DIOMEDES

SEVENTH FRAMEWORK PROGRAMME

DIstribution Of Multi-view Entertainment using content aware DElivery Systems

DIOMEDES

Grant Agreement Number: 247996

D3.6

Public report on 3D Audio / Video rendering and content adaptation

Document description	
Name of document	Report on 3D Audio / Video rendering and content adaptation
Abstract	This deliverable describes the concepts and structure of the stereo multi-view video and spatial audio rendering components within the DIOMEDES architecture. Besides the description of the rendering modules and their working principles the focus is set to the adaptation of these components to varying working conditions of the system (user parameters, data transmission parameters) controlled by an adaptation decision instance. Finally, these rendering components are evaluated regarding the rendering functionality and quality.
Document identifier	D3.6
Document class	Deliverable
Version	1.0
Author(s)	Hemantha Kodikara Arachchi, Safak Dogan, Xiyu Shi, Erhan Ekmekcioglu, Stewart Worrall (UNIS), Andreas Franck (IDMT), Christian Hartmann (IRT), Thomas Korn (IDMT), Peter Kovacs (HOL)
QAT team	Murat Tekalp (KOC), Osman Solakoglu (ARC)
Date of creation	09/01/2011
Date of last modification	10/31/2011
Status	Final
Destination	European Commission
WP number	WP3

Table of contents

TABLE OF CONTENTS **3**

INTRODUCTION **8**

1.1 PURPOSE OF THE DOCUMENT 8
1.2 SCOPE OF THE WORK AND SCENARIOS CONSIDERED 8
1.3 ACHIEVEMENTS AND STRUCTURE OF THE DOCUMENT 8

2 AUDIO-VISUAL RENDERING TECHNIQUES **9**

2.1 OVERVIEW 9
2.2 VIDEO RENDERING (INCLUDES EDITED VERSION FOR PUBLIC D3.6) 9
2.2.1 PURPOSE AND DEVELOPMENT PATH 9
2.2.2 STEREOSCOPIC RENDERER FOR EARLY INTEGRATION 9
2.2.2.1 VIDEO-AUDIO SYNCHRONIZATION 10
2.2.3 COMMUNICATION WITH THE DECODER 10
2.2.4 CONTROLLING FREE VIEWPOINT CHANGES AND BASELINE 14
2.2.5 MULTIVIEW PLUS DEPTH (MVD) RENDERER 15
2.2.6 IMAGE WARP BASED RENDERER 23
2.2.7 RENDERING FOR MULTIVIEW AND LIGHT-FIELD DISPLAYS 24
2.2.8 ADAPTATION TO CHANGES IN INPUT 25
2.3 AUDIO RENDERING 26
2.3.1 Object based audio scene approach 26
2.3.2 Spatial audio production and rendering in the DIOMEDES architecture 27
2.3.3 Spatial Audio Rendering Principles 29
2.3.4 Real-time Audio Rendering Implementation: Modular Software Framework 32
2.3.5 Audio Reproduction Setups During Development and Experiments 37
2.3.6 Generation of Channel Based Audio Streams: Audio Scene Downmix 39
2.3.7 Rendering of channel based audio formats (Upmix) 42

3 AUDIO-VISUAL ADAPTATION TECHNIQUES **43**

3.1 OVERVIEW 43
3.2 AUDIO-VISUAL ADAPTATION 43
3.2.1 ADAPTATION DECISION 43
3.2.1.1 ADE MODULE OPERATION AT THE USER TERMINAL INITIALISATION PHASE 43
3.2.1.2 ADE MODULE OPERATION WHEN A CHANGE IN CONTEXT IS DETECTED 44
3.2.1.3 FITNESS CRITERION FOR REAL CAMERA VIEWS TO SYNTHESISE VIRTUAL VIEWS 45
3.2.2 VIDEO ADAPTATION 47
3.2.3 AUDIO ADAPTATION 48
3.2.4 Adaptation to different reproduction systems 48
3.2.5 Adaptation to different reproduction configurations (listener & display setups) 49
3.2.6 Adaptation to varying video viewpoint of 3D multiview video renderer 50
3.2.7 Adaptation to transmission channel properties 51

4 FUNCTIONALITY AND QUALITY EVALUATION **53**

4.1 OVERVIEW 53
4.2 EVALUATION OF FUNCTIONALITY 53
4.2.1 Video Rendering Functionality 53
4.2.2 Audio Rendering Functionality 54
4.3 QUALITY EVALUATION 55
4.3.1 Visual Quality Evaluation based on viewer tests and Quality of Experience Model 55
4.3.2 Audio Quality Evaluation 55

5 SUMMARY **59**

6 REFERENCES **60**

List of Figures

Figure 1: Line interleaved stereoscopic output from the stereoscopic renderer 10

Figure 2: Visible artefacts of the naive point cloud algorithm 16

Figure 3: Reprojection to original image 16

Figure 4: Crack and holes appearing when moving away from the original camera position. 17

Figure 5: Classification of image pixels into edge and non-edge areas 18

Figure 6: Image after filling the gaps with triangles 18

Figure 7: Image after filling gaps with triangles having adjusted areas 19

Figure 8: From an extreme angle, the individual points are clearly visible. Filling the place in-between results in false geometry added to the scene. Filling the space with bigger points gives a more pleasing result. 20

Figure 9: The effect of point splatting shown on an ordinary camera position 20

Figure 10: Multiple instances of the persons appearing due to imprecision in the input data 21

Figure 11: Turning on camera priority based rendering, single instances appear, with minor amount of erroneous pixels. 22

Figure 12: GPU framework for the rendering algorithm 23

Figure 13: Image generated by the image warping algorithm 24

Figure 14: Object based audio scene production, downmix, packaging, transmission and rendering in the DIOMEDES architecture 28

Figure 15: Screenshot of an object based audio production software 30

Figure 16: Object based audio scene production software for 3D positioning of audio objects 32

Figure 17: Example arrangement of different module types within the software framework 33

Figure 18: Example clock recovery structure (from [2]) 36

Figure 19: Audio cluster: timing and clock recovery in audio scene decoder module 37

Figure 20: 3D audio lab at Fraunhofer IDMT: Horizontal 2D audio systems 38

Figure 21: 3D audio lab at Fraunhofer IDMT: 3D low resolution setup (dome) 39

Figure 22 Functional schematic and attenuation coefficients used with the IRT downmix system by default 41

Figure 23 DIOMEDES hardware demonstrator used for the processing of the 5.1 to 2.0 downmix 42

Figure 24. Real camera viewpoints required to render the user requested stereoscopic view pair 44

Figure 25. The concept of quality projection from real camera location to the user requested virtual viewpoint 46

Figure 26. The average MOS results vs. the distance between the real and rendered (i.e., user requested) virtual view locations 46

Figure 27. The MOS vs. distance plot for the Band and Music sequences showing the approximated function 47

Figure 28: Example listener and display configurations: audio scene adaptation 49

Figure 29 Measurement setup in the anechoic chamber 56

Figure 30 Measurement of coherent pink noise in the anechoic chamber 57

Figure 31 Measurement of natural music in the broadcast studio 57

List of Tables

Table 1: Audio cluster and encoder modules: state of progress 35

Table 2. The priority order of the PIDs 43

Table 3. The revised priority order of the PIDs excluding DVB streams 44

Table 4. An example priority order of the PIDs assuming V1, V2 and V3 are the core camera views 45

Table 5: Overview of the transmission channels and usable audio stream types (includes estimates of the expected bitrate ranges) 51

1 Introduction

1.1 Purpose of the document

This deliverable completes the documentation of the work on stereo multi view video rendering and object based spatial audio rendering done within DIOMEDES workpackage 3. While D3.1 focused on the generation of content for later processing and evaluation, and D3.2 and D3.3 provided a detailed look at the applied principles and planned structures of video and audio processing, D3.5 and D3.6 present the structure of the rendering architecture as implemented in the DIOMEDES user terminal. The document also describes the implemented concepts of system adaptation on the video and audio side and the adaptation control by the adaptation decision engine (ADE). These sections of the document are strongly interconnected with the overall system architecture that will be finally described in D2.3. Finally, an evaluation of the functional and qualitative aspects of the rendering implementations is described.

1.2 Scope of the work and scenarios considered

The summary of the WP3 work given with this document reflects the implementation and integration of the DIOMEDES demonstrator system. The crucial core components of the rendering architecture are described. Their interconnections related to the adaptation decision process are highlighted.

