environmental conditions, has to be implemented. The most important requirements for the usage of head tracking with mobile MTB (Motion Tracked Binaural) devices are:

- A tracking technology, which does not need a stationary reference point, a moving target could be measured relative to
- An update rate higher than 20Hz for smooth repositioning of audio objects and therefore an appropriate overall output latency < 50ms.
- No tracking inaccuracies > 1°
- A tracking resolution <= 1° on sensing of the yaw rotation angle

- An operation time of several hours (battery drain)
- Compact, lightweight and easy to use design
- Resilient against environmental influences like mechanic impacts as well as electric and magnetic interfering fields
- Reproducible at a reasonable price

Other difficulties are the limited computational power and RAM (Random-access Memory) capacity of state-of-the-art mobile devices. The computational demand of a BRS implementation for a 5.1 virtualization is still pretty high. Hence, the signal processing itself has to be enhanced and the BRIR have to be simplified. An enhancement of the signal processing can be reached by separating the impulse response into two parts. The first part contains only the direct sound of the impulse response and the second part only the reverberation, as it is schematically shown in Figure 3. To save storage space, RAM and computational power, the same reverberation part can be used for all impulse responses. A simplification of the used BRIR could be reached by shortening the impulse responses to a minimum number of samples.

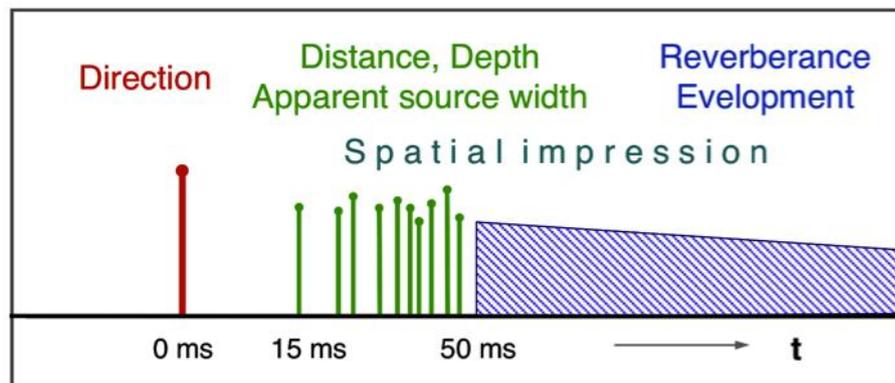


Figure 3: Components of the sound field over time [15]

## 2.2.4 Status of current work

### ARM implementation

The first release version of the BRS embedded software was built on a Pandaboard-ES, running a minimal Ubuntu-Linux, version 12.04. The hardware as well as the software meet the requirements of low latency processing. The processing power of the two ARM cores is sufficient for real-time reproduction of the audio stream on headphones. The software performs different tasks at the same time. The head position is located in an interval of at least every 20ms at a precision of 1°. The multichannel audio signal is convolved with the selected Binaural Room Impulse Response (BRIR) and rendered out on stereo output for the current head position. The latency for this operation does not rise above 85ms. Nevertheless, there is still room left for an even better performance. Optimizations have only been done on the software release, so far.

To conduct investigations regarding an optimization of the BRIR, a software tool has been implemented which allows creating and editing of BRIR databases for IRT's BRS implementation. With the developed BRS filter tool, the BRIR database could be optimized for a mobile device. By informal investigations the single BRIR could be shortened in such a way that the listening impression was not affected. This shortened version allows a reduction of CPU load by about 15% on a dual core arm processor, compared to the long version of the same BRIR. The BRS-client software was running stable with a consumption of 60%-70% of the CPU power.

Further optimizations could probably be made by running a specialized kernel instead of the very generic one provided by Ubuntu. Also the use of the newest convolution libraries by IRT could make an additional benefit. The current version of the BRS embedded software depends on the Jack Audio Connection Kit (JACK) – a system for handling real-time, low latency audio. Future releases could enable a direct communication between the BRS-client and ALSA (Advanced Linux Sound Architecture).

## **Mobile head-tracking device**

### *General requirements*

Since head-tracking is a crucial factor in terms of synthesis quality of BRS-devices [8][9] a solution for the mobile application had to be found within the scope of ROMEO. Head-tracking for virtual reality utilization is a sufficiently studied area of research, but nonetheless there is no commercially available and cost-effective solution on the market, yet, which meets all the requirements of mobile binaural playback. Therefore existing technologies of orientation sensing had been investigated, combined with the help of reasonable software algorithms and implemented into a portable hardware device.

In this respect particularly the aforementioned demands of mobile application have to be considered. While time of flight (TOF) based sensors are the head-tracking standard for fixed BRS installations, they have some serious drawbacks in terms of mobile usage. These sensors rely on the measure of the propagation time between a moving target in relation to a fixed reference and usually use media like infrared or ultrasonic pulsed signals. Commonly applied TOF-devices involve three or more emitters on the target side, as well as several receivers on the reference. The spatial position of the target in reference to the receiver location is subsequently determined by triangulation. TOF based systems can be considered as the most accurate tracking systems, but do compulsory need a fixed reference point, which is a difficult to solve problem in regard to mobile usage. Compared with other ordinary head-tracking applications, not the head movement in respect to the earth coordinate system, but the head movement in reference to the listeners torso coordinate system matters for BRS-playback, as shown in Figure 4. For that reason there might be a way to use TOF based trackers within a mobile context by attaching the receivers to the listeners torso, while sending out propagating signals from the listeners head. However, this approach is highly sensitive to diffraction and absorbance caused by human body parts, in case ultra sonic emitters are used. Infrared emitters as opposed to this, need a free line of sight in order to work correctly. That is why TOF-sensors do not seem to be convenient as a primary tracking-technology, but could eventually be used as a support system combined with other tracking-methods.

The currently very popular spatial scan technologies, which also include camera based videometry-tracking, are strictly inapplicable to mobile use at the present time. This is mainly due to performance requirements, which are hard to fulfil on today's mobile devices, as well as the fact, that the need of a continuous presence of a camera in front of the users head would inadequately diminish the benefits of a mobile spatial listening experience.

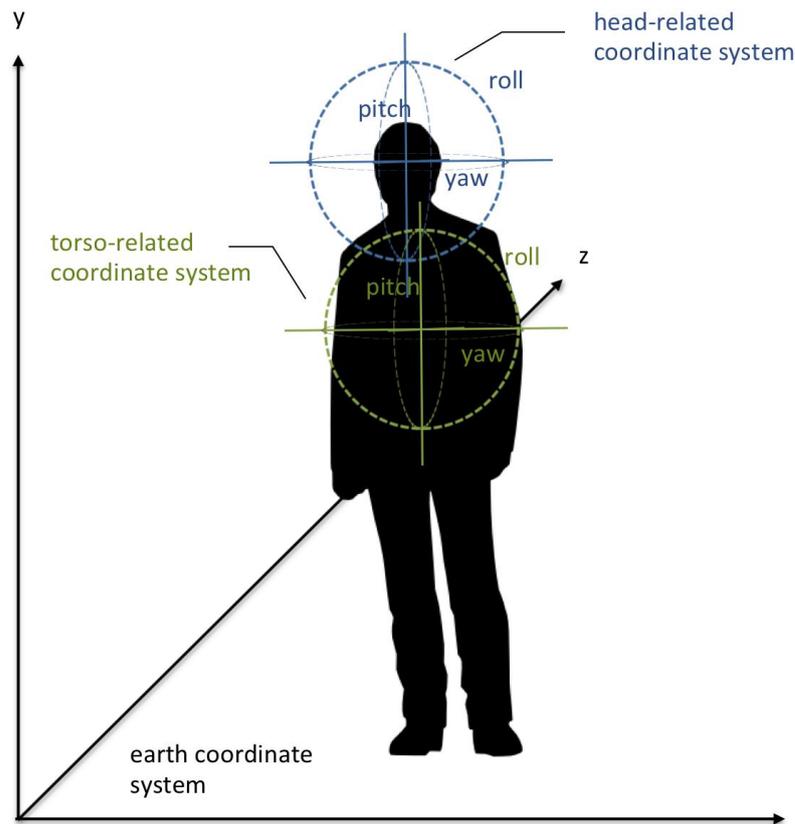


Figure 4: Coordinate systems relevant for mobile head-tracking applications

Pre-investigations showed, that within the scope of ROMEEO only a few principles of operation seem to be qualified for the realisation of a mobile head-tracking device. Out of the six main tracking principles inertial sensing and direct field sensing showed the greatest potential to be used in this field of application.

The principle of inertial sensing is based on the attempt to conserve either a given axis of rotation or the position of a weight. It can be subdivided into two basic technologies, which are preferably used in combination: The mechanical gyroscope, a system based on the principle of conservation of the angular momentum and the accelerometer, which measures the linear acceleration of an object to which it is attached. Usually not only one sensor of each type is used for detecting the position, but at least one or more sensors per dimension are coupled in order to form a so-called inertial measurement unit (IMU). Hence, an IMU commonly provides three degrees of freedom (DOF) or more and can sense three-dimensional position values. Gyroscopes and accelerometer are both lightweight and can be produced very cost-effective. Most convenient however in the context of mobile tracking is the fact, that inertial sensors don't need an external reference to work. As previously mentioned, the mobile application of BRS needs the head positioning angles transformed into the torso coordinate system (Figure 4), since the BRS renderer is not supposed to react on torso movements of the listener. Hence, the use of IMU sensors requires a second IMU device in order to gain the positional information of the listeners body. However a downside of gyroscopes is, that they are only able to provide relative measurements. As a consequence, gyroscopes suffer from accumulated numerical errors, caused by unavoidable measuring inaccuracies, nonlinearities and noise, which lead to a constant drift of the measured position. For this reason it is advisable to perform a periodic re-calibration of this sensor type, which will insure a higher accuracy over time.

An automatic re-calibration can for instance be done with a magnetometer, belonging to the category of direct field sensing devices. With three magnetometers the orientation of an object with respect to the magnetic field of the earth can be measured. One problem with this

technology is, that the earth's electromagnetic field is inhomogeneous and yields angular errors in the orientation measurements. Therefore it's most beneficial to use this technology as an enhancement for the precision of inertial sensors within an IMU.

Another possibility is to use a head-tracking device entirely based on magnetic field sensing, where a magnetic coupling between an emitting coil and a receiving coil is used to measure the relative distance and orientation between these parts. A problem with this type of trackers is the distortion of the magnetic field by ferro-magnetic objects and other sources of electromagnetic fields.

The pre-investigation showed, that there is no single type of orientation sensor, which meets all the requirements of a mobile binaural playback device. Therefore a hybrid sensor array consisting of three orthogonal arranged gyroscopes, accelerometers and magnetometers with altogether 9 degrees of freedom was chosen for this case of application.

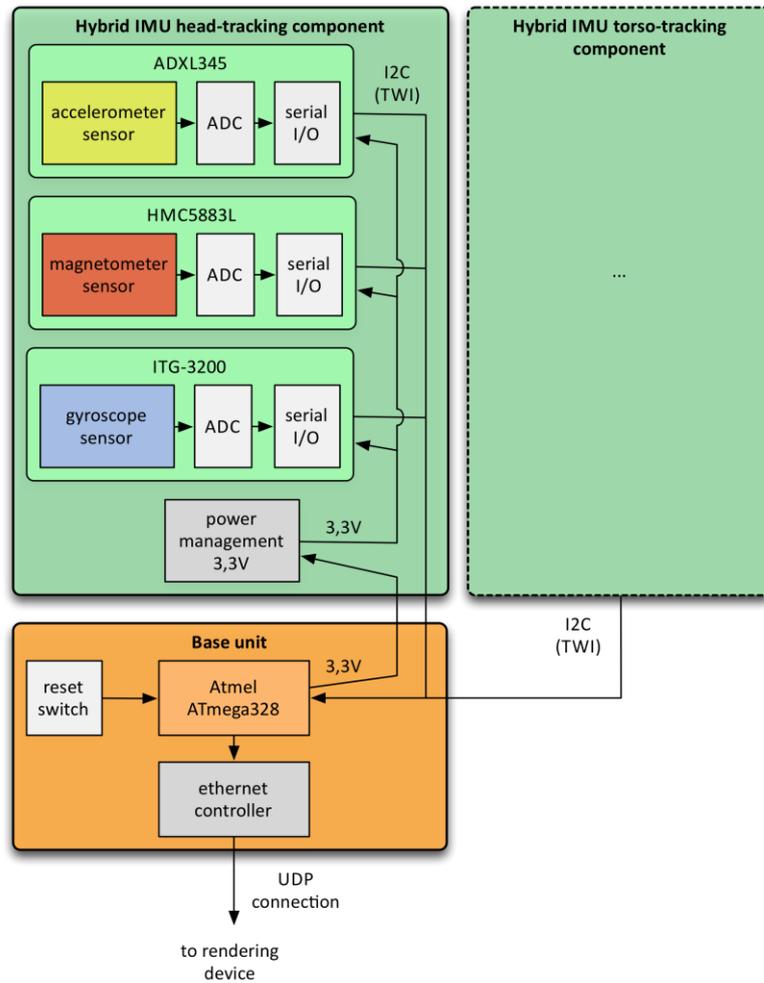
### *System design*

The overall mobile head-tracking device, as schematically shown in Figure 5, consists of three basic system components: The IMU head-tracking component, the IMU torso tracking component and the base unit, containing an Atmel ATmega 328 microcontroller, as well as an embedded ethernet interface. Each IMU component is equipped with an InvenSense ITG-3200 MEMS gyroscope, an ADXL345 accelerometer by Analog Devices and a Honeywell HMC5883L magnetometer.

All sensors support the measurement on three orthogonal axes (roll, pitch and yaw) and have built-in ADCs and serial interfaces available. The connection from the IMU to the microcontroller is respectively implemented via cable based I<sup>2</sup>C bus communication. As a result all sensors are independently addressable. The Atmel microcontroller is used for all kinds of logic operations and has the following main tasks within the tracking device:

- Initializing and maintaining the communication with the sensors
- Performing the fusion of the sensor data
- Converting the output data into the defined bitstream communication protocol
- Controlling the UDP streaming output to the binaural rendering device

Besides the microcontroller, the base unit also contains a reset switch, which allows the listener to calibrate the head-trackers zero position for any desired head to torso constellation.



*Figure 5: Schematic construction of the mobile head-tracking device*

The IMU head-tracking device can be mechanically interlinked with conventional headphones (see Figure 6) and thus track any movement of the listeners head. However the IMU torso component is fixed with a belt strap around the users chest in order to pick up the torso motion. In a commercial adaption both parts could be considerably downsized.



Figure 6: Artificial head with IMU head-tracking component mounted on a current headphone

#### *Sensor fusion and calibration*

The idea behind combining multiple sensors within a hybrid tracking component is that properties of a particular sensor device can be used to overcome the limits and weak spots of another sensor. In this way, for instance, the data of a drift-free magnetometer can be applied to compensate for a gyros integrating drift. Hence, the mathematical combination of various sensors in the course of a so-called sensor fusion is the critical main process of every hybrid tracking system.