1.3 Achievements and Structure of the document

This deliverable presents the results of module design for rendering modules and adaptation components. Evaluations of the components are substantiated by results of experiments and application tests of the implementations. The experimental results of quality evaluation refer to the deliverable D3.4 on quality of experience (QoE) modelling.

Chapter 2 presents the final and implemented concepts of audio and video rendering.

Chapter 3 presents the implemented mechanisms of rendering adaptation to varying system conditions.

Chapter 4 contains results of the evaluation of the DIOMEDES audio visual rendering implementations and their different aspects in quality and functionality.

2 AUDIO-VISUAL RENDERING TECHNIQUES

2.1 Overview

The DIOMEDES terminal in its proof-of-concept implementation consists of a terminal PC that coordinates media reception, processing and dispatching, user control and media rendering. The rendering of video and audio content is conducted in dedicated audio and video processing clusters that are physically separated from the terminal PC. All control and media data transfer between terminal PC and both rendering clusters will rely on socked-based communication.

The main processing operations of audio and video rendering clusters are basically independent from each other. Nevertheless, both clusters are interconnected in different ways regarding crucial system aspects:

- implementing similar temporal synchronisation approaches, to provide temporal congruency of audio-visual reproduction
- sharing common configuration data (e.g. on display dimensions, viewpoint), to provide spatial congruency of audio-visual reproduction

This chapter will describe the crucial processing aspects of video and audio rendering subsystems.

2.2 Video Rendering (Includes edited version for public D3.6)

2.2.1 Purpose and development path

The task of the video renderer is to display the incoming images and depth maps synchronized with audio rendering from the viewpoint selected by the user. The development of the renderer was done within two development branches: an MVD renderer partially described in the previous, interim version of this deliverable, and a simple stereoscopic / multiview renderer framework initially having no view generation functionality, but containing all the messaging and interfaces necessary to integrate into the prototype system. Before final integration, these two have been merged in one final renderer. This development path has been chosen to accommodate the renderer to the early stages of the integration, which did not provide depth maps yet, which are necessary for view synthesis.

2.2.2 Stereoscopic renderer for early integration

During early integration setups, only a stereoscopic pair without depth maps was transmitted on the whole chain. To enable seeing results quickly during integration on the stereoscopic display used in the consortium (JVC GD-463D10 Xpol-based display), a simple stereoscopic renderer was implemented to ease integration, into which the view generation methods were introduced later, when depth maps had become available through the chain.

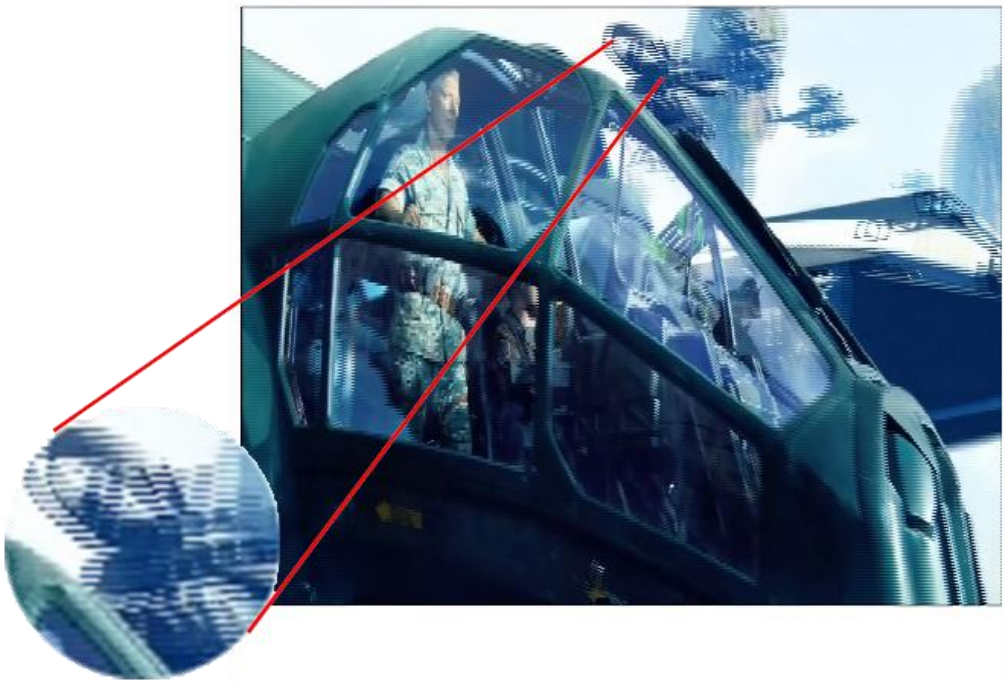

Figure 1: Line interleaved stereoscopic output from the stereoscopic renderer

This renderer is able to display the incoming streams on the selected output device (2D, line interleaved or side-by-side). The renderer is also able to show raw multiview data on an attached multiview display using a specified pixel pattern. It is able to handle all of the communication channels with other modules, and respond to every message in the specified format. To ease integration, messages that have not yet been implemented are responded with a warning JSON message to the other party. The first version of the interface supported communication with the decoder (which is feeding the renderer with uncompressed image data) and answered every other input as not-implemented. Testing communication with the decoder early was very important because we suspected that the large amount of data to be transferred between these entities could cause bandwidth issues. Once basic data transfer via shared memory had been working correctly, synchronisation of audio with video followed.

2.2.2.1 Video-audio synchronization

As the renderer is responsible for the video output of the whole system, even a slight error in the synchronization with other modules in the system is quite noticeable, thus a robust mechanism has been implemented. At the heart there is an internal clock, which can display the actual frame based on the received presentation timestamp (PTS). The clock is occasionally synchronised with the incoming PCR from the ES-Demux module, while it maintains its own PCR based on the computer's clock between these synchronizations. Thus, the renderer is robust for random amount and timing of sync signals. If only a few sync signals are arriving (for example, one every 10 seconds) the renderer is still able to maintain sync as long as the clock of the CPU is working correctly (which we can safely assume). If the renderer receives a lot of these signals (for example, more than one for each frame rendered) it can still check and correct the status of the internal clock, without causing any problems in the displaying and buffering tasks.

2.2.3 Communication with the decoder

According to initial planning, the video cluster (decoder and renderer) should run on a single computer in the stereo and multiview cases. If so, the decoder and the renderer are always running on the same PC (in one or several processes) and communication between them can be done locally. The selected method was to send image data over a shared memory channel, combined with JSON RPC over TCP/IP for signalling. This has been successfully

implemented and tested with a single decoder instance. Later, results of speed test with more views and full resolution revealed that multiple decoder instances were necessary, and to achieve the specified decoding speed, multiple decoder computers were needed, which put us in quite a different situation, and thus shared memory could be used (or just partially, if parts of the decoding are done on the rendering computer).

Quick calculations showed that the common Gigabit Ethernet interface was far from being sufficient to transmit several uncompressed HD views between computers. Thus, using a faster interconnect was necessary, and we have chosen InfiniBand, because one of the partners had already had some equipment for testing, specifically InfiniHost single-channel Single Data Rate mem-free 10Gb adapters. Performance tests shown that using native InfiniBand protocols, it was fast enough to transmit the data in real-time, the cards (called Host Channel Adapters in InfiniBand terminology) required only a free PCI-Express slot in the computers and the installation of the software stack. If more bandwidth is required, replacing the existing HCAs to Double Data Rate (20Gb) or Quad Data Rate (40Gb) HCAs, or dual-port units is straightforward.

The software stack provides multiple protocols, three of which have been considered as suitable for our needs. The Reliable Connection (RC) protocol is basically the equivalent of a TCP channel, providing a bidirectional, reliable channel between two processes, but unlike in the Ethernet case, all the protocol processing is done in hardware, thus reaching the peak performance of the hardware (approx. 7Gb on case of 10Gb HCAs, the difference comes from the overhead of the signalling on the wire and protocol overheads). The other possibility is Remote Direct Memory Access (RDMA), where one process is able to directly send the contents of a memory buffer to the memory of the other process (basically DMA between computers), which is the most similar to the shared memory approach used before. Both of these protocols require using the InfiniBand SDK, and the specific APIs in the applications, which APIs are quite complex and different from the WinSock / BSD sockets API. The third option is IP over InfiniBand (IPoIB), where the Open Fabrics Enterprise Distribution (OFED) software stack adds an extra layer over the IB channel, basically providing a virtual Ethernet / IP adapter that applications can use in the traditional way, with IP and sockets. The downside is that performance in this case is significantly lower (approximately 2Gb), which is still almost 3x increase in bandwidth compared to the practical upper limit of TCP over a GigE channel (~0.7Gb).