Commonly Kalman filters, a recursive filtering method for linear quadratic estimation, are used to perform a sensor fusion in the field of orientation tracking. A Kalman filter uses a physical model of the progression of a measurement value, as well as information on its noise and inaccuracy in order to generate an estimator, which is more precise than just a single measurement value [10][11]. Beyond that the Kalman filter is useful for combining data from several different noisy measurements, whereas particular sensor sources get weighted on the basis of their signal characteristics and therewith being utilized in the best way possible. For example unrealistic magnetometer values caused by electromagnetic distortion can be compensated with this process.

In the present case however, a complementary filtering, based on a direction cosine matrix (DCM) [12], is used as a more simple sensor fusion method. This is due to the fact, that a Kalman filter has high requirements in terms of computational power and cannot easily be implemented on a current microcontroller. The algorithm uses a proportional and integral feedback controller to compensate for the gyro drift. The gyro on the other hand is used as the primary orientation-sensing device. The principle is schematically illustrated in Figure 7.

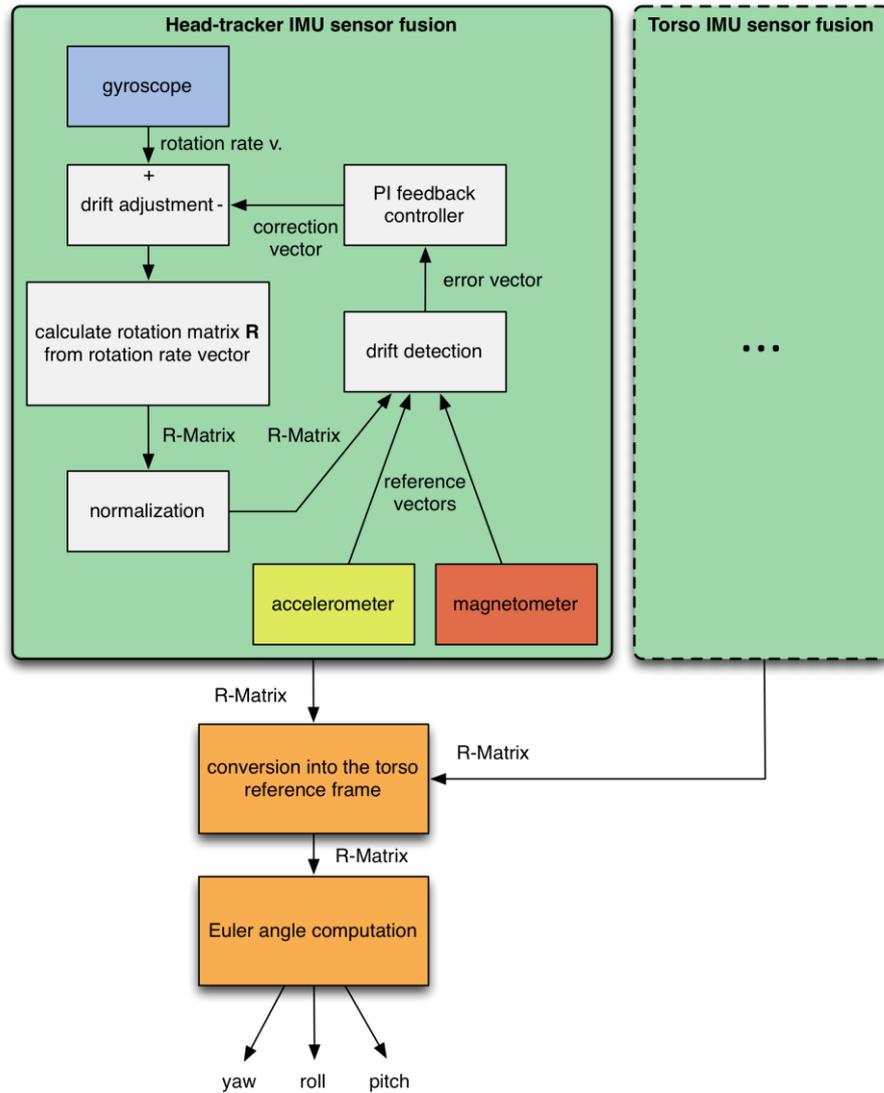


Figure 7: Sensor fusion process based on DCM complementary filtering

A series of succeeding rotation measurements from the gyro is not commutative and cannot simply be integrated. Their mathematical description of the kinematics forms a nonlinear differential equation, which can be assumed to be approximately linear for high measurement sampling frequencies. In this case we are able to numerically solve the direction cosine matrix  $\mathbf{R}$  by fast repeated matrix multiplications (Equation 1). Moreover the matrix is updated with the rotation rate vectors  $\omega_{gyro}(t)$ , readout from the gyroscope:

$$\omega(t) = \omega_{gyro}(t) + \omega_{correction}(t)$$

(Equation 1)

$$R(t + dt) = R(t) \begin{bmatrix} 1 & -d\theta_z & d\theta_y \\ d\theta_z & 1 & -d\theta_x \\ -d\theta_y & d\theta_x & 1 \end{bmatrix}$$

(Equation 2)

with

$$d\theta_x = \omega_x dt$$

$$d\theta_y = \omega_y dt$$

$$d\theta_z = \omega_z dt$$

In the next step the rotation matrix is normalized in a way, that every row forms a unit vector in order to comply with the orthogonality constraints. The rotation matrix now represents the relationship between the IMU coordinate system and the earth coordinate system. In a subsequent operation, the gyro drift is cancelled by calculating a correction vector  $\omega_{correction}(t)$ , whereby the gyro measurement signals get adjusted via the integral feedback controller (Equation 1). The integration of the feedback controller has the ancillary effect, that the gyro offset and thermal noise is widely corrected, as well. As a reference for the correction vector of the yaw angle the magnetometer data is used, while the roll / pitch angles are readjusted by the accelerometer signals. As seen in Figure 7 the rotation matrices of both of the IMU components are being merged into a common DCM, which represents the head movement in respect to the listeners torso. Finally the tracking result is calculated from the DCM by determination of the Euler angles (pitch, roll, yaw) and sent to the mobile rendering device.

Another essential point for improving the tracking results is to calibrate the sensors to its individual inaccuracies during their initialization, or even by continuous dynamic recalibration during current operation. As aforementioned this can partly be accomplished by sensor fusion, as well, but then will need an appropriate lead time and usually will not be as accurate as with previous calibration applied.

At the present head-tracking prototype merely the gyroscope sensor offset and the scaling of the accelerometers measurement range have been considered. Moreover the magnetometer needs more sophisticated calibration in order to cope with distortions of the earth's magnetic field. Exclusively distortions are compensated, which rotate with the IMU components coordinate system. Adjustment of errors, where the source of distortion is not bound to the sensor, requires complex adaptive algorithms and is not implemented, yet. Magnetic distortions divide into hard-iron and soft-iron effects and are both separately addressed by a specific calibration process. Hard-iron effects are caused by magnetic materials and generate an additive constant value to the magnetometer, independently of its position. Headphone speaker magnets for example are critical to induce hard-iron effects with regard to head-tracking devices. Soft-iron effects on the other hand are caused by ferromagnetic materials and generate distortions of the earth's magnetic field, which are dependent on the spatial position of the magnetometer.

#### *Status of work*

The head tracker implementation has been finished, tested and integrated with the binaural rendering device. Conclusive measurements have been performed to verify the quality attributes of the head-tracking device, such as the effectiveness of drift compensation. Figure 8 shows the gyros relative drift with and without drift compensation applied over a 10 minute period of time. It is clearly recognizable, that slight mechanical inaccuracies lead to a significant drift of more than 50° without the utilization of drift compensation, although the gyroscope remains in absolute position of rest during the measurement. On the other hand the gyros output remains perfectly stable when the drift detection and adjustment is applied. In order to illustrate the absolute attained accuracy and the effectiveness of the magnetic distortion correction further testing is required.

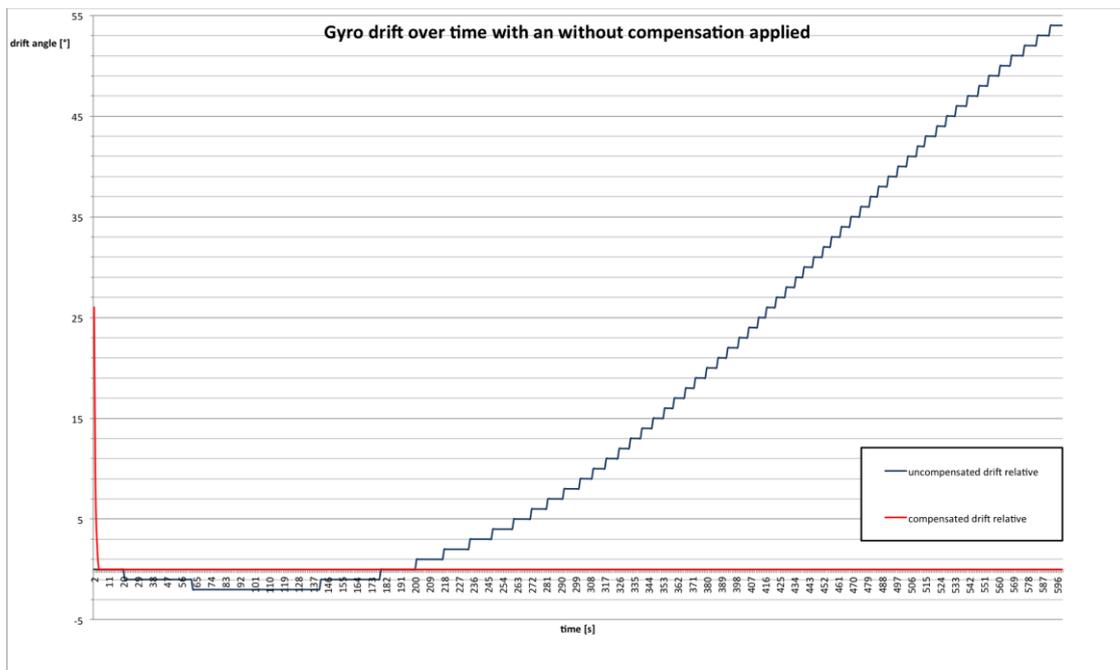


Figure 8: Measurement diagram of the gyros output angle with and without drift compensation applied

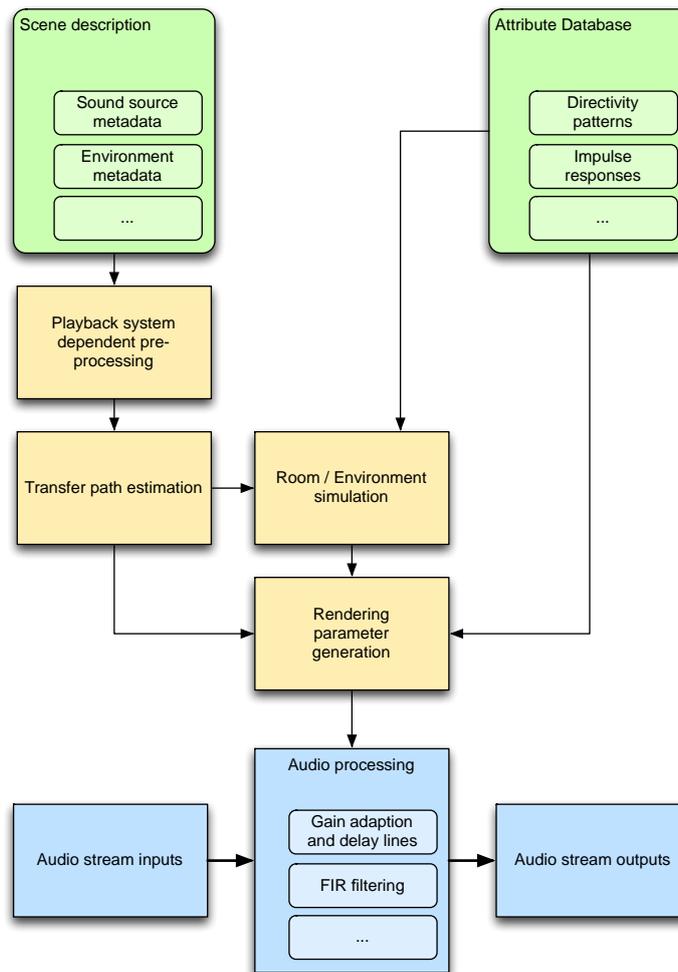
## 2.3 Portable Terminal

The portable terminal of the ROMEEO project targets devices such as laptops. Since audio with these devices is either played back via integrated / external loudspeakers or via headphones, the audio rendering approach of ROMEEO for this kind of terminal will take into account both playback solutions. As the usage of binaural signals with headphones offers a much better spatial impression, the mainly targeted playback solution will also use headphones.

### 2.3.1 Object-based Rendering

Considering the representation of a (possibly 3D) audio scenery for spatial audio rendering, a common approach in the research field is the use of individual audio objects instead of channel-based audio formats [8][14]. These audio objects carry additional metadata, such as position, gain or directivity. Furthermore, such an audio scene may be extended with a room model to enhance the spatial audio impression and capture the creative intent. Such a description of a spatial audio scene is in principle independent of the playback system used and very flexible. It is therefore possible to provide renderers for output formats or playback technologies, e.g. binaural audio or stereophony.

In Figure 9 the basic building blocks generally needed to process object-based audio are depicted. Using the object-based audio scene description the rendering parameters specific to the targeted playback system are generated for each sound source. These parameters include gain factors, delays or synthesized impulse responses. If necessary, the scene description can be pre-processed to adjust and optimize certain parameters within the scene to the capabilities of the targeted playback system. This process is repeated every time the audio scene changes, e.g. by a moving sound source and commits the generated parameters to the audio processing engine. This audio processing engine is computing the output signals in real-time based on the input signals and its parameter set.



*Figure 9: Basic building blocks to process and render object-based audio*

For binaural rendering, the rendering parameters mainly consist of artificial BRIRs, which are calculated based on the acoustic transfer paths of each audio source to the listener. The BRIRs can be synthesized by correctly modeling time delays, attenuation and a HRTF database. For stereophonic playback, basically two options exist. The first one is to use an identical process as for binaural rendering, but replacing the HRTF database with impulse responses of multi-channel main microphones, thus simulating a conventional microphone recording for multi-channel setups. The second option is to use techniques like Vector Based Amplitude Panning, thus mapping the audio source signals to the discrete output channels with different attenuation values and time delays [16].

To enhance the spatial impression of the audio playback, additional room simulation may be added to parameter generation process (Figure 10). Based on a geometric description of the room, two methods can be used to simulate virtual environments. The first is the so-called mirror-image method. The basic idea is that every reflection generated by a sound source and a surface can be substituted by introducing another virtual sound source. The second method is to use ray-tracing algorithms to calculate the reflection pattern generated by a sound source. As each of these two methods has its advantages and drawbacks, they are often used in combination as a hybrid approach [17].