As the Windows API based shared memory protocol is not working over InfiniBand, a new protocol over TCP/IP (which was in turn running over IPoIB) has been implemented to handle image data transfer. This configuration temporarily solved the problem of distributing the decoding task to multiple computers. A more efficient mechanism based on one of the native IB protocols (eg. RDMA) will also be implemented to make more efficient use of the hardware and to avoid using multiple parallel channels.

Shared memory connection

Initialization is initiated by the decoder. The decoder creates named shared memory channels for each view, with size according to the image data to be transmitted (one full frame), and sends JSON RPC messages to the renderer with the parameters of the shared memory channels for each decoded view to set up the connection on the other end. The same memory buffer is reused for all successive frames for the same camera / view.

The format of the shared memory initialization message accepted by the renderer is explained through the following example:

```
{
  "id": 0,
  "jsonrpc":"2.0",
  "method": "init",
  "params":
  {
    "ConnNumber": 40,
    "ConnType":"SHR",
    "ShrMemName":"helloworldchanel",
```

```
    "ShrMemSize":3110400,
    "vID":0,
    "width":1920,
    "height":1080,
    "FPS":25
  }
}
```

The header contains a unique message id, the method name „init", while the connection number 40 specifies the message type to handle it correctly. The parameter "ConnType" selects between shared memory and TCP. The name of the shared memory channel (as used in the Windows API) is defined in "ShrMemName", followed by its size. The identifier of the memory connection is stored in "vID". The additional time invariant parameters are the resolution and "FPS" (the latter used only for asynchronous playback, synchronised playback does not use it).

When a new frame arrives over a shared memory channel, the notification message contains only "vID" and Presentation Timestamp (PTS).

TCP/IP connection

If data transfer with a remote decoder is done via TCP/IP (either over InfiniBand, GigE or locally), signalling is similar to the shared memory case. An example of the initialization JSON RPC message that arrives over the TCP/IPoIB connection is shown below.

```
{
  "id": 1234,
  "jsonrpc":"2.0",
  "method": "init",
  "params":
  {
    "ConnNumber": 40,
    "ConnType":"TCP",
    "PortNumber": 8087,
    "PacketSize":3110400,
    "vID":0,
    "width":1920,
    "height":1080,
    "FPS":25
  }
}
```

The difference is in the "ConnType" field, which is "TCP" in this case, and "PortNumber" which is a new parameter for the TCP connection. Before responding the message the renderer creates a TCP/IP server listening on the IP of the JSON server and the specified Port.

The signalling of a new frame is the same as in case of Shared memory.

```
{
  "id": 1234,
  "jsonrpc":"2.0",
  "method": "render",
  "params":
  {
    "ConnNumber": 40,
    "vID": 1,
    "PTS": ["9876", "1234"]
  }
}
```

The “vID” parameter pairs TCP channels with the views / cameras, while PTS points to the time instant when the frame should be displayed, until they are stored in the buffers of the renderer.

Metadata used by the renderer

The processing done on the incoming images can be anything from interleaving and displaying to the generation of new views. As the renderer can handle an arbitrary number of incoming and outgoing views, as well can do arbitrary compositing of the views for a specific multiview display, information about the cameras, views and the display is necessary. When rendering for HoloVizio displays, the type and serial number of the display also needs to be known (due to unique calibration information).

Input stream description

The information describing the cameras is provided by the Decoder Control Module, and sent via JSON RPC. The format of the message is as follows:

```
{
  "id": 1234,
  "jsonrpc":"2.0",
  "method":"inputsetup",
  "params":
  {
    "ConnNumber": 50
    "PTS": [0,0],
    "ChannelCount": 5
    "ChannelConfig":
    [
    {
      "ID":0
      "Type": "colour"
      "Camera":
      {
          "Intrinsic": [906.529,0,624.000, 0,906.529,344.000,  0,0,1,]
          "Extrinsic": [1,0,0, 0.2,0,1,0,-0.0157962, 0,0,1,-0.0334732]
          "Near": 3857.570
          "Far": 10072.544
      }
      "IDofC":2
    }... {"ID"...}
    ]
  }
}
```

For each channel, the renderer needs a view ID which is unique and is different for depth and colour images too. For such a view, we need the precise camera calibration information (described as intrinsic and extrinsic matrices as well as near and far clip planes), and the relation to the other channels (which image belongs to which depth map).

The parameters in this message define one or more input channels. The renderer can handle updates to this message, according to changing camera configuration over time. Configuration / reconfiguration either takes place immediately after the reception of the message, or at a specified PTS. The PTS parameter can thus be used to signal camera changes / movements to be done very precisely, regardless of the actual streaming / buffering status of the whole system.

Output configuration

The output of the renderer can be anything from a 2D window to a full screen composite image for a stereo / multiview display, or light field content to be shown on a HoloVizio display. The desired outgoing views of the system can be configured separately by defining virtual

cameras, or defined as a simple bypass of the incoming views. This definition is sent via JSON RCP as the example below shows:

```
{
  "id": 1234,
  "jsonrpc":"2.0",
  "method": "outputsetup",
  "params":
  {
    "ConnNumber": 51,
    "PTS": [0,0],
    "RenderMode":
    {
      "Mode":"stereo",
      "WindowPos": [1024,0,1024,768],
      "StereoMode": "sbs"
      "DisplayModel":"HV80C-20050001"
      "CameraSystem":
      [
      {
        "Intrinsic": [906.529,0,624.000, 0,906.529,344.000, 0,0,1,]
        "Extrinsic": [1,0,0,0.2, 0,1,0, -0.0157962,0,0,1,-0.0334732]
        "Near": 3857.570
        "Far": 10072.544
      },
      {"ID":0 },
      ...
      ]
    }
  }
}
```

Render mode defines the type of the output device which can be 2D, stereo, multiview and light-field. The position and size of the rendering window have to be specified, depending on the number of attached monitors, the resolution of each, and their layout. In case of light field display there is no window, the layout is determined by the Display Model.

Stereoscopic displays accepting side-by-side or line interleaved input are supported, the desired output type is specified with the StereoMode field.

If the requested view is the same as one of the input streams (pass-through), the ID specifies the input channel to be used. If new views are to be synthesized, the virtual camera representing this new view is defined in the same way as the input cameras (intrinsic and extrinsic matrices). Specification of outgoing views as a floating point number with respect to the incoming cameras is also possible (for example, view 1.5 is halfway between incoming view 1 and 2), provided that the number of cameras does not change, and all the cameras are perfectly rectified and equidistant. Applying this simplification in the message format will be considered later.

2.2.4 Controlling free viewpoint changes and baseline

As described above, the renderer is able to synthesize new views based on colour, depth and camera information. This can be used to allow free navigation in the 3D space, synthesizing arbitrary views (at least in theory). This navigation feature will be exposed to the user by means of moving left and right in the scene, inside the baseline captured by the cameras. This functionality is provided irrespective of the display mode, that is, navigation is possible in 2D mode as well as 3D modes requiring multiple output views. These movements are translated into specific virtual camera positions, and sent as intrinsic / extrinsic matrices to the renderer as output specification updates, using the JSON message described above. Although a simpler message that describes the outgoing views as a floating point number (and to relate

all other views to this central view) is possible, this description gives more flexibility, as it also allows arbitrary baseline modifications. That is, in case of stereo, the user can choose the distance between the two views, and increase / decrease the amount of depth in the scene by modifying the distance (similarly in the multiview case).

During viewpoint changes, the audio rendering system will also receive the same information, and update the location of sound sources accordingly, to provide a consistent audiovisual experience to viewers.