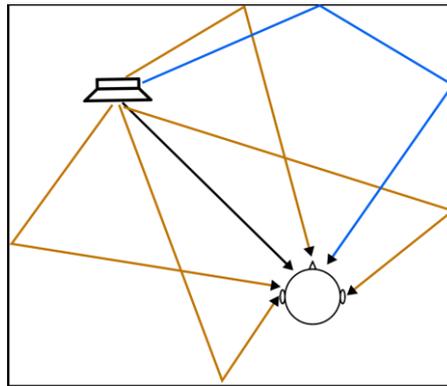


Figure 10: First and second order reflection pattern generated by a sound source. Direct path (black), 1st order (blue) and 2nd order (yellow)

### 2.3.2 Status of current work

All core functionalities and libraries have been combined into a fully functional rendering framework. This framework is capable of rendering dynamic, time-varying scenes with multiple objects for various output formats, including binaural and stereophonic 3D audio playback systems. It was used to build an object-based audio renderer, which is suitable for both server-side, offline as well as client-side real-time rendering. The framework itself is highly modular and therefore allows extending and modifying input, output and transformation operations easily. For this, the performance and time critical operations that have been implemented in C++ are scriptable and extendable using plugins in the python scripting language.

## 2.4 Fixed Terminal

For the fixed terminal, conventional multichannel audio rendering, particularly in 5.1 channels, is introduced as the main configuration. Additionally, Wave Field Synthesis (WFS) is also under investigation towards more practical use with reduced number of transducers.

### 2.4.1 Spatial Audio Rendering Principles

This section describes the principles and backgrounds for the conventional multichannel audio rendering system and the WFS system.

#### 2.4.1.1 Conventional Multi-channel Audio

Through the recent years, spatial audio reproduction systems have progressed in various configurations using multiple loudspeakers, in order to deliver to the users the perception of auditory scenes realistic in terms of sound source distribution, or enriched in terms of spaciousness or envelopment. The mechanism and cues for the spatial audio perception have been identified through a number of research works. It is widely known that the difference in the arrival time of sound between the two ears (Inter-aural Time Difference) and the difference in the amplitude of sound between the ears (Inter-aural Level or Intensity Difference) are the main cues for lateral sound source localisation. Additionally, the spectral difference of sound caused by the head, the structure of outer ear, and the torso acting as filtering elements are known as the main vertical localisation cue. The frequency content, loudness, and the amount of reverberation are known to affect the perception of the distance or depth. On the other hand, the similarity of the signals detected at the two ears is known to be related to the perception of apparent source width and envelopment of sound. The conventional multichannel surround audio reproduction systems have been developed generally to render these cues, mainly in the horizontal plane. The number of loudspeakers has been increasing through research and investigations since the days of 2-channel stereo, to overcome the

limitations in the reproducible auditory scene. Amongst the various configurations, the 5.1-channel system is the most widely accepted and standardised one, which will thus be introduced for the ROMEEO project.

The loudspeaker layout, seen from above, is specified in ITU-R BS.775 and shown in Figure 11, with a typical location for the Low-Frequency Effects (LFE) channel denoted by the '.1' component. The front loudspeakers maintain compatibility with two-channel stereo systems and reinforce front-oriented sound reproduction. The two “surround” loudspeakers contribute in creating sound images behind the listener and in providing lateral energy for good envelopment. The LFE channel is responsible for the low frequency content (below a maximum of 120 Hz) that cannot be handled by the other 5 loudspeakers. The EBU comments that this channel content should be considered optional and the overall reproduction should sound satisfactory even without this channel. The dotted rectangles in the frontal area describe the recommended screen placements.

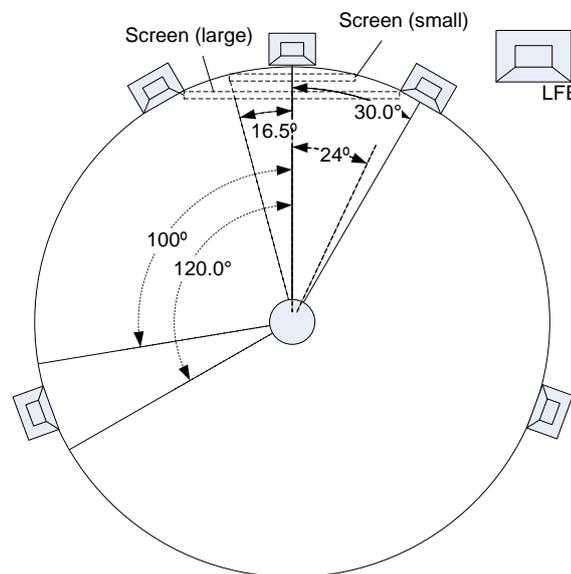


Figure 11: Loudspeaker layout for 5.1-channel system according to ITU-R BS. 775 standard, with a typical location for the LFE channel

#### 2.4.1.2 Wave-Field-Synthesis (WFS) and Simplified WFS

WFS is a practical application of the Huygens' principle [18]. The main feature of this methodology is its ability to automatically reproduce spatial sound equivalent to the original one generated at the source position within an entire domain by using secondary sound sources [19]. The secondary sources establish an active boundary, rendering a virtual lifelike soundscape in the given region. The boundary (control sources) can be composed with different combinations of acoustic monopole and dipole. However, the bulky dipole source can be omitted for simplification of the system as we are not interested in the sound field outside the domain[20]. Theoretically, Rayleigh I integral (Equation 3) allows us to implement the WFS system with a series of acoustic monopole sources [21].

$$P(\vec{r}, \omega) = \rho_0 c \frac{jk}{2\pi} \iint_S (V_n(\vec{r}_s, \omega) \frac{e^{-jk|\vec{r}-\vec{r}_s|}}{|\vec{r}-\vec{r}_s|}) dS,$$

(Equation 3)

where  $\rho_0$  denotes the air density,  $V_n$  the normal component of the particle velocity,  $k$  is a wave number,  $\omega$  is the angular frequency of a wave, and  $c$  is the speed of sound.  $\vec{r}$  denotes a position vector of a listening position inside the domain surrounded by an arbitrary surface  $S$ .

$\vec{r}_S$  is a position vector on the surface  $S$ .  $\varphi$  denotes the relative angle between the vector from a position on  $S$  to the listening position  $L$  and the normal vector  $\vec{n}$  in Figure 12.

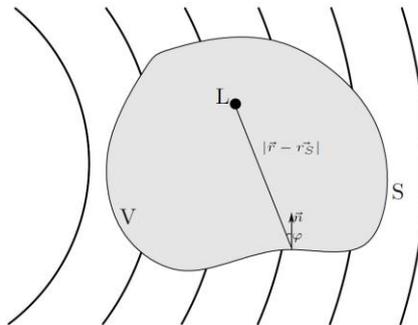


Figure 12: Problem sketch

In the equation a continuous distribution of secondary sources has been supposed [22]. However, no acoustic line source controllable precisely in detail for WFS has been discovered yet. In practical realization, the secondary source can be implemented only with arrays of number of discrete loudspeakers. The practical limitation requires the equations described in discrete form. Some simulation models in the discrete form have been developed at the I-Lab for effective evaluation of the algorithm in various given environments. A typical example of them is illustrated in Figure 13 where a non-focused virtual acoustic source is supposed to be at  $x=0$  and  $y=-2$ , i.e. outside a listening room, and a listener is situated in the middle of the room.

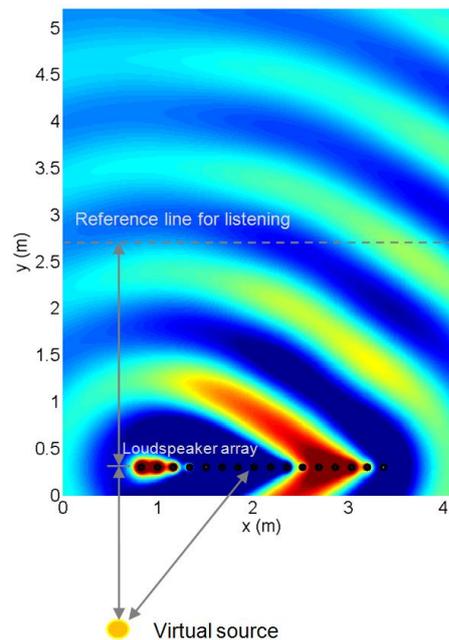


Figure 13: Sound field propagation generated by the algorithm

According to further requirements, the function of the WFS system can be enhanced to enable to reproduce sound fields generated by moving sources and focused sources, i.e. sources inside the listening area in Figure 14.

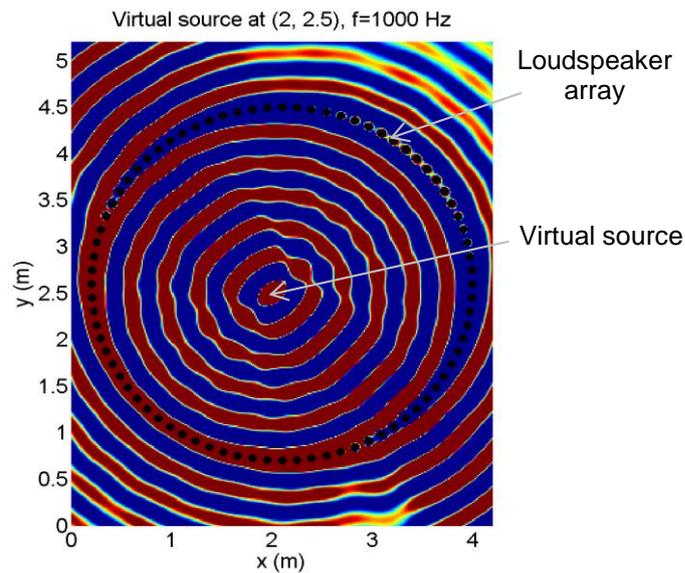


Figure 14: Sound field produced by a typical WFS system

Existing conventional WFS systems generally provide the advantages of sound performance on a large scale. The project's focus is on the fundamental development of the simplification of WFS and its application to practical 3D audio. The scaled-down WFS has an attractive advantage over the established conventional WFS where a suitable space is unavailable, and can provide a major contribution to home or private users. The solution is developed for WFS with minimum (or less) loudspeakers and defines limitations of the system, e.g. the size of a 'sweet area' (see Figure 15). For the purpose more effective distribution of loudspeakers and shape of the array can be developed through the project

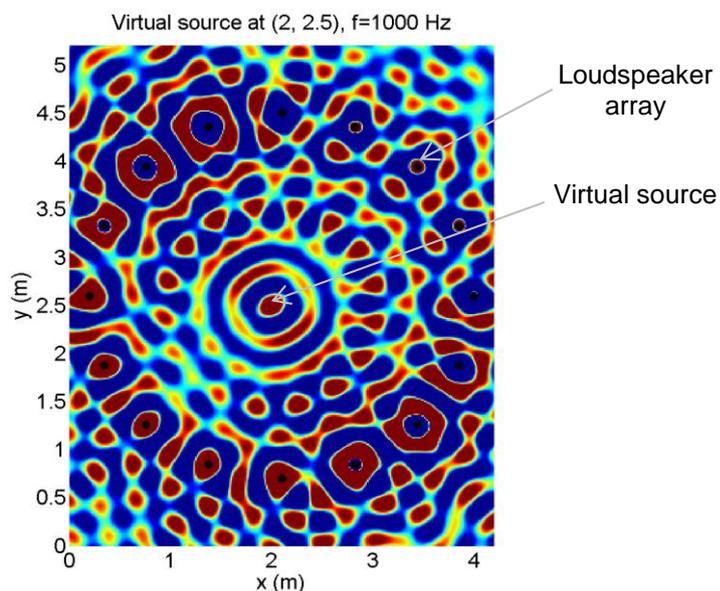


Figure 15: Sound field produced by a simplified system

Recall that the active boundary should be necessarily discretized in realisation. The reproduced sound is supposed to be identical to the original one only within a certain frequency range that is largely limited by the distance between the two neighbouring

loudspeakers. The highest limitation to the frequency range corresponds to the spatial Nyquist frequency (Equation 4).

$$f_{nyq} = \frac{c}{2\Delta x}$$

(Equation 4)

where  $c$  is the speed of sound and  $\Delta x$  is the distance between the two loudspeakers. Spatial aliasing can happen when the signal's frequencies exceed the Nyquist frequency.[23] Spatial aliasing might result in a spatial distortion of the sound field between the listener and source in the listening area. In order to reduce the number of the drivers for the simplification  $\Delta x$  in (Equation 4) can be defined by optimising the combination of  $f_{nyq}$  and the relative position of the listener to the virtual source.

## 2.4.2 Audio Rendering Implementation Framework

### 2.4.2.1 Conventional 5.1-channel rendering system

Most modern consumer PCs and audio devices support 5.1-channel surround audio rendering. In PC applications, the processing units and D-A converter, in the form of an internal chipset or of an external hardware device, are controlled by software applications by means of various audio data transfer protocols through interfaces that allow for the required bandwidth for the multichannel audio. Particularly for professional audio production under Windows environment, the ASIO (Audio Stream Input/Output) protocol is the most widely known multichannel audio driver format. Developed by Steinberg, this is designed to bypass the generic communication layer of the operating system and to enable direct connection between the audio device and the software application. It therefore allows for audio data transfer for larger number of channels with low buffering latency. In order to handle a large number of reproduction channels, professional audio rendering devices usually operate as external hardware with the input/output ports and connectors. They are connected to the controlling PC via interfaces such as USB, Firewire, and PCI Express, with the former two being most widely utilised. Figure 16 conceptually describes the framework, the data flow and the operations, which is applied in the audio rendering system with the ROMEEO project.

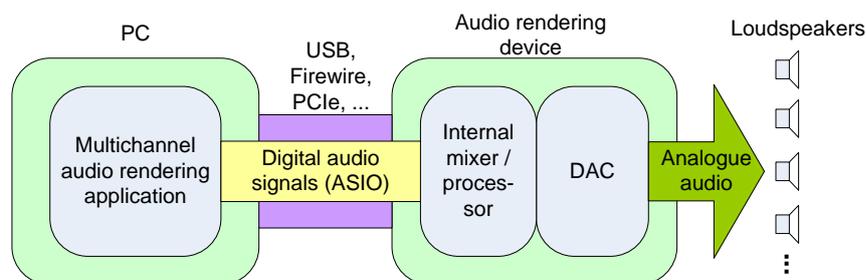


Figure 16: Typical multichannel audio rendering framework using external rendering hardware and ASIO protocol

### 2.4.2.2 Simplified WFS system

The WFS software generates signals for each loudspeaker and hence the virtual sources in the scene. The software is developed with MS Visual C++ in MS Windows 7 64-bit environment. In addition, several models have been developed for the simulation of WFS algorithm. Regarding the verification of the main driving function, it is helpful to start by analysing a simple case in the first instance. The advantage of this is that not only the system can be modelled easier but the sensitivity of the system and spatial reverberant effect of the room can also be determined. This is important for the WFS implementation, because the big shortfalls of WFS in practical situations are that it requires both an anechoic environment for

accurate sound field reproduction [24][25] and continuous bulky loudspeaker arrays surrounding a domain. Therefore, it is important to quantify the reverberant effects of the given room and to reduce the number and size of drivers in quantity and volume. The earlier considerations in the literature on this subject of WFS appear in [26][27]. The effect of reverberation is very limited in the current project as long as the interior walls and ceiling of the room, except a window glass and door, are entirely covered by sound absorptive panels and the floor also by thick absorptive fibreboard.

The main process of DSP is illustrated in Figure 17. The data loaded from the input consists of the audio signal, source information, sampling frequency etc. The source information includes the spatial position vector of each virtual sound source. In addition, the distance from the loudspeaker array to the listening position, the number of drivers and a gap between them are given in advance.

The first step into the algorithm is calculation of the distance between the driver and source. The driving function calculates the time in each sample that the sound front takes from the virtual source arriving to each driver, based on the input data. Then the convolution of the output driving signal is performed with the impulse response of each driver. The signal after the convolution is described in a frequency domain through the FFT process so that the directivity can be included. The directivity of each driver according to the frequency and angle of emission is measured before the WFS algorithm starts. Generally, it can be measured in the calibration process in advance. The frequency contents are transformed back into time domain via the inverse FFT (IFFT) process. The output vectors can be saved as audio files.

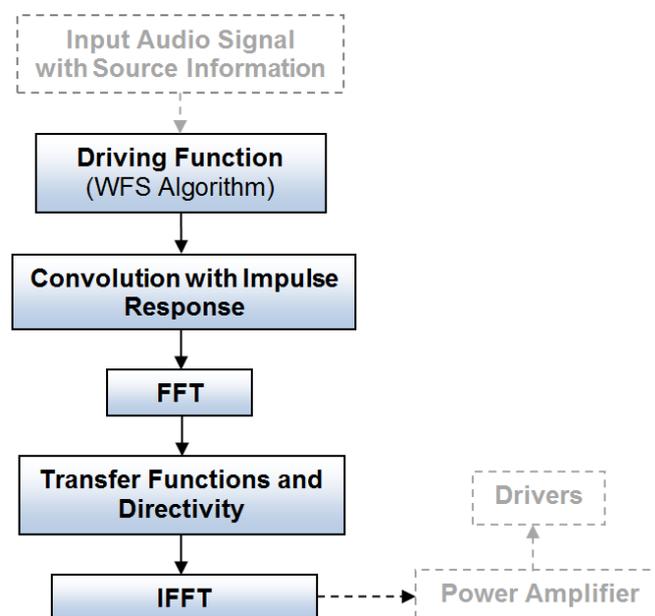


Figure 17: Block diagram of main software algorithm

In practical realization, the secondary source can be implemented with arrays of number of discrete drivers. The hardware modules for WFS are illustrated in Figure 18. It is supposed that the measurement system obtains beforehand the audio signals and additional objective information for the original sources.

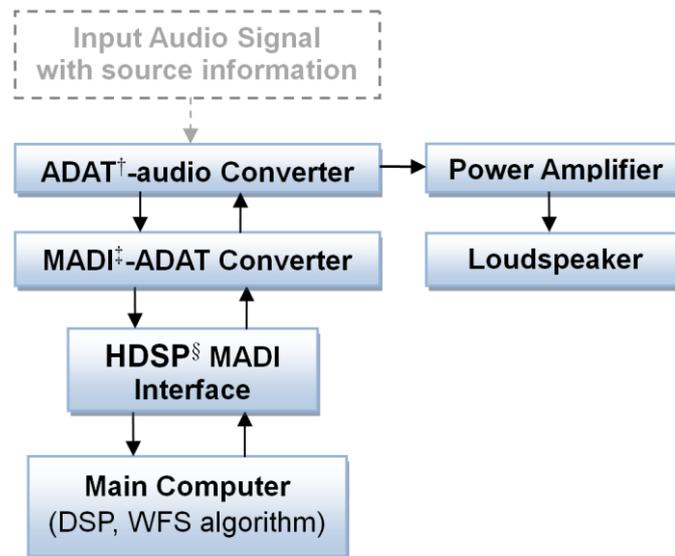


Figure 18: Block diagram of main hardware, <sup>†</sup>Alesis Digital Audio Tape protocol, <sup>‡</sup>Multi-Channel Audio Digital Interface, <sup>§</sup>Hammerfall DSP

### 2.4.3 Audio Rendering System Settings for the Fixed Terminal

In this section, the system and settings available for the spatial audio rendering with the fixed terminal are introduced.

#### 2.4.3.1 5.1-channel rendering

For the 5.1-channel spatial audio rendering, I-Lab, University of Surrey has 16 units of Genelec 8030A loudspeakers along with 2 Genelec 7060B loudspeakers for the LFE channel, as can be seen in Figure 19. These can be set up at various places for audio-specific applications, or for cases where visual displays of various sizes need to be accompanied by the multichannel audio.



Figure 19: Loudspeakers available for 5.1-channel rendering in ROMEEO

These are driven by a dedicated external audio rendering hardware, MOTU 828 mk2 (Figure 20). This is connected to a PC with Windows 7 OS running the software applications, via Firewire.



Figure 20: Multichannel audio rendering hardware in ROMEEO

As previously mentioned, the 5.1-channel rendering is supported by most professional and consumer level hardware, and thus by most available software applications including media players and digital audio workstations (DAWs). In the ROMEEO project, the audio decoding module as a software application has the 5.1-channel rendering feature integrated. It communicates with MOTU 828 mk2 using the ASIO driver to feed the decoded audio signals to the corresponding channels

### 2.4.3.2 Simplified WFS system setup

All the loudspeaker panels are controlled individually by a powerful PC and hardware control modules situated outside the studio as shown in Figure 21. The control PC is equipped with HDSP cards which are compatible with fast PCI express buses. Digital signals are then fed into D/A converters. Each of them converts 64 channels of the digital inputs into analogue outputs. Loudspeaker cables connecting the control modules with drivers are fed through the wall to each loudspeaker panel. The control software includes a series of audio APIs as libraries to implement complicated audio DSP techniques and WFS algorithms.

The WFS system takes place in a studio lab (4.2 x 5.2 m<sup>2</sup>) at the University of Surrey. A plan view of the new WFS system in the lab is illustrated in Figure 21.

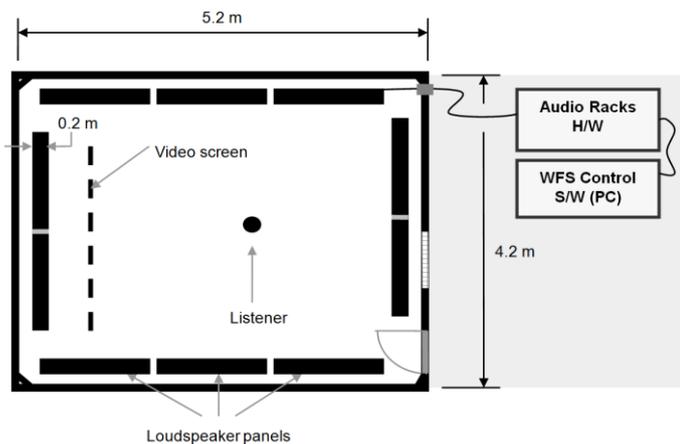


Figure 21: Arrangement of WFS setup at I-lab

Currently ten loudspeaker panels (see Figure 22) as secondary sources stand close to the walls of the studio inside. In order to prevent an unwanted reflected sound the drivers can be mounted into the walls of the reproduction room. In addition, layered loudspeaker arrays and absorptive walls are also considered.



Figure 22: Loudspeaker panel

In many practical cases, e.g. at ordinary private houses, bulky loudspeaker arrays for typical WFS systems are not available to be mounted permanently in a small room. From the practical point of view, hardware of conventional WFS systems, especially loudspeaker arrays, should be scaled down in size and also in number.

Regarding positions of the virtual sources, a further simplification of the active boundary is possible. If the sources are created in a particular side only, the array can be shortened. In this case the array serves as an acoustic window through which the sound entering from the side is supposed. This simplified version of a wave field synthesis can be suitable for modular construction of the WFS system as the active boundary covering a complete 360 degree angle can be readily extended from it.

Unlike many other existing WFS methods, our approach aims to implement a simplified solution by using a practically reduced number of drivers in a limited listening angle corresponding to the view angle to a video screen as shown in Figure 23. By requiring only limited hardware resources, the unique characteristics of the proposed approach can provide a practical and cost-effective WFS system for general home users. In addition, the system may be equipped with simple and easy user interfaces as well.

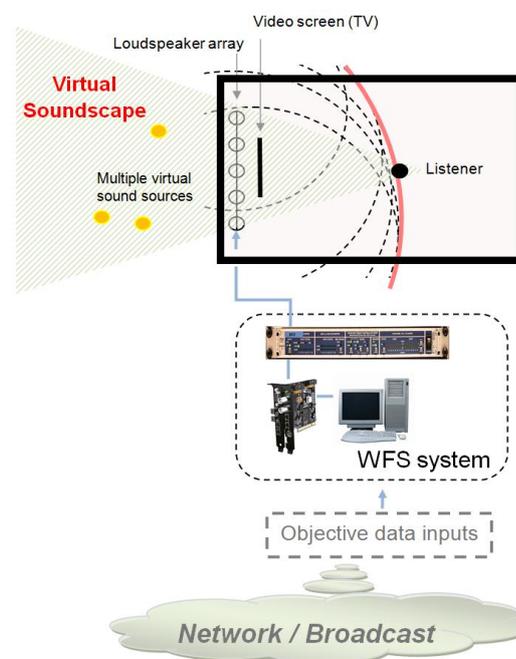


Figure 23: General configuration of the simplified WFS system

## 2.4.4 Virtual Source Localisation based on WFS

### 2.4.4.1 Focused source

This section describes some studies on numerical models to validate the fundamental characteristics of the driving function based on the WFS approach in a space. The general solution (Equation 3) is applicable in 2D, and also in general 3D cases.

The geometry in the numerical simulation is given to reflect the actual physical conditions of the studio where the WFS system takes place (See Figure 21). The valid problem domain for the solution (Equation 3) is situated in anechoic conditions. The domain is defined in the area where  $0.5 < x < 3.5$  m and  $0.3 < y < 4.5$  m. The gap between the two neighbouring control sources is defined as 0.17 m. The active control boundary consists of 16 acoustic monopole sources, e.g. loudspeaker drivers in practice. An example of the typical result is illustrated in Figure 23.

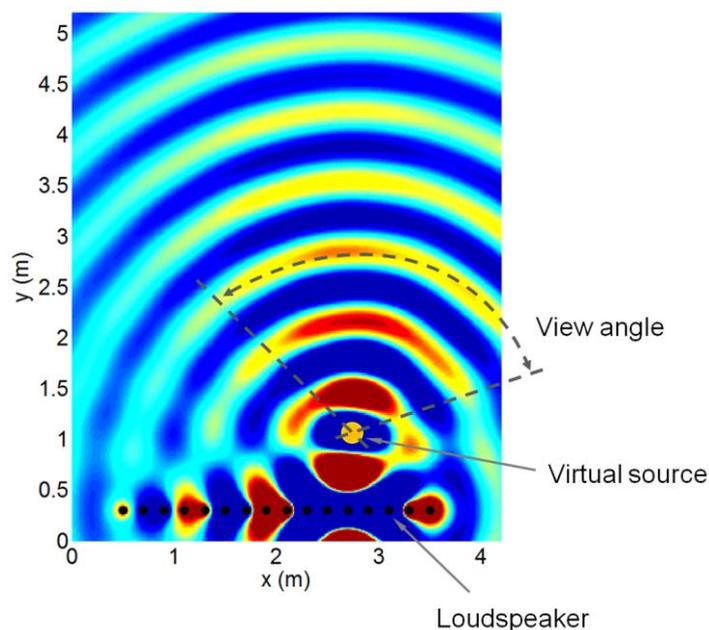


Figure 24: A focused virtual source at (2.8, 1), 500 Hz

In the Figure 24 a virtual point source vibrating at 500 Hz is supposed to be situated at the position  $x = 2.8$  and  $y = 1$  m, i.e. inside the problem domain. When the solution (Equation 3) based on the WFS theory is applied to the space, it is found that the combination of all the output wave fronts generated by the 16 drivers produces a wave front equivalent to the original virtual one in a given domain. Near the boundary especially on the left hand side in Figure 23, some artefacts are found. This is due to the fact that the limited length of the secondary source array can cause discontinuity of a reconstructed wave front at each end. Overall, the typical result shown as an example is mostly valid within the view angle which is indicated in Figure 24. The result shows that the solution (Equation 3) which is used for an audio source created outside the domain in Figure 13 is also effective and can be extended to the case for a focused source after some relevant modifications.

### 2.4.4.2 Experimental validation of the source localisation

Regarding the validation of the 3D audio solution based on the WFS, it is helpful to start by analysing a problem with a single virtual point source in the first instance. However, it does not mean that the use of the application is limited to only problems with a single virtual source. The solution remains applicable even in a case where complex sounds of multiple virtual sources are created, if the system resource requirements, e.g. CPU cores, memory, can be

satisfied. The position of a virtual source is determined by measuring the propagating direction of the sound field in a studio situated in the I-Lab. The studio is sufficiently absorptive to allow just few reverberations on the walls.

The measuring system is composed of a combination of B&K half-inch free-field measuring microphones, as shown in Figure 25(c).

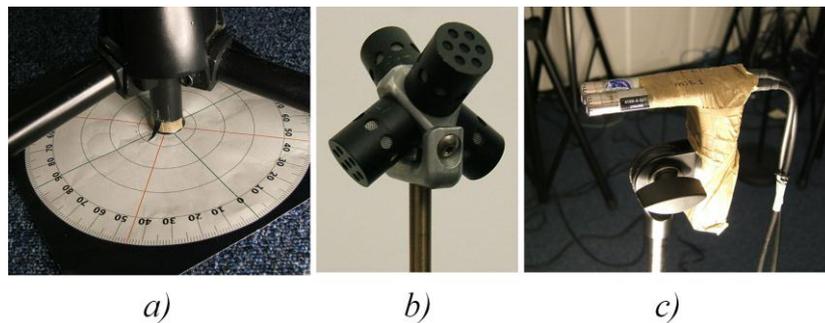


Figure 25: Measuring setup including a) angle meter, b) B-format microphone, and c) a pair of half-inch free-field microphones

In order to identify the position of an acoustic source created by the rendering system, the perpendicular direction to a wave front is chased by using the two microphone setup mounted at predefined reference positions. The technique leads to determine basically the maximum pressure gradient at a measuring position. The phase difference between a pair of the microphones are measured using a B&K PULSE Sound & Vibration analyser with the WFS sound rendering system switched on. Additionally, the directions of sound intensity vectors are also measured by using a B-format microphone in order to consolidate the result. The configuration of the measuring system and B-format microphone is shown in Figure 25 (b).

Figure 25 shows that the solution based on the WFS provides for a localisation of a virtual source situated at around the initially targeted position which appears as + in red. After the WFS system is turned on, the minimum phase between acoustic pressures is measured at different points, i.e. 'A' and 'B' along the reference line in the rendering domain (See Figure 26). In the experiment a broadband linear swept sine signal is used to generate a sound field. The signal is swept forward in frequency up to 1.6 kHz with the rate of 380 Hz/s.