2.2.5 MultiView plus Depth (MVD) renderer

View synthesis / rendering with depth maps using DIBR

We have initially chosen a point-based MVD renderer, the approach of which is quite different from the commonly employed image warping approach. The reasons for doing so are twofold. First, it is much more suitable for GPU based implementation (most image warp renderers are still CPU based), and we were sure that the performance targets set can only be met with a GPU-based implementation. Secondly, such a renderer can take into account arbitrary nonlinear transformations during the view generation process, which is very useful for direct light-field generation. Generating the light-field geometry in one step avoids the need to make the light-field conversion step afterwards, which involves combining pixels from all of the different views based on a huge lookup table. The operation is itself simple, but requires that all views are available in all rendering nodes of the light-field display, the consequence of which is either huge redundancy and performance drop (generate all the images everywhere), or a huge communication effort between nodes (N to N with large amounts of data).

The basic concept is that the depth and frame buffers are treated as a special representation of 3D points. The renderer projects back all pixels of the camera images based on the camera parameters to world space and creates a point cloud or mesh based on the disparity image. After the inverse camera projection, the points or the mesh are projected back from world space to camera space with the new projection matrix. Visibility is resolved by writing the depth calculated from the disparity images to the z-buffer.

There are several advantages of this approach. It is very fast and can handle non-linear camera arrangements. We are also able to do a fast reprojection of the geometry on light field displays. However this approach requires a fairly precise depth representation as errors in the depth map immediately show up.

Unfortunately disparity map values generated from colour image pairs are based on matching similarities on the two images, e.g. colour differences, texture patterns or colour gradients. As these result in the closest match of pixels rather than the closest match of real depth, significant errors can occur on the depth map generated from the disparity map. Typical errors and their consequences are:

- Due to the different rasterization of the two views, black pixels appear on the target where no points of the source images are projected
- Due to the noise of the depth map flying pixels appear
- Due to the sampling and quantization of the depth map the foreground colours appear as background (and vice versa) at the edges of the objects (ghosting effect)

Figure 2: Visible artefacts of the naive point cloud algorithm

Even reprojection to the original camera position can yield minor errors due to the inverse projection projecting pixels to the wrong position in world space.

Figure 3: Reprojection to original image

These errors manifest on the reconstructed image either as small cracks or larger holes, the more the camera position is different from the originals, the bigger they become.

Figure 4: Crack and holes appearing when moving away from the original camera position.

Switching to a mesh based representation covers all in-image holes, but adding quads from another colour and depth pair results in overlapping geometry. This results either in z-fighting, or on extrapolated areas large areas intersecting with different interpolated colours.

This algorithm can be improved in several ways, as described below.

Removing the outliers and ghosting effects

Using simple image filtering techniques based on the generated depth map the outliers can be removed [1]. A small kernel is placed over each pixel and per-pixel classification is generated. It is defined for each pixel if they are background or foreground depending on the jumps in the resulting depth values.

Multiple classifications exist, such as edge detections using the Sobel or Canny methods; or using some voting technique depending on the depth values in the kernel. At the moment the most suitable solution is the latter one, where an average of the valid depth values are calculated first, then the depth relation is considered within the kernel. If the number of depth values larger than the average is bigger than its counterpart background is detected.

[1] Christoph Fehn, Depth-Image-Based Rendering (DIBR), Compression and Transmission for a New Approach on 3D-TV; Fraunhofer-Institut fÄur Nachrichtentechnik, Heinrich-Hertz-Institut (HHI), Einsteinufer 37, 10587 Berlin, Germany

[2] A Comparative Analysis of Image Inpainting Techniques, Michael E. Tächler, 2006.

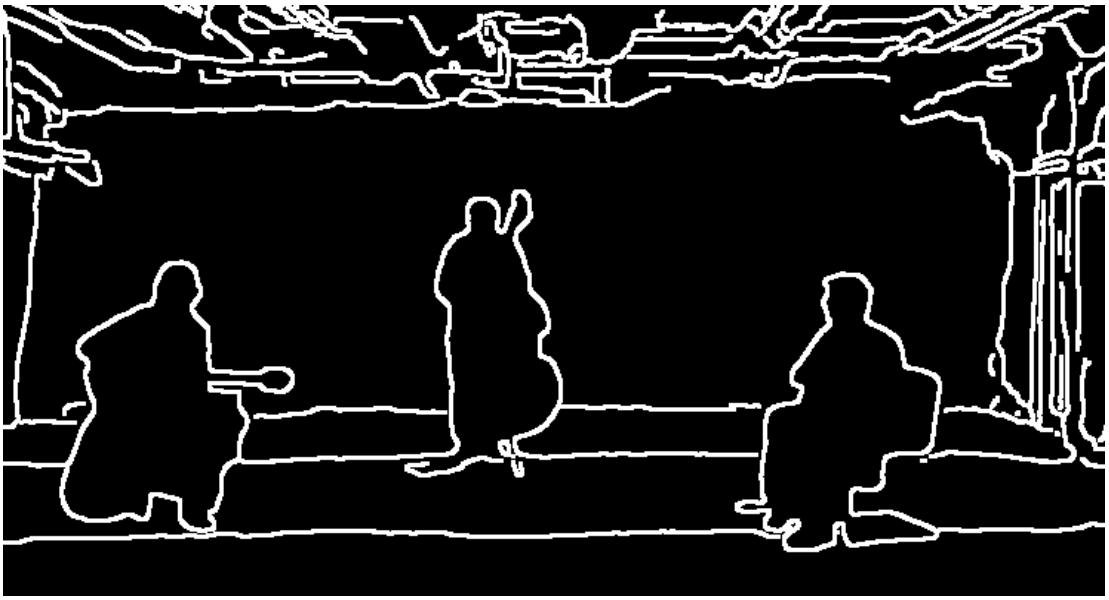

Figure 5: Classification of image pixels into edge and non-edge areas

After the classification is generated, the layers are merged to compose a single image for the current process camera view.

Filling the gaps with geometry

Filling gaps in a way that fits well for the architecture of the GPU is to use the incoming images as patches instead of standalone points. Rendering the triangle fills missing background from the target image. Using only brute force patch rendering the depth values from the other images might be treated inconsistently resulting in strange blending of background and foreground.

Figure 6: Image after filling the gaps with triangles

These effects can be almost completely removed if the sizes of these triangles are also considered. As for highly sampled parts of the image the triangles are small and the size

increases as we have less information, the area of the projected triangles (with respect to the whole image) is used as a blending (filtering) factor.

Figure 7: Image after filling gaps with triangles having adjusted areas

Filling the gaps with point splatting

Looking at the synthesized images when the virtual camera position is between the capture camera baseline, we can realize that the holes to be filled are quite small, thus, if the points of the point cloud would be slightly bigger, we could easily fill them. On the GPU, outputting bigger pixels is a simple operation, thus we can – depending on the position and orientation of the source and destination cameras – determine the desired size of the point to be drawn. According to our experiments, this results in better overall image quality compared to the geometry filling approach, which introduces false geometry between large depth discontinuities.

The effect of this improvement is demonstrated on two examples, one from an extreme camera position (far away from the capture cameras, from an angle that the viewer will not be allowed to see), and one “ordinary” camera position.

Figure 8: From an extreme angle, the individual points are clearly visible. Filling the place in-between results in false geometry added to the scene. Filling the space with bigger points gives a more pleasing result.

Figure 9: The effect of point splatting shown on an ordinary camera position

Priority-based rendering

In case camera calibration information is not precise, then feeding all the incoming images into the process described above leads to doubled / tripled objects in the scene. This happens if the pixels coming from the other cameras have a depth value telling that the pixels are in front of the ones already rendered, which often happens.

Figure 10: Multiple instances of the persons appearing due to imprecision in the input data

Using information from the closest (most reliable) camera first, and using other information from all other cameras stepping further from this camera ensures that (supposedly) valid pixels are not overwritten with other pixels coming from distant cameras. Instead, information from distant cameras is taken into account only in holes (where no information is available). Due to this change, which can be implemented using alpha / stencil testing on the GPU combined with multiple rendering passes using the different input images, the worst that can appear is some faulty pixels inside the holes, instead of complete replicas of objects adjacent with the originals. However, if the incoming data's camera calibration information is not consistent (vertically), then small jumps can occur when changing between cameras.

Figure 11: Turning on camera priority based rendering, single instances appear, with minor amount of erroneous pixels.

Real time video rendering with depth maps

These methods are combined into a single framework. Each incoming frame goes through a pipeline to enhance the target image. In each step the result from the previous camera image is updated by the new source image according to the following steps:

- The incoming camera frame is rendered to both depth and frame buffer. During this step the special projection characteristics of the display can be considered. The input camera calibration data is also used in this step.
- The depth buffer is blurred to find the average values for each pixel.
- The depth is classified and the blurred foreground and background images are generated. This step is carried out in a single pass using multiple render targets to reduce the CPU load.
- The multiple passes and pipeline elements communicate through OpenGL textures and off-screen rendering buffers.

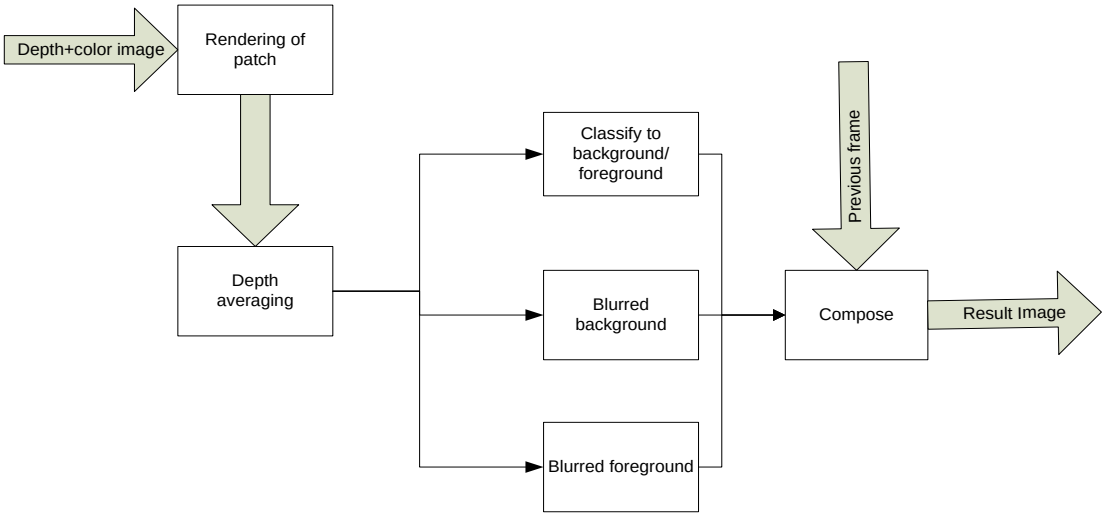

Figure 12: GPU framework for the rendering algorithm

The framework is implemented with OpenGL using GLSL programs. The result is calculated in multiple steps and to reduce the number of passes multiple render targets are used. During the patch rendering and triangle size calculation the use of geometry shader improves performance.

2.2.6 Image warp based renderer

The DIBR approach described above is quite sensitive to depth map errors and the precision of camera calibration information, it can even happen that when the virtual camera placed to the same position where one of the capture cameras is located, errors from projections done with erroneous depth values result in an image that's not the same as the original.

For this reason, we have implemented a different algorithm too. The image warp approach utilizes the disparity map for interpolation and extrapolation. It shifts the pixels from the original image with the normalized distance between the camera position on the baseline and the reconstructed image's camera position multiplied by the disparity value of the image. This approach has the advantage that it always results in the correct image when the original and reconstructed cameras coincide.

On the other hand, image warping only works when the cameras images are rectified, parallel, and there are no vertical shifts between the camera images, which is not always the case with live content. For this reason, we currently prefer using the point-based renderer.

We have implemented the image warping based renderer on GPU to achieve better performance than what is possible using CPU based code. Although this method is not really suitable for such massively parallel implementation, most of the steps can still be expressed this way.

Figure 13: Image generated by the image warping algorithm

Once we generated the images with image warping, those can be used directly with stereo / multiview displays (after mixing the pixels in the way determined by the display). However, in the HoloVizio case, image generation needs to be follow by the light-field conversion step.

Disparity map error filtering

Disparity values typically have some errors when there are large differences between disparity values (edges). Therefore it is a typical image enhancement method to generate images by removing the boundary pixels of the disparity image, which is done in a similar way than in the other algorithm. As a first step an edge detector is being run on the disparity image. This generates a mask that describes the boundary. Then a low pass and a high pass filter are run to get the foreground and the background of the image. Rendering the three layers separately enables us to run different post processing steps on different layers. While it is possible to try handling boundary pixels separately, we have found that the best course of action is to remove them and their neighbourhood from the image altogether.

Post processing

Background post processing is usually done by either diffusing or inpainting the missing parts. There are several existing inpainting methods from simple temporal background replacement for stationary cameras to various PDE diffusion approaches, edge-based algorithms and texture search and replace methods.[2]

Foreground post processing usually consists of a low pass filter along the edges for a smoother, more natural look of foreground edges.[3]

The combined image can be filtered for point noises in the depth map. If the depth differences around a pixel exceed a threshold, it is replaced by the pixel with the median depth value.

2.2.7 Rendering for multiview and light-field displays

Rendering for multiview displays is similar to rendering for stereo displays, with some differences. First, instead of two views, we need multiple views (typically 5-9), depending on the specific display. If we assume that view synthesis is used for generating the stereo pair,

[2] A Comparative Analysis of Image Inpainting Techniques, Michael E. Tächler, 2006.

[3] Intermediate view interpolation based on multiview video plus depth for advanced 3D video systems, Smolic et Al, 2008.

then generating multiple views is just a matter of number of images generated, which of course has a performance impact. On the other hand, multiview displays show reduced resolution images to the different directions. Due to the way they are constructed, they distribute a fixed amount of pixels available in the underlying HD panel to some viewing directions, that is, if a multiview display provides 5 views, then the number of pixels shown in one direction is be approximately full resolution / 5 pixels. The resolution loss is usually evenly distributed between the horizontal and vertical resolution (using slanted lens arrays), but knowing the pixel structure, the effective resolution can be approximated.

This takes us to the other difference, pixel structures. In stereo displays, the possible pixel arrangements are side-by-side, over-under (not commonly used), line interleaved and checkerboard (not commonly used either). Most display accept more of these formats (although some filtering / upsampling may occur if the non-native format of the display is used), and all of these are well known and easy to assemble in software. This is not the case with multiview displays, where the pixel / subpixel structure of the views is usually kept secret by the display manufacturer, and users can only use their proprietary tools to create content for the display. One exception is Alioscopy, who solve this problem by providing a GPU shader that does the magic inside, that application developers can build in their software. Other than this, developers / researchers are commonly forced to reverse engineer the subpixel structure [4].

In this project, we have received the pixel pattern of two multiview displays. One is a display developed by Arcelik, the other one is the NewSight 42" 8-view display.

The most straightforward way of representing such subpixel patterns is in a texture, having equal size with the underlying panel and the view numbers represented by RGB values in the texture. Such a texture can be generated programmatically, still it provides enough flexibility to take any kind of structure or irregularity into account. Then, this texture in the renderer is used as a look-up table during the assembly of the final multi-view image.

2.2.8 Adaptation to changes in input

As the basic layer arriving via DVB is a stereoscopic image pair, and all others views and all the depth maps arrive via P2P, the minimum scenario is that the renderer has stereoscopic input without depth maps. The most optimistic case is that the renderer is fed with 4 views and 4 depth maps (or even 8 views and 8 depth maps). In between, according to the bandwidth available to P2P, any combination is possible, and the situation may change during the transmission.

In theory, the possible degradations possible are:

- missing views
- missing depth maps
- partial views or depth maps
- different spatial or temporal resolution of views and / or depth maps (all changing at the same time or change independently from each other)
- different quantization of views and/or depth maps (lower bit depth)
- different compression levels of views / depth maps

Loss of views or depth maps can be handled by the renderer, as the number and layout of the cameras can be changed on the fly, even with every frame. However, due to the way the images are used during the rendering process, losing an image that is close to the virtual viewing position may result in an abrupt change (jump) in the rendered image, if the data coming from the adjacent views (which will be the closest ones after losing the current one)

[4] A. Boev, R. Bregovic, A. Gotchev, "Measuring and modeling per-element angular visibility in multiview displays", Special issue on 3D displays, Journal of Society for Information Display, Sept. 2010 Vol. 26, No. 09, pp. 686–697

contain slightly different image / depth information about the scene. Fortunately, the renderer contains a buffer of frames ahead, thus if the loss of a frame is signalled in time, the renderer can gracefully change from using the will-be-lost camera to an adjacent one. That is, the contribution of the image coming from the camera can be reduced to zero by the time it will be lost, and then this change can be done through multiple frames (if the size of the buffer allows), thus the transition from dominantly using one camera to dominantly using another adjacent camera can be smooth.