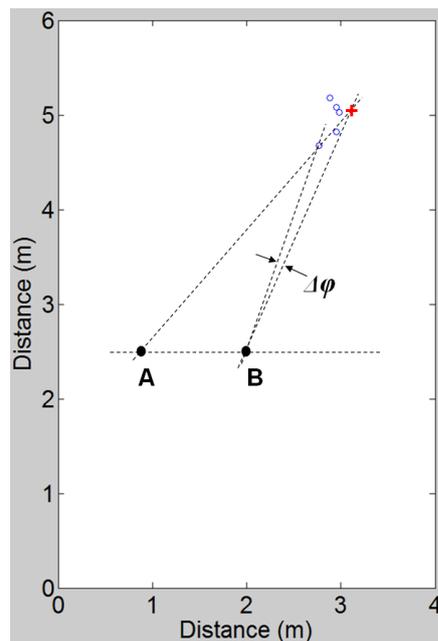


Figure 26: Virtual source positions on the horizontal cross-section of a space, +: originally intended, o: measured at different positions on the reference line. Points 'A' and 'B' denote positions where the monitoring microphones are mounted

The errors  $\Delta\phi$  are generally below 7 degrees in the frequency range between 80 and 1600 Hz, as illustrated also in Figure 26. The error sources in acoustic measuring can be classified largely in two groups, one related to position, and the other to time. The physical size of the measuring devices and the gap between two neighbouring loudspeaker drivers causes some errors in monitoring, as long as the sound field is measured as close as possible to the effective centre of the control source unit. For instance, errors in the measuring position,  $\Delta x$ , and the separation between loudspeakers forming an active boundary,  $\Delta l$ , may exist in the realization of any multi-channel audio rendering system.  $\Delta l$  in the measurement equals 0.17 m. The uncertainty which can be caused by both  $\Delta l$  and  $\Delta x$  corresponds to about 10 degrees in the measurement. In addition, time delay and extra phase errors can occur in digital signal processing or measuring equipment. Therefore, the resulting errors are not uncommon in real environments. In the context of human hearing perception also the errors are quite acceptable in practice as long as the ear generally cannot detect the change of a source position when the direction is altered by about 10 degrees.

In addition, instead of detecting the minimum phase as shown above, the source direction can be identified by measuring sound intensity vectors [28]. x, y, z components of the intensity vectors in a space are measured by using a B-format microphone. The dominant intensity vectors obviously show the overall directivity of the sound field near 106 degrees which is the intended virtual source direction in Figure 27. In the result the data set is spread near the exact source direction. One of the main reasons can be found in the fundamental theory that the global wave front generated by the driving function is composed of a lot of smaller secondary wavelets produced by each loudspeaker. In actual hearing the dispersion of the vectors can affect colouration and localisation of the audio object [29]. However, the adverse effects are rather limited and controllable by increasing the number of drivers in practise.

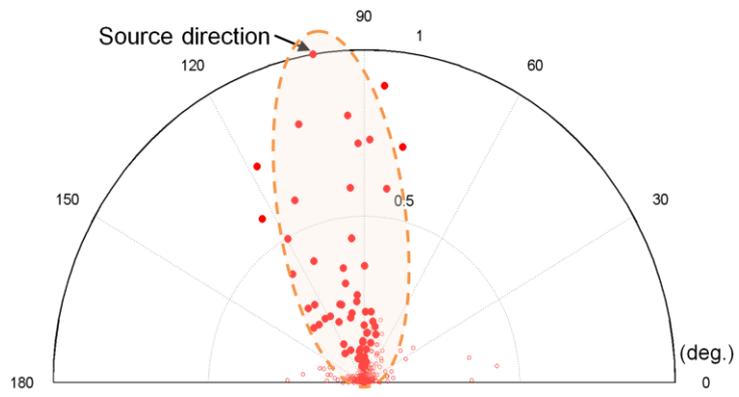


Figure 27: Direction of the virtual source

### 3 VIDEO RENDERING

#### 3.1 Introduction

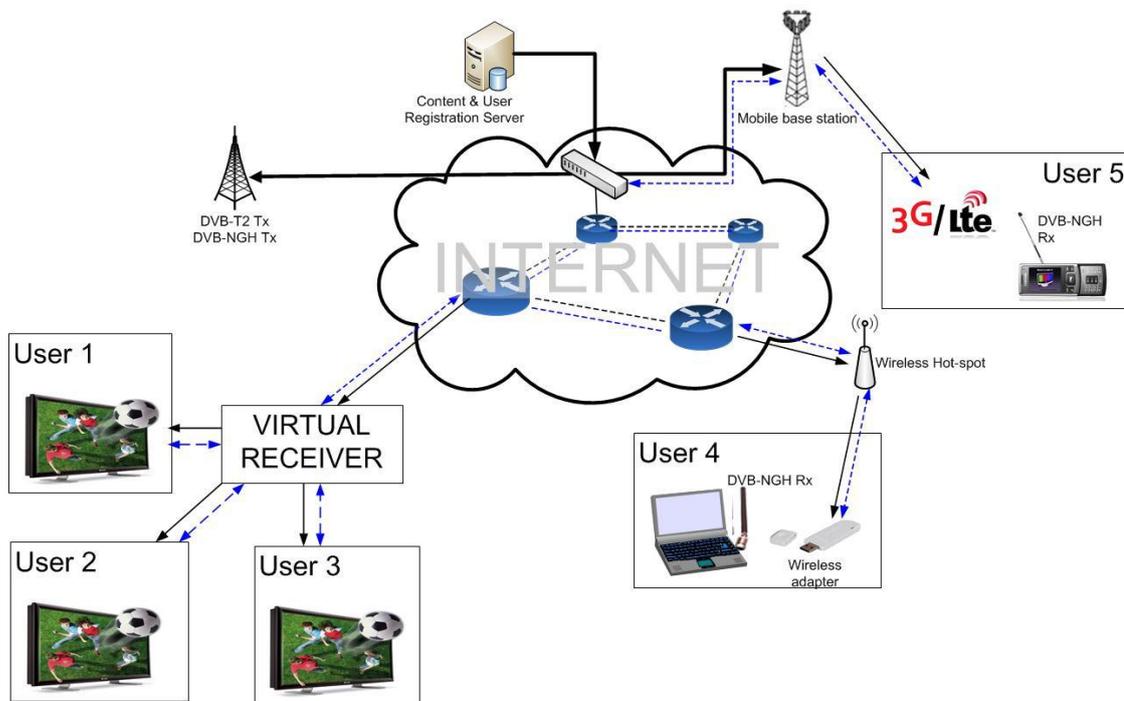


Figure 28: Overall ROMEEO demonstration scenario

As described in Figure 28, the final ROMEEO demonstrator will address different users having different terminals. Each terminal has its own video processing capability and dedicated video rendering will be provided depending on this capability. There are 3 different terminal types:

- The fixed terminal corresponding to the highest processing capabilities
- The portable terminal
- The mobile terminal

In the following section, respective processing will be presented showing adaptation of the processing to terminal capabilities.

#### 3.2 Fixed Terminal

##### 3.2.1 Experience acquired during the MUSCADE project (range of disparity)

In the field of the FP7 European project Multimedia SCALable 3D for Europe (MUSCADE), an end to end chain has been demonstrated from the multi-view capture up to a multi-view rendering. The project was targeting two different display technologies: multi-view auto-stereoscopic displays (e.g. Alioscopy 8-view display) and light-field displays. As described in [30] the light field display can render a field of view of about 70°. To achieve that, the MUSCADE content was shot to ensure a large range of disparity. Details of shooting are described in the MUSCADE deliverable <sup>1</sup>.

This very large range of disparity was not optimized for auto stereoscopic multi-view displays. There are too main drawbacks working which such high range of disparities:

<sup>1</sup> <http://www.muscade.eu/deliverables/D2.2.2.PDF>

- For transmission purposes, the disparity value should be encoded using a standardized 8-bit encoder (2 disparity maps encoded together using MVC encoder). To fit with the 8-bit limitation, the disparity between satellite and central cameras is normalized compared to the one between central views. After transmission, on the rendering side, the 8-bit representation is too low to allow a precise Depth Image based Rendering (DIBR) processing. It is not possible to process the disparity with a sub-pixel precision.
- Large disparities on multi-view display can create undesirable cross-talk. Only a limited number of views are displayed on such screen. Any misalignment of the lens array on the screen or/and a wrong position of the user create a cross-talk. The level is directly related to the disparity range. The current level of the technology cannot ensure any cross-talk free system. It is then important to ensure that the level of disparity is not too high. The MUSCADE content shot with large baseline (above 30cm) was then processed to avoid such problem.

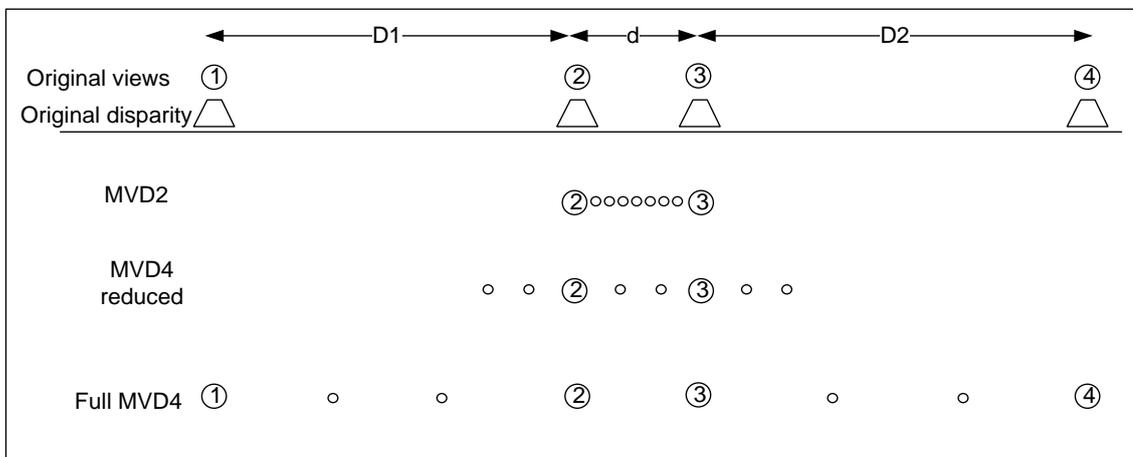


Figure 29: MUSCADE camera and relative view interpolation positioning

As described in Figure 29, the camera baselines are different between central cameras ( $d$ ) and between central and satellite cameras ( $D1$  or  $D2$ ). The ratio between  $D1$  and  $d$  is between 4 to 5. In MUSCADE, a scalability was defined whether the end user receives the full content (called MVD4 for 4 views + 4 associated disparity maps) or only part of it (MVD2 corresponding to central views + 2 associated disparity maps). In Figure 29 we describe the relative position of interpolated in the different configuration. In MVD2 case, only central views are received and interpolated are in between central views (2 and 3). In theory in MVD4 case, we should apply a scheme corresponding to the full disparity range the MVD4 mode can provide. But the corresponding disparity range of such mode was too high for autostereoscopic displays. The cross-talk level was just not acceptable. It was then mandatory to modify the interpolation scheme by creating what was called a reduced mode (MVD4 reduced in Figure 29). The relative disparity between each view is globally  $1/3$  of the disparity between central views.

### 3.2.2 Range of disparity defined for ROMEO

To avoid having too large range of disparity, we defined a camera set-up with smaller baseline for satellite cameras than in MUSCADE. The shooting conditions are specified in the ROMEO deliverable D3.1: *Report on 3D media capture and content preparation*.

In Figure 30, the camera and relative view interpolation positioning are described. In one of the camera set-up chosen in ROMEO, inter-camera distance is constant (called  $DR$ ). One

associated interpolation format is described in Figure 30. An adaptation to user preferences is described in D3.4 deliverable.

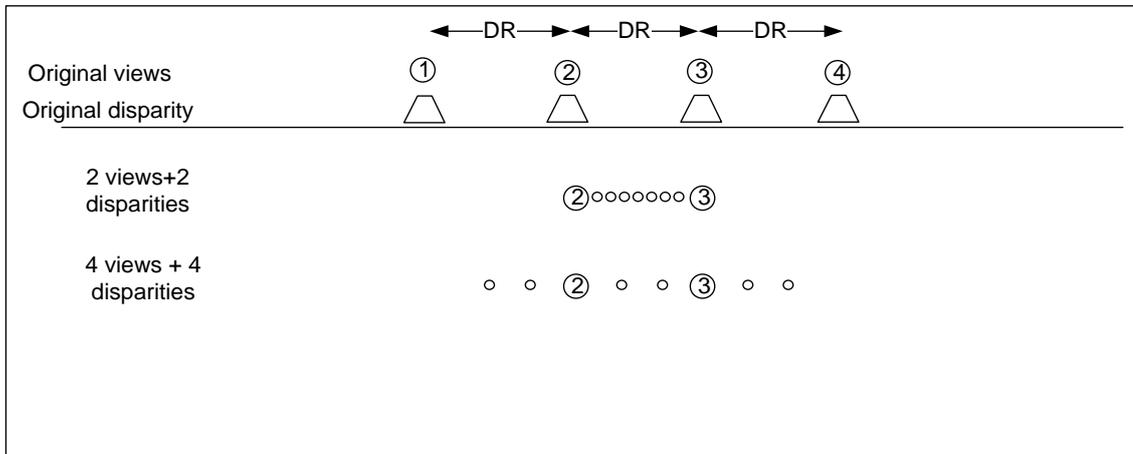


Figure 30: ROMEEO camera and relative view interpolation positioning



Figure 31: ROMEEO content: ACTOR sequence



*Figure 32: Disparity map of the ACTOR sequence*

Figure 31 is corresponding to one picture of the ACTOR sequence, shot during the second ROMEO shooting in Paris. The corresponding disparity map extracted is shown in Figure 32.

### **3.2.3 Depth Image Based Rendering Algorithm improvements**

#### **3.2.3.1 Overview**

To feed the multi-view auto-stereoscopic displays, it is necessary to generate extra views. Also for depth adjustment, it is necessary to interpolate intermediate views corresponding to the 3D level requested by the user. In both cases, these views are interpolated using a Depth Image Based Rendering (DIBR) algorithm [31]. These interpolations are done using the nearest available views on each side of the interpolated view as well as the disparity map associated to this left view (the nearest available view on the left).

The disparity map corresponding to the interpolated view is first interpolated. During this disparity map projection, information on the occluded areas (part of the 3D scene that is not visible in one view) is determined. And then the video information in the left and the right views pointed by the interpolated disparity values are used and combined depending on the occluded area information to render the interpolated view.

#### **3.2.3.2 Disparity map interpolation**

The main part of the disparity map interpolation consists in projecting the left disparity map to the intermediate position. Each disparity value is first scaled, i.e. multiplied by a coefficient between 0 and 1 corresponding to the position ratio of the interpolated view in relation with the corresponding original views considered for the interpolation. The coefficient is near 0 if the interpolated view is near the left view. This coefficient is near 1 if the interpolated view is near the right view. Then each disparity value is projected at the position defined by this scaled disparity. This position is rounded to the nearest pixel location (Figure 33).