Similarly, when a view and the corresponding depth map is re-introduced (and the new camera is closer to the virtual view than the others), such a transition can be done in the opposite way, smoothly transitioning to rely on the new view.

Due to the way it operates, if the renderer faces views without corresponding depth maps, or depth maps without corresponding views, it cannot use this partial information for view synthesis, so the partial information will be dropped (unless the view is used in pass-through mode, when a depth map is not needed).

Different spatial resolution could be handled by the renderer (by using the respective textures upsampled), but this also needs updating the signalling between the decoder and the renderer, so that the decoder can indicate the reduced resolution of the image which is placed in the (partially used) shared memory / network buffer.

Different temporal resolution is the kind of degradation with which the renderer cannot do much, as interpolating the lost frame based on motion vector prediction is far out of scope. Thus, two options are using the previous available frame until the next one arrives, or, if this produces disturbing effects (parts of the scene are not updated with the same frequency as the others), we can avoid using that image.

Different quantization and / or compression levels of video and depth streams results in lower precision of colour and depth values, the latter of which can cause hard edges and discrete depth levels in the scene. Well-known compression artefacts in the images are also have an effect on the depth maps, which is manifested as false depth values on the sides of macroblocks, and incorrect depth values around sharp edges. Edges in the depth map are already treated separately, however we are not aware of any way avoiding the effect of blocking artefacts in the depth map.

An alternative way of handling drastic losses in image streams is forced animation of the user's viewpoint. For example, in the worst case of falling back from MVD to a pure stereo pair, if the renderer can foresee such a loss, it can smoothly animate the viewpoint back to the central position if the stereo pair (if it was not there already), and when it's there, switch to using pass-through of the two images onto the stereoscopic display instead of view synthesis. Such an automatic camera movement may not even be noticeable by the viewer in case the 3D content uses camera movements itself. When depth maps become available, the user is allowed to change viewpoints again.

According to the current state of the overall system, only losses of full images and / or depth maps are possible, and the renderer cannot face any of the other degradations described.

2.3 Audio Rendering

The following section gives the final description of the implemented audio render techniques in the DIOMEDES system architecture. The section refines and updates the interim descriptions of D3.2.

2.3.1 **Object based audio scene approach**

Compared to other established broadcast architectures, the DIOMEDES architecture extends audio transmission by the use of so called object based audio scene transmission in addition to conventional channel based transmission approaches.

Conventional channel based transmission modes are based on a mix-down of the audio production to a set of audio channels that is assigned so a set of loudspeakers of the audio reproduction setup.

The object based audio scene transmission and rendering approach applies an abstract audio scene description based on pairs of individual audio object signals that are not assigned to dedicated loudspeaker positions but to a set of object description data (e.g. audio object position). The assignment of these signals to the reproduction loudspeakers is done using the spatial audio renderer. Signal processing adapts the incoming audio object descriptions to the individual loudspeaker channels connected to the audio renderer. By using this approach, advantages for audio scene reproduction can be achieved that are described in the following sections.

2.3.2 Spatial audio production and rendering in the DIOMEDES architecture

The following block diagram shows the object based audio scene production and application of spatial audio rendering in the DIOMEDES system architecture on both broadcast and terminal sides in a schematic way. The shown blocks represent processing steps in the DIOMEDES architecture, that partly work simultaneously in realtime (production side: monitor rendering, 5.1 downmix, object based scene encoding; terminal side: all modules) or that are used as offline tools for the DIOMEDES demonstrator production (2.0 downmix, channel based encoders and MPEG-2 TS packagers, multiplexer).

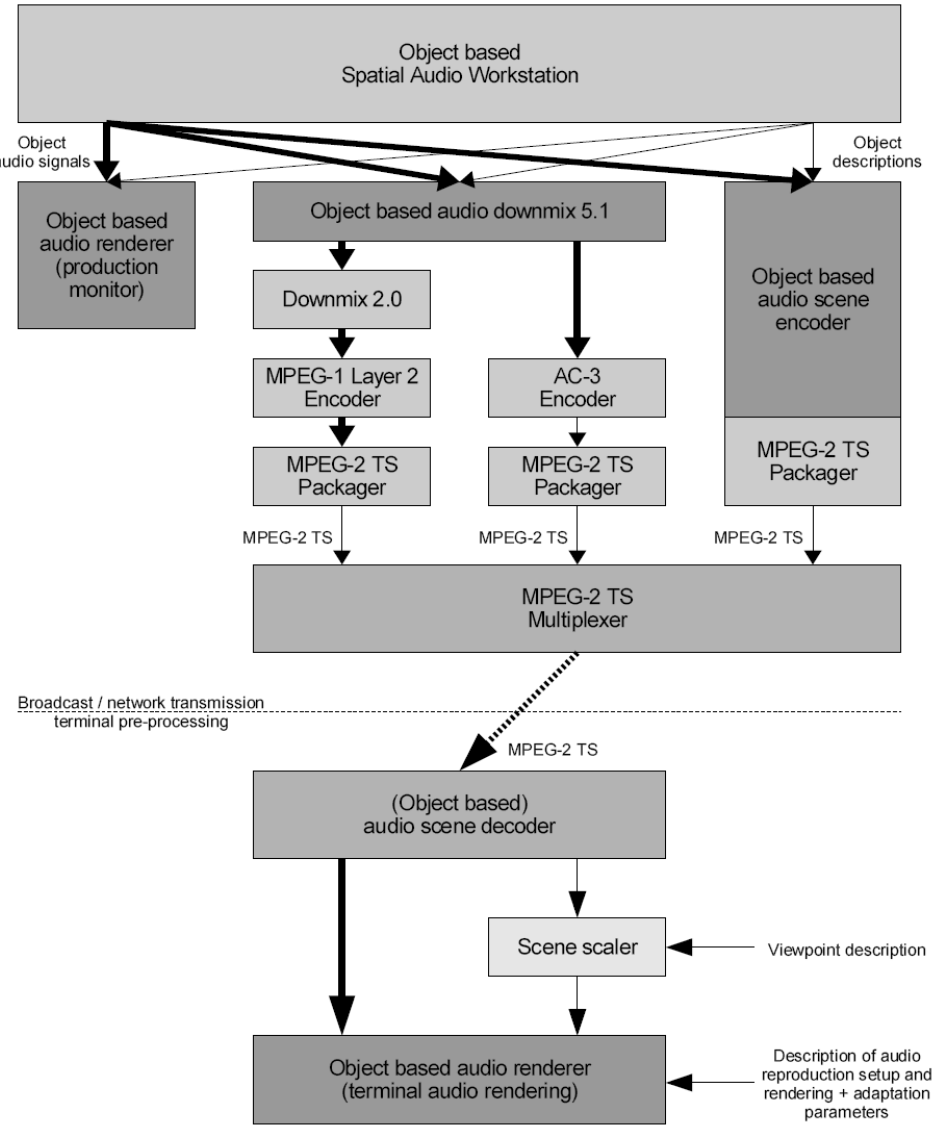

Figure 14: Object based audio scene production, downmix, packaging, transmission and rendering in the DIOMEDES architecture

Object based spatial audio rendering is used on the production side of the DIOEMDES architecture for production monitoring and on terminal side of the DIOMEDES architecture for audio reproduction.

For production, an object based audio production software, the so called spatial audio workstation, assigns audio object description data to a set of object audio channels. This step leads to the so called object based audio scene.

The audio scene is reproduced during production by a realtime spatial audio renderer that controls the production loudspeaker setup. The same audio scene is processed by the DIOMEDES spatial audio scene encoder that conducts a compression of the object audio signals and a multiplexing with the accompanying object description data. The third component of this architecture that processes the object based audio scene is an object based audio scene downmixer. It creates a set of 5.1 audio channels from the object descriptions. A 2.0 audio downmix is derived from the 5.1 downmix by a dedicated downmix module. At this stage, the object based audio scenes is complemented by two channel based scene representations that are used within the DIOMEDES structure for DVB transmission.

All 3 scene representations are coded and encapsulated into MPEG-2 Transport Streams (TS) for transmission. A multiplexer creates a TS containing all 3 audio streams. For transmission via DVB or P2P within the DIOMEDES architecture, the stream is demultiplexed and combined with synchronised video streams before broadcast or network transmission.

On the terminal side of the DIOMEDES architecture the received DVB and / or P2P streams are synchronized and remuxed to be fed to the audio and video clusters. The audio scene decoder module of the audio cluster receives the incoming audio TS and decodes the streams into object based audio scenes. The availability of the 3 scene representations in the incoming TS determines which representation will be rendered as an audio scene. Before the decoded audio scene is processed by the spatial audio renderer for reproduction, a dedicated module for object scene scaling adapts the audio scene for different video viewpoints geometrically. The object based audio renderer finally generates the audio signals for the reproduction loudspeaker setup, adapting the audio scene to the given loudspeaker number and placement, and the configured display and listener positioning.

2.3.3 **Spatial Audio Rendering Principles**

Two approaches of audio rendering were chosen for application in DIOMEDES terminal:

Wave-Field-Synthesis (WFS)

The audio reproduction principle of wave field synthesis can be implemented by individually controlling a high number of closely placed loudspeakers to achieve an approximation of real sound wave fronts. This principle is derived from the Huygens-Fresnel principle of decomposing a wave front into elementary waves [3] [5].

Reconstruction of a wave front approximation is realised by superimposing the individual wave fields of all loudspeakers. The individual speaker signals are derived from a signal of a so called virtual sound source by applying speaker-specific signal filtering, delaying, and level modification. From theoretical point of view, the reproduction of realistic 3D wave fronts is possible, if a listener area is completely enclosed by transducers (full 3D enclosure). Due to the high number of loudspeakers, that are necessary to conduct wave field synthesis, a reduction to horizontal loudspeaker arrangements is usually accepted and applied.

Several unique features of Wave Field Synthesis can be used to create and reproduce audio scenes of a new perceptual quality:

- The human localisation of virtual sound sources approximates the localisation of real sound sources. A virtual sound source object can be placed apart from positions of existing loudspeakers (that means even “behind” the loudspeaker setup, as it is seen from a listener position) and the listener will have a nearly correct directional perception of the source objects’s position. This perception will not be determined by the position of one loudspeaker but from the overall wave field characteristics. The directional perception will be nearly correct not only for one ideal listening position but for an extended area of listener positions compared to conventional surround reproduction systems (sweet area instead of sweet spot).
- The WFS approach offers the reproduction of so called focused sound sources. If a virtual sound object is placed within the enclosing loudspeaker ring and within the listener area, the resulting wave field has characteristics of level distribution and localisation that create the impression of sound source positions close to the listener. A continuous change of the source’s position within the listener area leads to aural perceptions that are not possible with conventional audio reproduction systems.

Current WFS implementations are based on a model based rendering: the generation of loudspeaker signals from virtual source signals is done by applying a set of loudspeaker driving coefficients (e.g. conducting a convolution of FIR coefficients or applying matrices of amplitude factors and signal delays) that are derived from a set of virtual source parameters

[4] [6]. Virtual source's position and type are example parameters of this set. The virtual source object description structure can vary in different WFS implementations. The object description of one virtual source directly affects the driving coefficient set.

Since a usual WFS implementation supports the simultaneous rendering of numerous virtual sound sources, a WFS audio production leads to virtual source object data sets that each consists of the source's audio signal and synchronised object description data.

Dedicated tools are used to generate this combination of audio signals and description data during a production process. Depending on the implementation of the WFS system, these tools are more or less comfortable. Basic functions of these production tools are

- positioning of sound objects,
- animation of sound objects,
- setting and automation of object properties,
- handling of multiple audio objects simultaneously.

Advanced functions of dedicated object based production tools are

- manipulations of object motions,
- hierarchical management and manipulation of multiple objects,
- cooperative editing approaches (simultaneous editing by multiple users).

These tools can be implemented e.g. as stand-alone applications or add-ons for established audio workstation software. In current implementations they control the separate audio rendering processors via socket-based communication. In using the above-mentioned tools, complex audio scenes can be created. Following figure shows an example of a graphical production tool for object based audio scenes.

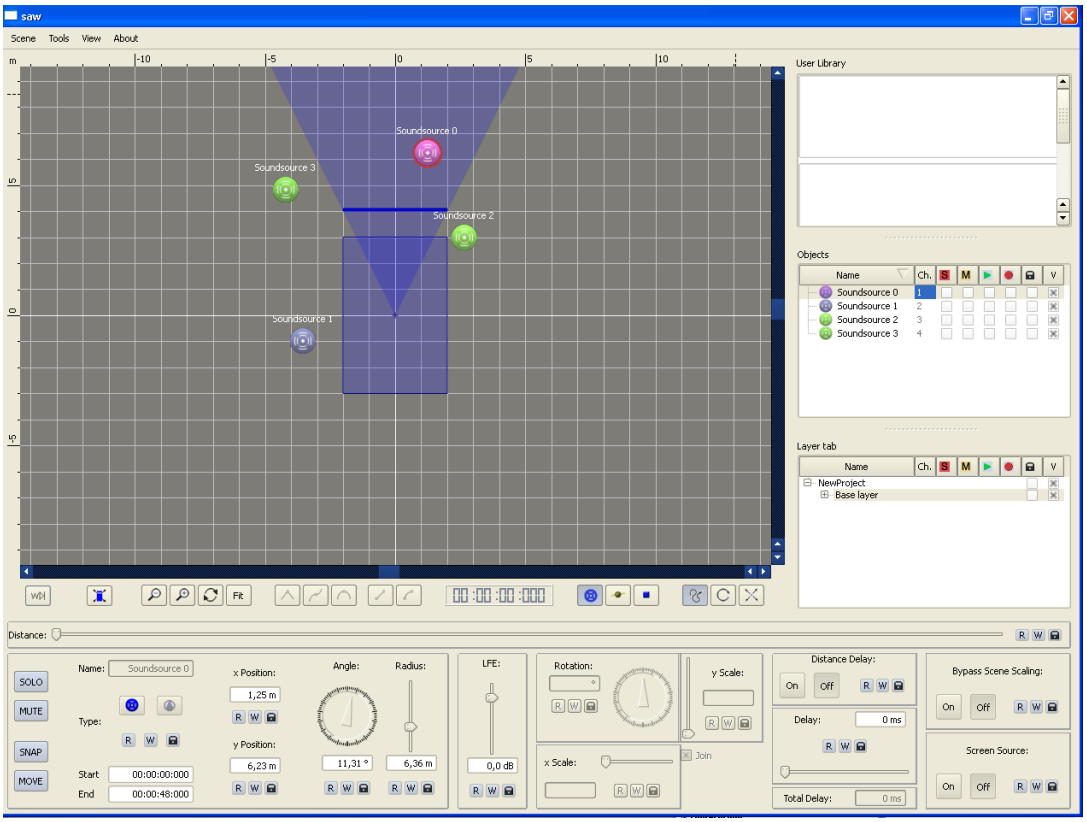

Figure 15: Screenshot of an object based audio production software

Reproduction on low-resolution and 3D loudspeaker setups

As WFS reproduction systems require a significantly higher number of loudspeakers / signal channels / amplifier and D/A-conversion modules than conventional surround systems, their application is restricted to a limited range of applications mainly due to their costs. Thus, low-resolution reproduction systems seem to be useful to allow a playback of object based audio scenes even on loudspeaker systems of lower loudspeaker numbers.

Based on experiences with current WFS rendering implementations, dedicated modules to control loudspeaker setups of low loudspeaker numbers were developed at Fraunhofer IDMT. Driving coefficient generators were implemented that cope with lower loudspeaker numbers and higher loudspeaker inter-distances compared to WFS setups. Coefficient calculation can adapt to given listener setups and loudspeaker arrangements with the aim of reproduction the same object based audio scene that originally was produced for WFS setups with an overall similar listener experience.

Driving coefficients are calculated, applying parametrically controllable models that invoke listener and loudspeaker setup descriptions. The underlying geometric models are adapted to current WFS implementations. Coefficients are generated to provide localisation clues to the listener corresponding to a given virtual source position. Although wave front reconstruction cannot be sufficiently be achieved to perform true wave field synthesis, source localisation (of reduced accuracy) is still possible by providing stimuli that can be interpreted by the human auditory directional perception system (see also [7]). At the listener position, these localisation clues lead to a directional impression that is strongly influenced by direction of arrival, level of arrival and time of arrival of the individual loudspeaker contributions.