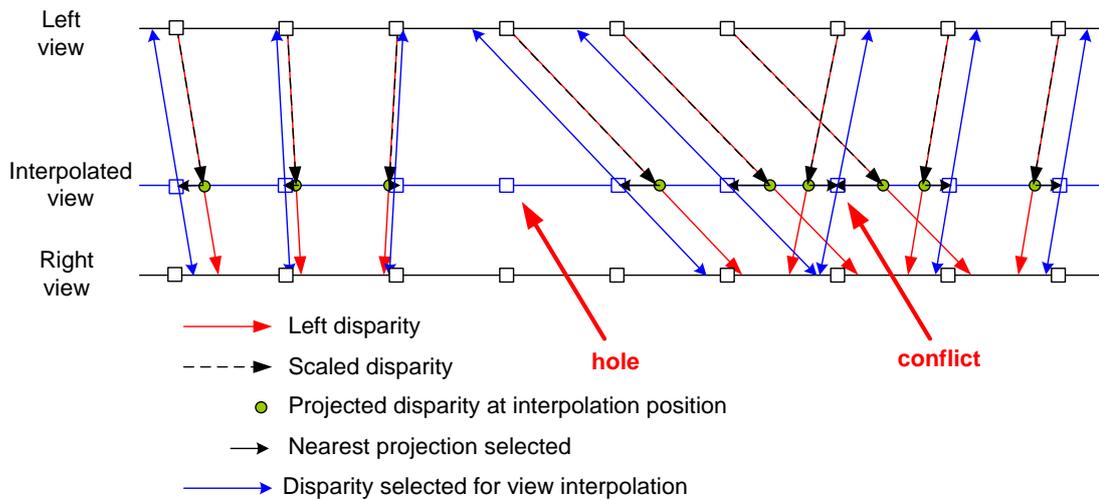


Figure 33: Projection of the left disparity map

In order to solve conflicting disparity values (different disparity values pointing to a same pixel location), the left disparity map is scanned from left to right so that information concerning occluding pixels overwrites information concerning occluded pixels. In order to fill the disoccluded areas (the holes), during the projection each disparity value is assigned to all pixels on the right side of the previously assigned pixel up to the pixel the current disparity value points at.

### 3.2.3.3 Occlusions identification

It is necessary to identify the regions of the interpolated view that are only visible in one view in order to be able to render them using only this view. These regions correspond to parts of the scene that are occluded in the other view.

For this purpose, the object positions in both views are required. This information is obtained from the left disparity map, which contains the position of the objects in the left view. The position of the objects in the right view is obtained by projecting this disparity map onto the position of the right view using the same process as the one described in the previous section. The projection of the left disparity map is used instead of the provided right disparity map in order to avoid additional errors. The occluded areas are then identified by comparing the projection of both disparity maps at the synthesized view position.

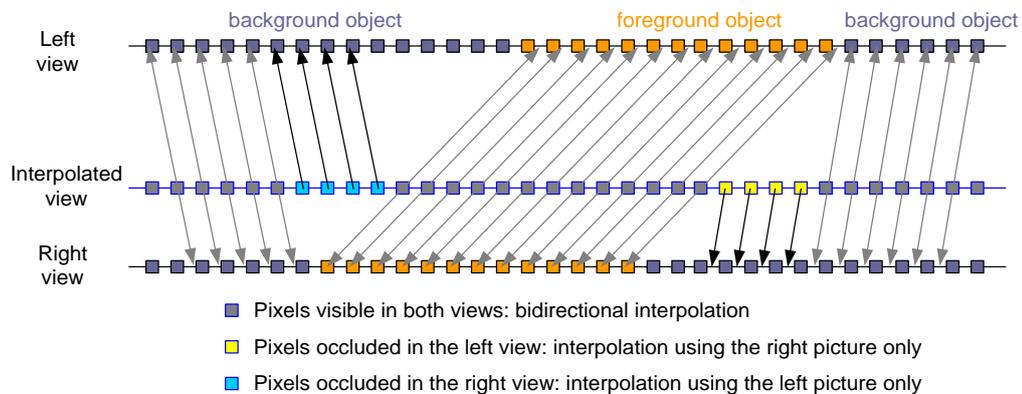
Information on what is occluded in the right view is obtained by comparing the disparity values in the left disparity map and the disparity values they point to in the right disparity map. Positive differences indicate objects visible in the left view but occluded in the right view. This comparison process is applied by scanning the left disparity map from left to right and by projecting it onto the view to be interpolated.

Information on left occlusions view is obtained by a similar process by comparing the disparity values in the right disparity map and the disparity values they point to in the left disparity map. Positive differences indicate objects visible in the right view but occluded in the left view. This comparison process is applied by scanning the right disparity map from right to left and projecting onto the view to be interpolated.

### 3.2.3.4 Occlusion border management

The occlusion information tells which view (left or right or both or none of them) can be used to render each pixel. Accordingly during video interpolation, in the occluded areas unidirectional

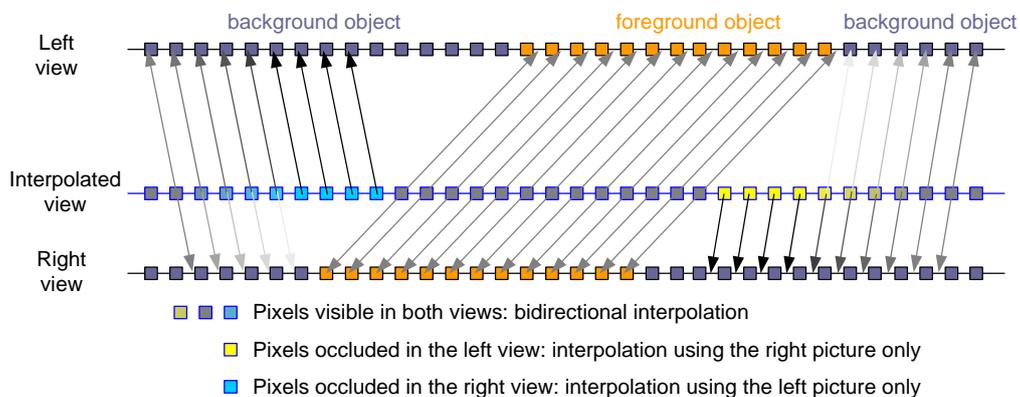
interpolation is used to synthesize the new view, while in the rest of the picture bidirectional interpolation is used, as shown in Figure 34.



*Figure 34: Interpolation scheme without border management*

The abrupt transition between these unidirectional (in the disoccluded area) and bidirectional interpolations can however be visible for various reasons. For instance, the disparity map does not necessarily match perfectly the object borders. Particularly in case of blurred borders it is difficult to define where the object ends and where the background starts. This can generate echoes in the new view. Also differences of luminance, color, reflections or flare between left and right views can create halos around objects.

In order to reduce the visibility of the aforementioned artifacts, the transitions between the unidirectional and the bidirectional interpolations are softened on the borders of the occluded areas. The coefficients used for the bidirectional interpolation are changed on the right of the regions occluded in the left view, and on the left of the regions occluded in the right view (Figure 35).



*Figure 35: Interpolation scheme with border management*

The value of the blending coefficient depends on the results of the occluded area borders management. It means resolving issues of areas located just between regions occluded in the left view and regions occluded in the right view, and also the case of regions occluded in both views.

The size of the altered area around the occlusion is defined depending both on the width of the occlusion and the difference of disparity between the background and the foreground objects. Thus, since this disparity difference depends on the inter-axial camera distance, a specific parameter has been introduced in order to avoid excessive filtering. Results are shown in Figure 36, the halo around hairs is strongly reduced.



Figure 36: Interpolation results: without (top) and with (bottom) border management

### 3.2.3.5 De-aliasing

Aliasing can appear on the object borders during view synthesis [32]. This effect occurs because in the 1-layer disparity map format, only one disparity value is stored per pixel. Therefore in the disparity map object borders are located at pixel borders, and thus also the object borders in the interpolated view.

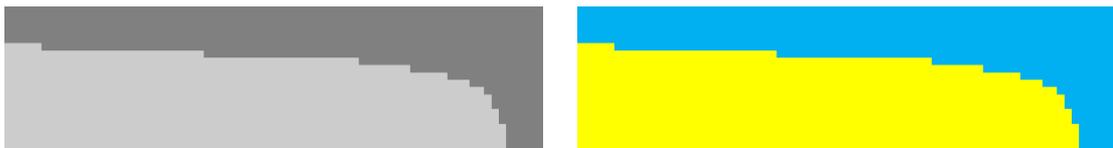


Figure 37: Disparity map and corresponding interpolated video without de-aliasing

To solve this issue, the aliased borders are detected in the interpolated disparity map (by comparing the disparity values) and a sub-pixel shape is virtually added to smooth these contours. Therefore one line of pixels along the object borders uses a 2-layer representation, this means two disparity values, and thus a combination of background and foreground information.

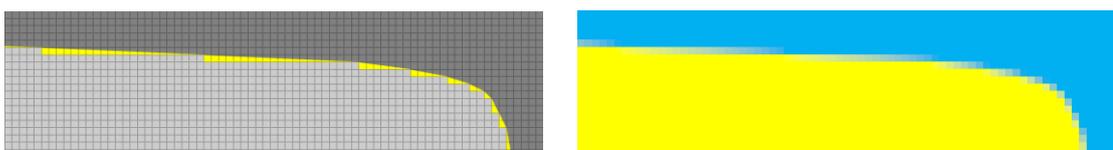


Figure 38: Disparity map and corresponding interpolated video with de-aliasing

This process is illustrated on a basic example in Figure 37 and Figure 38. Figure 37 shows on the left an interpolated disparity map and on the right the result of interpolation without de-

aliasing. Figure 38 shows on the left the same disparity map with the virtual shape of foreground information added onto the background of the aliased contour and on the right of the corresponding result with de-aliasing.

Figure 39 shows the result on real content. It can be observed how the aliasing and the “cut-out” appearance are reduced.



*Figure 39: Interpolation results: without (top) and with (bottom) de-aliasing*

### 3.2.3.6 Interpolated view rendering

The video information of the view to synthesize is interpolated using the video samples of the left and the right views and the disparity value (or the two disparity values in case of a de-aliased contour). Linear interpolation is used to point at sub-pixel position in the left and the right views. Finally the video samples pointed by the disparity value(s) in the left and the right views are combined depending on occlusion information and the occlusion border management results.

### 3.2.4 Algorithm adaptation for parallelization

In order to reach real-time constraint, the Depth image based rendering described in the previous section must be integrated into a GP-GPU (General Purpose Graphic Processor Unit) processor. The target processor is a Nvidia GPU processor (Tesla board C20) which allow developing code in CUDA 2.0.

#### 3.2.4.1 CUDA programming model

The CUDA programming model is based on an extension of C, allowing the programmer to define C functions called “kernels”, that will be executed N times in parallel by different CUDA threads on the GPU. It means that each thread executes the same C code, applied to different data, depending on the thread index.

These threads are group in “warps”. The number of thread per warp depends on the hardware architecture (32 in our case) and cannot be modified by the programmer. All the threads of a warp will execute the same instruction. It means that if only one thread have to execute a part of code after a conditional test, all the other threads of the warp have to wait during this

execution, reducing the global efficiency. By definition, all the threads of a warp are synchronized.

Then, the threads are organized in “blocks” of threads, defining a “grid”. The total number of threads executed by a kernel is equal to the number of blocks multiplied by the number of threads per block. The number of threads per block and the architecture of a block (1D, 2D or 3D) have to be defined by the programmer, depending on the problem to be solved, and performance considerations.

The variables used in the thread’s C code will be affected by the compiler to registers. The number of registers is limited in each multiprocessor composing the GPU. If too many registers are required, the number of blocks of threads resident in a multiprocessor is reduced, reducing the device occupancy.

In a block, the threads can be explicitly synchronized by the programmer and can share a small amount of memory, the “shared memory”. If this amount of shared memory required per block is too high, the number of blocks of threads resident in a multiprocessor is reduced.

### 3.2.4.2 View interpolation CUDA kernel

The auto-stereoscopic monitor is a HD display (1920 x 1080 pixels) where eight views are shuffled, at a sub-pixel level. It means that each view composing the multi-view content has a resolution of 1/8 HD. In comparison, on a 3D stereoscopic monitor supporting side by side input video, the horizontal input resolution is divided by two.

Since the final resolution of the interpolated picture is lower than full HD and in order to limit the amount of shared memory required by thread block, it seems an acceptable trade off to perform the disparity projection with a horizontal resolution of 960 disparity pixels per line.

Even if the target hardware for Romeo is a Tesla board having a CUDA 2.0 capability and implying a maximum of 1024 threads per block, the CUDA code has been developed to be compatible with CUDA 1.3 capability, to support hardware of the previous generation. In this case, the maximum number of threads per block is 512.

The number of threads per block has been set to 480 and these blocks are 1D blocks (blocDim.x=480, blocDim.y=1). The 480 threads of a block are executed in parallel to perform the projection of a full disparity line. So, each thread has to process 2 disparity pixels. To process a full picture, a grid composed of 1080 blocks of threads have to be launched. It means that 518400 threads are launched to perform a projection. Since a line is composed of 1920 pixels (or 5760 sub-pixels), each thread has to compute 4 RGB output pixels (or 12 sub-pixels).

To save memory access time and to allow threads cooperation, shared memories have been used to perform the projection. These memories are faster than global memory and can be shared by all the threads of a block of threads. The shared memory `DispIn[960]` receives the input disparity, `DispOut[960]` the projected disparity and `RightView[960]` a mask indicating the de-occluded areas.

The variable ‘x’, represents the horizontal position of the disparity pixel to be processed by the current thread. Since each thread has to compute 2 disparity pixels, each thread has to compute the interpolated disparity pixels ‘x’ and ‘x+1’. The thread 0 computes the pixels 0 and 1, the thread 1 computes the pixels 2 and 3..., until thread 479 computing the pixels 958 and 959.

The first step consist in initializing the `DispIn[960]` shared memory. Each thread initializes `DispIn[x]` and `DispIn[x+1]` with the disparity values.

Then the pixel disparity from  $x-D_{max}$  to  $x-D_{min}+1$  ( $D_{max}$  and  $D_{min}$  representing the disparity range) are projected, as indicated in Figure 40. If a projected disparity provides an output disparity for pixels  $x$  or  $x+1$ , the disparity value is stored in `DispOut[x]` or `DispOut[x+1]`. When several disparities are projected to  $x$  or  $x+1$ , indicating an occlusion, the smaller disparity value is stored, corresponding to the front plane.

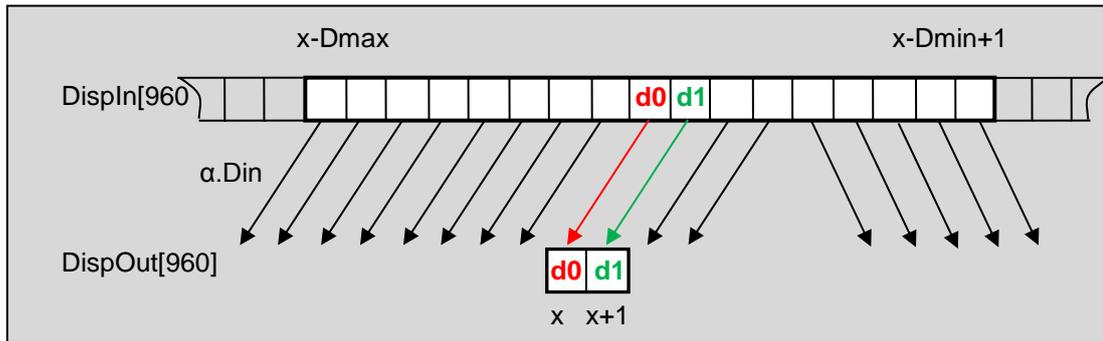


Figure 40: Disparity projection per thread

When a block of thread has been executed, the DispOut[960] shared memory contains the projected disparity, as presented in Figure 41. The occluded part in the left view did not receive any disparity value. In this case the RightView[960] shared memory receives the value 1, indicating that the projected video view will receive the pixels from the right view.

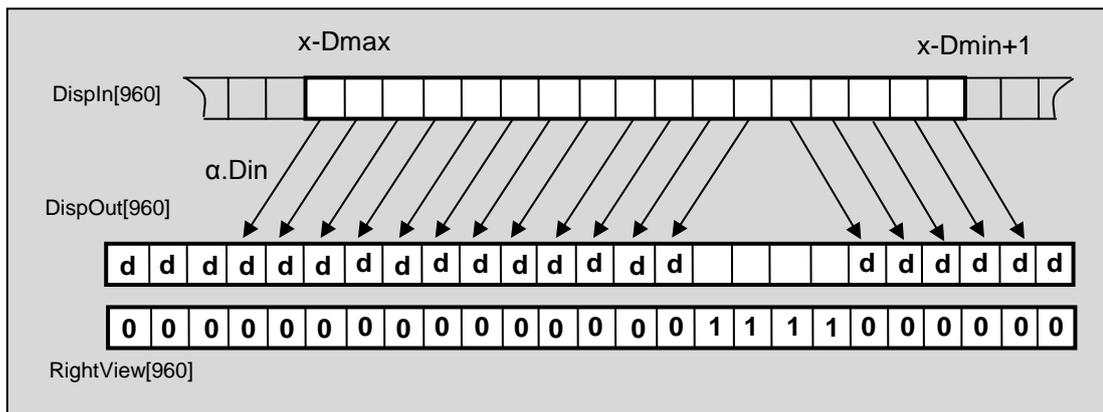


Figure 41: Projected disparity by all the threads

Then, the holes in the projected disparity map corresponding to occluded areas are filled with the background values. To minimize artefact close to the right to left view transition, a filter is applied on RightView[960] (Figure 42).



the CPUs that can be multi-threaded, the data transfers in both directions that can be performed in parallel with the kernel execution, and the massively parallel threads of the kernel executed by the GPU.

The following diagram represents the functional description of the CUDA rendering object integrated into the PC environment. It presents the different buffers allocated in the host (CPU) memory space or in the device (GPU) memory space, the data flow, and the processing operations.

The video buffers `DataVid[0..3]` and `DataDisp[0..3]` containing the frames of decompressed video and disparity, are allocated by the player part in the CPU memory space. They are dimensioned to ensure the synchronization between the different streams coming from the decoder part. The video buffers `Host_VidPin[0..3]` and `Host_DPin[0..3]` are also allocated in the CPU memory space, but these allocations are performed in CUDA, by the object in the rendering library.

The GPU board can be considered as a computing resource (a GP GPU), receiving the MVD4 video stream and providing the multi-view result. This multi-view stream is directly displayed, exploiting the possibility of interoperability between CUDA and OpenGL. In this case the output buffer is a buffer shared by CUDA and OpenGL, and the GPU board is considered by the system as a graphic card.

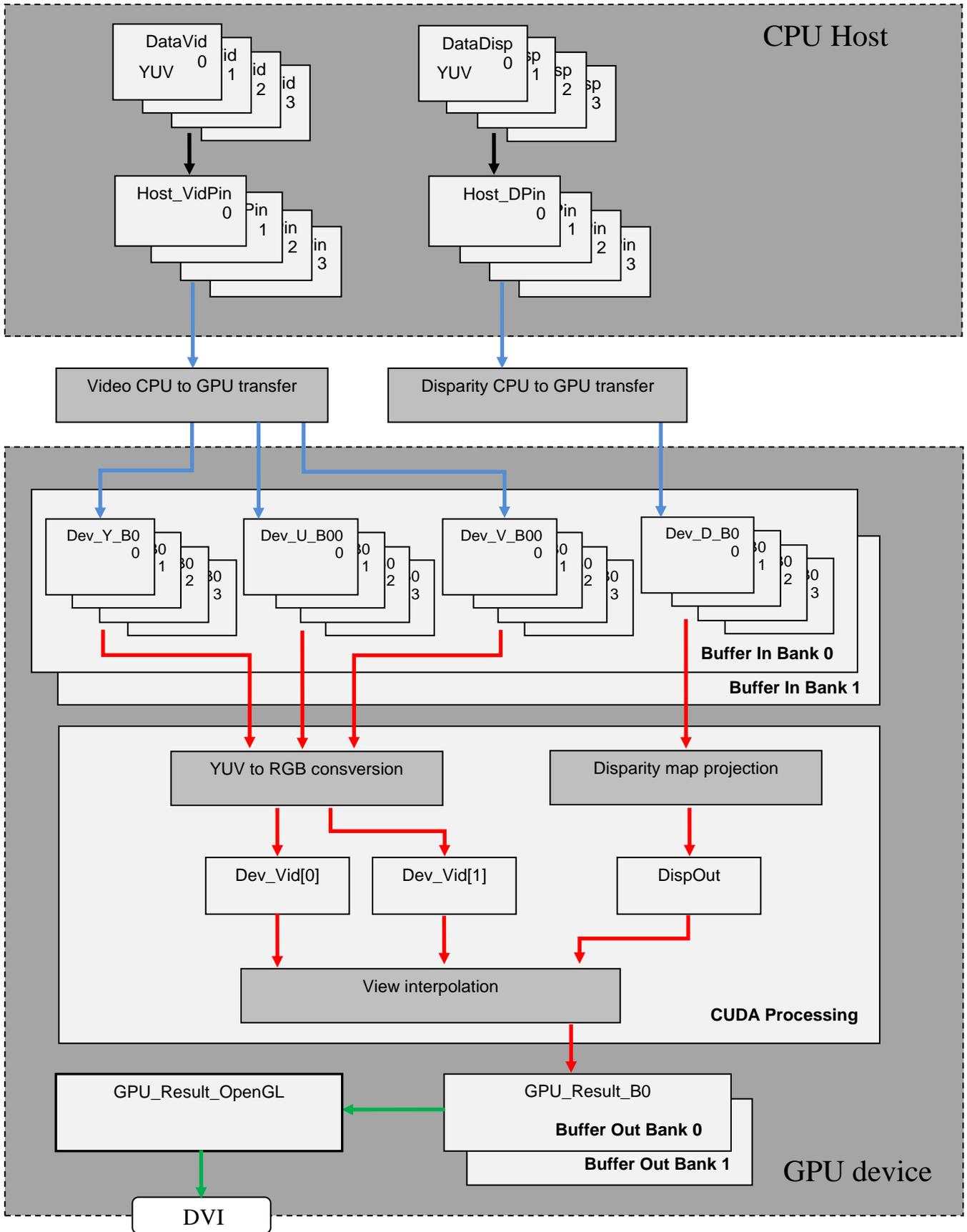


Figure 43: Functional description

To be executed concurrently, the transfer operations and CUDA computing must be able to access to different memory banks. The input video stream, comprising the video and the disparity, can be written by the CPU in memory Bank 0 or memory Bank 1. The CUDA processing can process one of the memory banks during the transfer of the other. In the same way, the multi-view result can be written by the GPU in two memory banks. This result stored in one of the memory banks can be displayed during the processing of the other. These concurrent operations, represented by the different colors, will be detailed later in this document.

To be efficient, the data to be transferred from or to the GPU must be stored in page-locked CPU memory, also known as pinned memory. This memory guaranties that the data are physically present in CPU memory for the transfer, and allows concurrent transfers and CUDA kernel execution. Since this memory is no more accessible by the operating system, the application must minimize the amount of allocated page-locked memory to avoid decreasing the overall system performance.

Even if only two views are required to compute each projection/interpolation, the four video views are transferred in a memory bank during the CUDA processing of the other memory bank. After decoding, these views are provided to the GPU in YUV 4:2:0. The buffers `Dev_Y_Bi[0..3]` receive the Luma of the four input views, and the buffers `Dev_U_Bi[0..3]` and `Dev_V_Bi[0..3]` receive the two planes of chroma. In 4:2:0 mode, the Luma buffers have the same size as the input video resolution, and the chroma buffers have a half resolution in both directions.

The 4:2:0 to 4:4:4 and the YUV to RGB transformations are performed by the GPU in CUDA to save time, compared to the CPU. Firstly, the amount of data to be transferred from the CPU to the GPU memories is reduced in 4:2:0. Secondly, the computation can be massively parallel and take advantage of the GPU architecture. For an input video size of `SizeH x SizeV` pixels, a CUDA kernel executes `SizeH x SizeV / 4` threads (518.400 threads in HD) in parallel. The results of these operations are written in the buffers `Dev_Vid[0]` and `Dev_Vid[1]`.

All these buffers are allocated in global memory. To accelerate the reading operations to these buffers, the data are accessed thanks to textures. These textures are not represented in the block diagram, but are dynamically bound to the buffers before the reading operations.

The disparity has to be transferred from the CPU to the GPU memory in the buffers `Dev_D_Bi[0..3]`. This disparity is coded with eight bits in the Luma channel of the video stream. To transform this eight bit disparity to a floating point disparity representing a real disparity in term of horizontal pixels, some parameters may be required. These parameters could be fixed, coded as constants, or could be dynamic depending on the video stream, and provided as metadata.

During the disparity projection, all the threads of a video line must be able to access to the same memory buffer `DispOut`. This buffer must be declared as a shared buffer. The position of the projection in between the two input views is provided by a phase parameter. When a line of disparity has been projected to this position, the video output interpolation can be performed. Then the next line is processed. Only the output sub-pixels representing the current projected view are computed. For a eight view rendering, requiring eight disparity projections  $P_i$ , the projection  $P_i$  (with  $i$  from 0 to 7) is performed and the output sub-pixels dedicated to the view  $i$  are written in the output `BufferOut`. It means that after each projection  $P_i$ , only 1/8 of the interpolated sub-pixels are computed and written in the buffer. The number of sub-pixels to be computed is imposed by the auto-stereoscopic monitor, 3x1920x1080 for a HD monitor.

The previous diagram (Figure 43) is a static view presenting the hardware architecture. The following diagram (Figure 44) is a dynamic representation of the different operations. It shows the level of system parallelism where the memory transfers in or from the two memory banks, the GPU processing and the CPU processing can be performed concurrently.

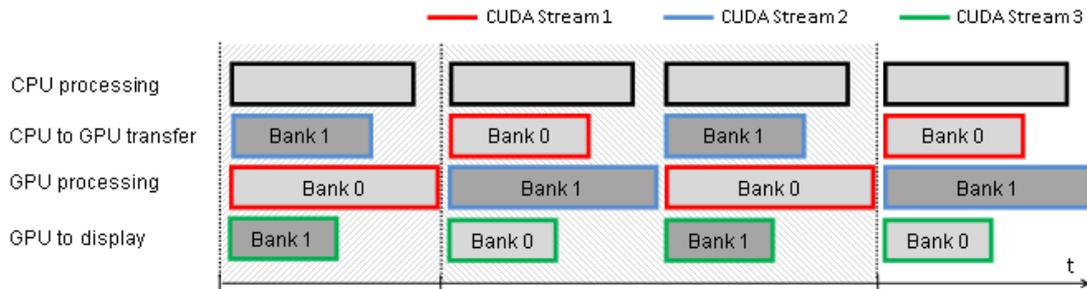


Figure 44: Concurrent execution

In order to facilitate concurrent execution between the CPU host and the GPU device, some function calls are asynchronous: control is returned to the host immediately after the call.

A CUDA application can manage concurrency through streams. Thanks to these streams, associated with the asynchronous function calls, a system can simultaneously transfer data from the CPU memory to the GPU memory, process some data available in GPU and transfer other data from the GPU memory to the CPU memory.

### 3.3 Portable Terminal

The portable terminal of the ROMEEO project targets devices such as laptops equipped with a stereoscopic display. Rendering algorithms targeted for such terminal are similar to the ones described for fixed terminal (based on DIBR techniques). The use of a GP GPU is also foreseen for such terminal. Nevertheless the capabilities of GPU available on laptop are lower than the ones for the fixed terminal. The rendering processing on portable terminal is then simplified. The following section describes briefly the kind of application possible on portable display. This is a subset of what can be achieved on fixed terminal.

#### 3.3.1 Video rendering on portable terminal

The video rendering applied on the portable terminal is depending on the display technology. Some laptops have today auto-stereoscopic display. They will be able to display stereo content (2 views). The goal of the video rendering processing is to deliver this stereo from ROMEEO available content. As any ROMEEO terminal is receiving DVB stereo content, it is possible to display it directly after decoding. Nevertheless to address user requirement, it is possible to consider changing the depth range and/or view point as soon as additional information are available. The minimum required is the availability of disparity maps associated to the central views. In this case it is possible to render new video frame from central views and then create the expected view point.

Technology required for such viewpoint modification is similar to the one described for the fixed terminal. It is based on DIBR technology. Figure 45 illustrates the different scenarios possible to generate stereo from ROMEEO content:

- The stereo scenario does not require any DIBR processing. It is only displaying the incoming DVB content.
- The depth adjustment scenario requires disparity maps from central views to allow the interpolation of one view in between view 2 and 3. The depth of the displayed signal is adapted to the user requirement. Extreme cases are in one direction when the user requires the lowest disparity, the interpolated view is then near the incoming central view. The other direction is a range of depth just below the initial depth.
- The same scenario can be considered but with the possibility to receive to decode and to process not only central views but also satellite ones. The number of decoded views is still the same but this is not mandatory to be only central ones. Then the depth adjustment scenario can be applied to a combination of satellite and central view.

Figure 45 illustrates the different cases. More explanations about all adaptation scenario will be provided in the D3.7 deliverable due in M27.

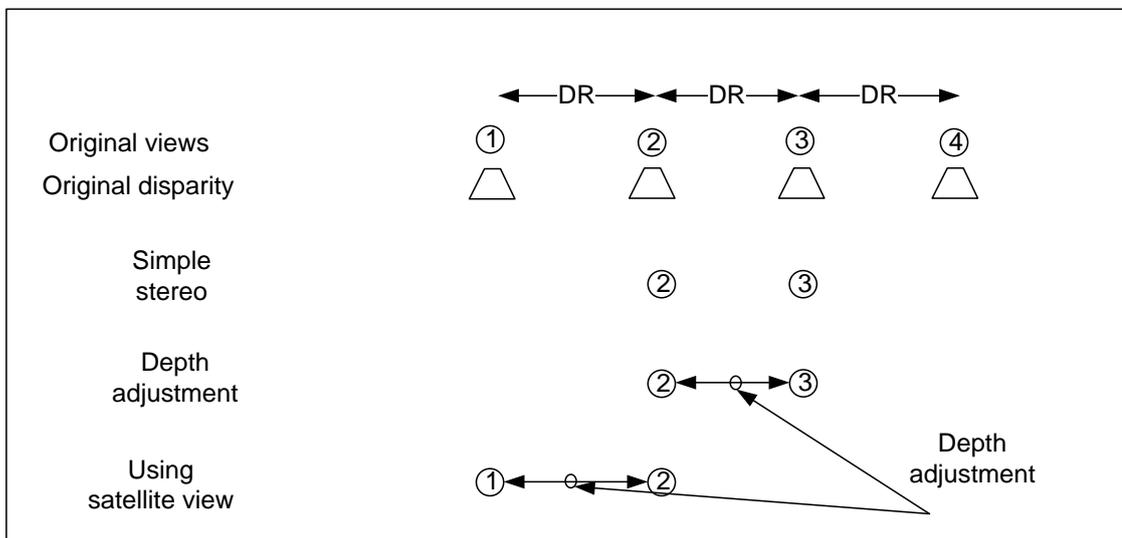


Figure 45: View interpolation for portable terminal

### 3.4 Mobile Terminal

#### 3.4.1.1 Introduction

Because the graphics subsystem of the mobile processors is not very powerful, a depth image based rendering algorithm pipeline cannot be realized in real time. That is why the Romeo system integrates video encoding formats via side-by-side sub-sampling so the devices with reduced computational power can render stereo video streams. The normal video rendering in the mobile terminal is realised in the display subsystem using OMAP4/5 hardware. An experimental implementation of an algorithm for depth image based rendering using GFX of the OMAP with reduced frame rate and resolution will be tested.

#### 3.4.1.2 Stereo rendering in Mobile Terminal Device

Mobile terminal device supports top-bottom, left-right, full-frame and sub-sampling modes. Video rendering is realized in the display subsystem, where the buffers, rotation and pixels interleaving are managed. For this purpose the assistance of the OMAP H/W is used.

### 3.4.1.3 OMAP Display Subsystem Capabilities

OMAP family processors include several HW engines to support different display devices:

- 1 Graphics pipeline (GFX)
- 3 video pipelines (VID1-VID3)
- 1 Write-back pipeline (WB)
- 2 LCD outputs (LCD1-LCD2)
- 1 TV output
- Panel support with MIPI DPI protocol

Common supported features are:

- Rotation 0, 90, 180, and 270 using DMM-TILER engine
- Synchronized buffer update
- Hardware cursor (using the graphics pipeline or one of the video pipelines)
- Independent gamma curve support on primary LCD output, secondary LCD output and TV output
- Multiple-buffer support
- Mirroring/Flip-flop support (using DMM-TILER engine)
- Programmable color phase rotation (CPR)
- Alpha blending support

### 3.4.1.4 Stereo rendering pipeline implementation

In Figure 46 is shown stereo rendering pipeline implementation on OMAP HW for complex case with top-bottom full-frame stereo format. The pipeline is realized using several blocks:

- Vid1 and Write-back - used for video rotation and scaling. The output is in the memory.
- Vid3 - send the stream to LCD1 and DSI (display serial interfaces) to Stereo LCD device

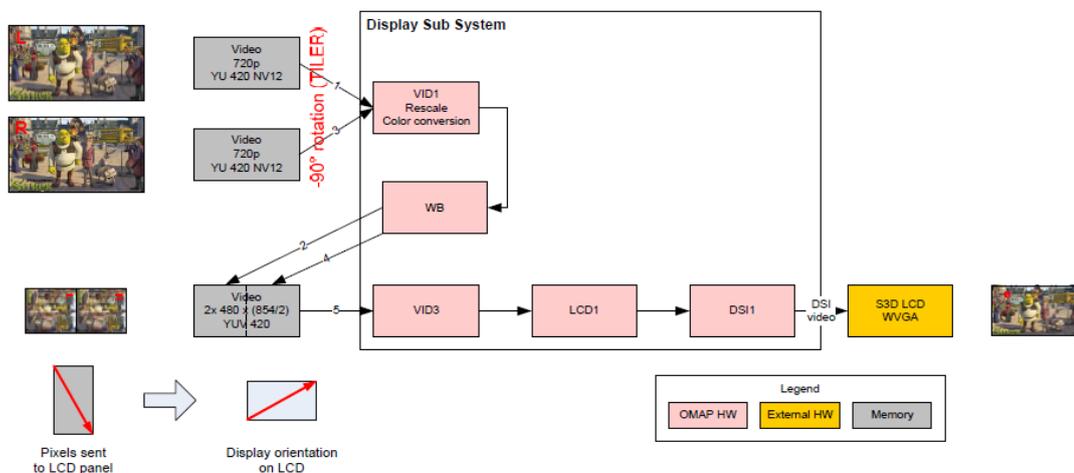


Figure 46: Stereo rendering pipeline example

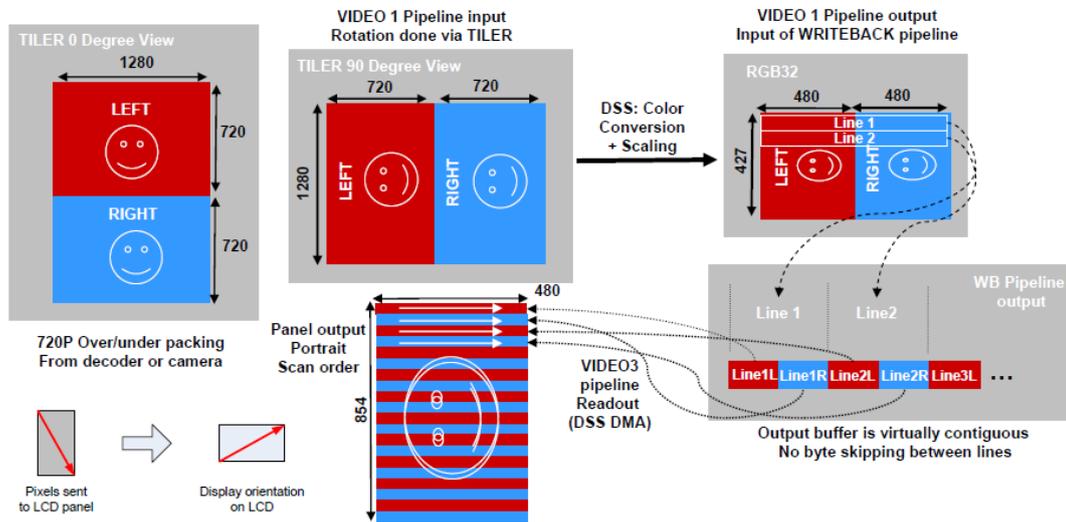


Figure 47: Stereo rendering pipeline operation

In Figure 47 is shown the diagram of pipeline operations:

Stereo Video frames 1080p/720p are in YUV420p format (12bpp) and sent alternatively each at 30 fps to VID1 path. (with/without 90deg rotation). If the case is “Top-Bottom” frames alignment and a 90-degrees rotation, the VID1 path is used.

- Video frames resized to 480 x (854/2) in YUV420p format at 30 fps.
- WVGA image in YUV420p format sent to LCD at 60 fps.
- A simplified version of the “Depth Image Based Rendering Algorithm” will be implemented on Mobile Terminal Device for experimental purposes. The algorithm will be tested with streams with reduced frame rate and sizes.

## 4 CONCLUSIONS

This deliverable describes final studies and developments made in the field of audio and video rendering algorithms. It is related to the task 3.4 of the project. This is the final version of the deliverable, an interim version has already been published (D3.3) in M16.

In this document, various audio and video rendering techniques/algorithms are introduced including the framework and the system to implement the required features. An overview of the audio and video rendering work is provided for the different terminals used in ROMEO. The algorithms and principles (state-of-the-art and beyond) are described as well as the current status of the implementation.

More specifically on the audio part, for mobile terminal, binaural room synthesis is presented as well as an implementation of this technology on an ARM processor. The use of a mobile head-tracking device is also described. Object based rendering technology is presented and the field of portable terminal.

For fixed terminal, multi-channel audio rendering as well as Wave Field Synthesis (WFS) are described. The audio rendering framework is then presented including a simplified version of the WFS.

On the video part, the basic DIBR algorithm is presented including recent algorithm improvements. The experience acquired from another European project MUSCADE is also explained. Multi-view rendering for fixed terminal is described as well as simplified version for portable terminal. The stereo rendering pipeline of the mobile terminal is also presented.

## 5 REFERENCES

- [1] J. Blauert, "Spatial Hearing, the Psychophysics of Human Sound Localization", Cambridge, MA: MIT Press, 2001
- [2] C. I. Cheng, and G. H. Wakefield, "Introduction to head-related transfer functions HRTF's: Representations of HRTF's in time, frequency, and space", J. Audio Eng. Soc., vol. 49 no. 4, pp. 231–249, 2001
- [3] F. Rumsey, "Spatial Audio", 2nd ed. Oxford, England: Focal Press, 2001
- [4] P. Mackensen, "Auditive localization head movements, an additional cue in localization", Ph.D. dissertation, Technische Universität Berlin, Berlin, Germany, Jan. 2004
- [5] U. Zolzer, "DAFX - Digital Audio Effects", 1st ed. Chichester, England: JohnWiley Ltd, 2002
- [6] D. Menzel, H. Wittek, G. Theile, and H. Fastl, "The Binaural Sky: A Virtual Headphone for Binaural Room Synthesis", In: Tonmeistersymposium, 2005
- [7] U. Felderhoff, P. Mackensen, G. Theile, "Stabilität der Lokalisation bei verfälschter Reproduktion verschiedener Merkmale der binauralen Signale" [Stability of localisation versus distorted reproduction of binaural cues], 20. Tonmeistertagung, 1998
- [8] P. Minnaar, S. Krarup Olesen, F. Christensen, et al.: "The importance of head movements for binaural room synthesis", *Proceedings of the International Conference on Auditory Display*, 2001
- [9] U. Horbach, A. Karamustafaoglu, R. Pellegrini, et al.: "Design and applications of a data-based auralization system for surround sound", in 106<sup>th</sup> AES Munich, no. 4976. AES, 1999
- [10] D. Roetenberg: "Inertial and magnetic sensing of human motion, Universiteit Twente, Diss., May, 2006
- [11] L. Klingbeil: "Entwicklung eines modularen und skalierbaren Sensorsystems zur Erfassung von Position und Orientierung bewegter Objekte", Rheinischen Friedrich-Wilhelms-Universität zu Bonn, Diss., February, 2006
- [12] W. Premerlani, P. Bizard: "Direction Cosine Matrix IMU: Theory", Draft, 2009
- [13] Potard, G: „3D-audio object oriented coding“, University of Wollongong, Diss., September 2006
- [14] Geier, M.; Ahrens, J. and Spors, S.: „Object-based audio reproduction and the audio scene description format“, *Organized Sound*, 15(3): 219-227, October 2010
- [15] G. Theile and H. Wittek, "Principles in surround recordings with height," in 130th AES Convention London, no. 8403. AES, 2011
- [16] Pulki, V.: "Virtual Sound Source Positioning Using Vector Based Amplitude Panning", J. Audio Eng. Soc., vol. 45, no. 6, 1997
- [17] Vorländer, M.: "Auralization – Fundamentals of Acoustics, Modelling, Simulation, Algorithms and Acoustic Virtual Reality", Berlin Heidelberg, Springer, 2008
- [18] S. Spors, H. Teusch, and R. Rabenstein, "High-Quality Acoustic Rendering with Wave Field Synthesis," *Vision, Modeling, and Visualization*, pp. 101-108, 2002.

- [19] A. D. Pierce, "Acoustics. An Introduction to Its Physical Principles and Applications," McGraw-Hill, Inc., 165, 1991.
- [20] E.G. Williams, "Fourier Acoustics, Sound Radiation and Nearfield Acoustical Holography," Academic Press, 1999.
- [21] D. de Vries, "Sound Reinforcement by Wave Field Synthesis: Adaption of the Synthesis Operator to the Loudspeaker Directivity Characteristics," J. Audio Eng. Soc., 44 (12), pp. 1120-1131, 1996.
- [22] J. Ahrens, R. Rabenstein, and S. Spors, "The Theory of Wave Field Synthesis Revisited," Audio Engineering Society Convention, 124, 2008.
- [23] R. B. Randall, and B. Tech., "Frequency Analysis, Brüel & Kjær," Denmark, 1987.
- [24] G. Theiler, and H. Wittek, "Wave field synthesis: A promising spatial audio rendering concept," Acoust. Sci. & Tech. 25, 6, 393-399, 2004.
- [25] J.-J. Sonke, "Variable acoustics by wave field synthesis," Thela Thesis, Amsterdam, Netherlands, ISBN 90-9014138-3, 2000.
- [26] S. Spors, A. Kuntz, and R. Rabenstein, "An approach to listening room compensation with wave field synthesis," Proc. AES 24th Int. Conf., pp. 70–82, 2003.
- [27] E. Corteel, and R. Nicol, "Listening room compensation for wave field synthesis. What can be done?," AES 23rd Int. Conf., Paper 7, 2003.
- [28] B. Günel, H. Hacıhabiboglu, and A.M. Kondoç, "Intensity vector direction exploitation for exhaustive blind source separation of convolutive mixtures," Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference, pp.41-44, April 2009.
- [29] S. Spors, R. Rabenstein, and J. Ahrens, "The theory of wave field synthesis revisited," 124th AES Convention, Amsterdam, The Netherlands 2008.
- [30] T. Balogh, "The HoloVizio System," Proc. SPIE 6055, (2006)
- [31] C. Fehn, "Depth-image-based rendering (DIBR), compression, and transmission for a new approach on 3D-TV", Proc. SPIE 5291, Stereoscopic Displays and Virtual Reality Systems XI, 93 (May 21, 2004)
- [32] P. Robert, C. Thébault, P.H. Conze, "Disparity-compensated view synthesis for S3D content correction", Proc. SPIE 8288, Stereoscopic Displays and Applications, XXIII, 82882F (February 6, 2012)

## APPENDIX A: GLOSSARY OF ABBREVIATIONS

<b>A</b>	
ADAT	Alesis Digital Audio Tape protocol
<b>B</b>	
BMP	Bitmap
<b>C</b>	
CA	Consortium Agreement
<b>D</b>	
DoW	Description of Work
DOF	Degrees of Freedom
DSP	Digital Signal Processing
DCM	Direction Cosine Matrix
<b>E</b>	
EC	European Commission
<b>G</b>	
GA	Grant Agreement
GM	General Meeting
GP-GPU	General Purpose Graphical Processing Unit
<b>H</b>	
HDSP	Hammerfall DSP
<b>I</b>	
IMU	Inertial Measurement Unit
<b>J</b>	
JPEG	Joint Photographic Experts Group image compression
<b>M</b>	
MADI	Multi-Channel Audio Digital Interface
<b>N</b>	
NEMS	Microelectromechanical systems
<b>P</b>	
PB	Project Board
PC	Project Coordinator
PNG	Portable Network Graphics
<b>Q</b>	
QAT	Quality Assurance Team
QMR	Quarterly Management Report
<b>T</b>	
TIFF	Tagged Image File Format
<b>W</b>	
WFS	Wave Field Synthesis
WP	Work Package