Like the WFS reproduction systems, the low resolution reproduction systems are based on loudspeaker setups that surround the listener area.

The significant reduction of loudspeaker numbers impairs the localisation precision compared to that of a full WFS loudspeaker setup. The effect of focused sound sources is also reduced.

Control parameters include the same audio object description data set that is used to control the above-mentioned WFS rendering approaches. Due to this compatibility of object control interfaces, the same object based audio scenes can be reproduced by both approaches.

Driving coefficient calculation from source positions utilizes geometric models that were also designed to support true 3D loudspeaker arrangements. The requirement of enclosing loudspeaker setups in the 2D horizontal plane is extended here to a virtual 3D enclosure. That means that the listener area must be enclosed by a loudspeaker setup forming a volume that contains the listener area. A typical case of such a loudspeaker setup would be a ring of loudspeakers distributed in the horizontal plane and additional loudspeakers above the listener area. This setup can be controlled to approximate virtual elevated sound source positions.

The following figure shows an audio object based production software that was extended by 3D positioning of virtual audio objects. The example screenshot shows a top view of a semispherical dome setup of loudspeakers and different audio objects combined to an audio scene. To derive 3D coordinates from the 2D graphical user interface positions, object positions can be transformed from the horizontal plane by placing them onto a configurable 3D surface that is indicated as light blue area.

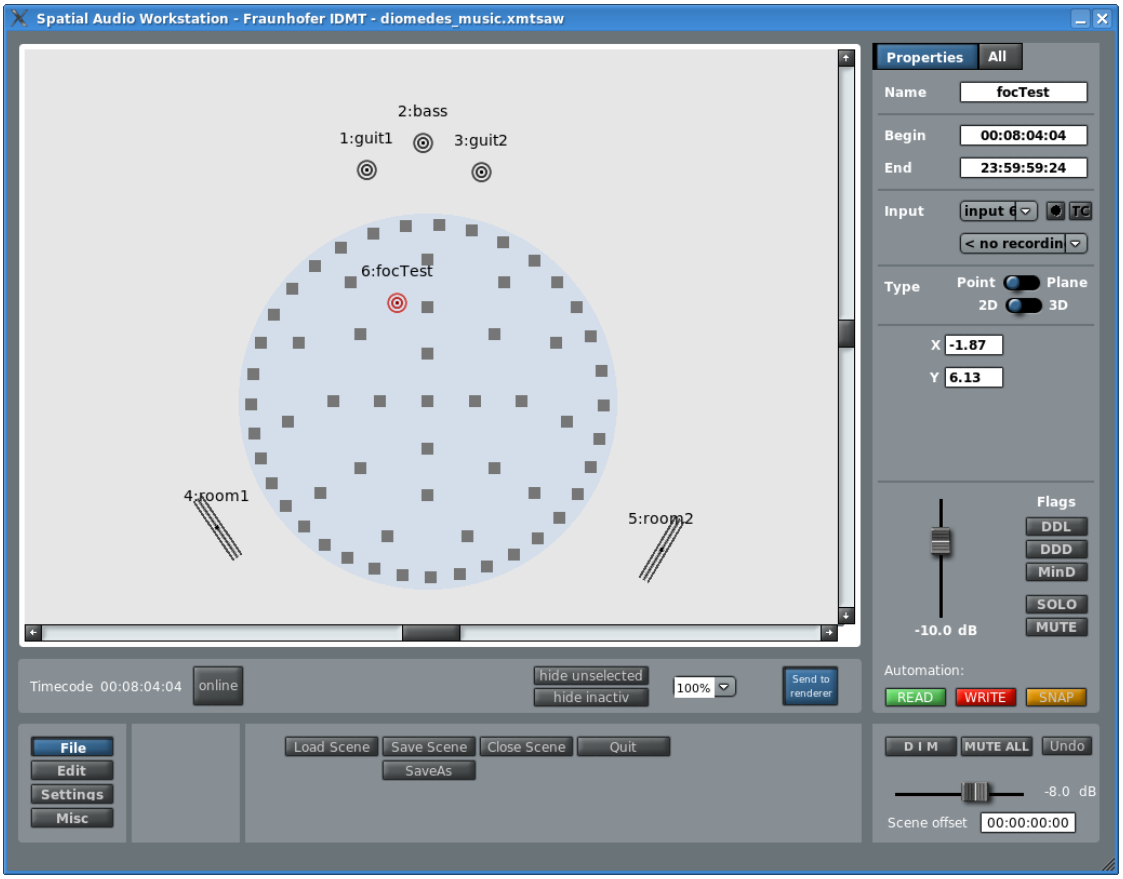

Figure 16: Object based audio scene production software for 3D positioning of audio objects

The driving coefficient calculation approaches are similar for low resolution 2D and 3D setups. In the currently implemented module, a selection of one of both approaches is conducted by evaluating a configured loudspeaker setup. As invalid loudspeaker setups are neglected, the design of a reproduction loudspeaker system should be prepared considering the described restrictions.

2.3.4 Real-time Audio Rendering Implementation: Modular Software Framework

As described above, the operation of rendering a virtual audio source with a given input signal to a loudspeaker setup consisting of a number of N loudspeakers, can generally represented by conducting N convolution operations of the input signal with N loudspeaker specific driving coefficient sets. If a number of S virtual sources should be rendered simultaneously on a reproduction system, a number of SxN convolutions have to be conducted applying individual FIR kernels for each crossing point of the resulting filter matrix.

As processing costs grow with filter length, number of rendered virtual sources and number of loudspeakers, wave field synthesis (WFS) rendering processes lead to a high occupation of available processing capacity on the processing platform. Additional processing effort is spent for a real-time driving coefficient design as it is usually applied in such reproduction systems. Finally, controlling operations and additional signal processing (e.g. audio decoding) increase the needed processing power.

The usual platform for audio rendering within the DIOMEDES architecture will be a PC system. Current PC systems are capable to handle the complete signal processing for a WFS system. To conduct rendering on such systems, the software processes have to be optimised. While the overall complexity of the rendering software processes can be reduced by adapted optimising and consolidating the processing operations, a real time rendering of complex audio scenes to complex loudspeaker setups can also be made possible, by exploiting the multi processing features of modern PC systems.

For that reason, a multi threading framework was developed specially for the application in audio reproduction systems. The design of this software framework was adapted to the needs of multi-channel audio processing, rendering and related audio operation as needed for WFS and low resolution rendering.

Basic features of this software framework are:

- support of multi-threading on multi-core/multi-processor machines
- support of modular processing system designs built from a small set of basic module classes
- standardised set of configuration and connection interfaces (message connections and audio connections)
- extensible infrastructure: implementation of new modules fulfilling specific requirements
- configurable module arrangements (signal flow graphs) and individual configurable blocks
- automatic generation of processing schedule

The following figure schematically outlines the module-type arrangement of an example configuration structure.

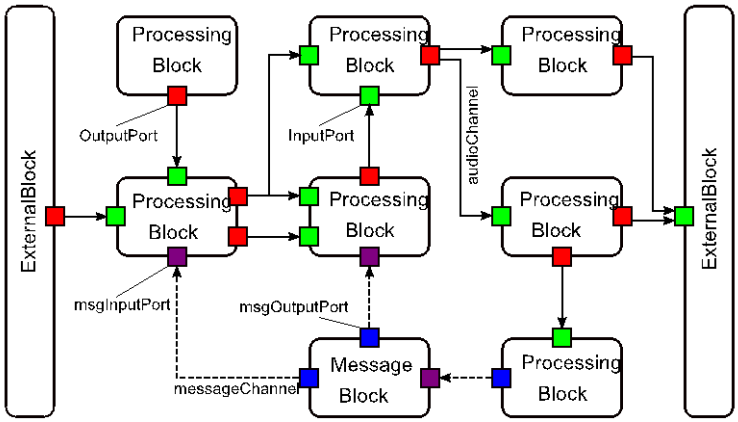

Figure 17: Example arrangement of different module types within the software framework

The module type “ExternalBlock” has the function of global audio signal input and output. This type of block can be connected with modules of the “ProcessingBlock” type. All modules of this type require audio signal connections to other modules. In typical audio rendering configurations, the convolution matrix is implemented in dedicated modules of this type. Modules of the so called “MessageBlock” type can handle and distribute non-continuous message data, e.g. control instructions. In the diagram, these modules control processing modules that are able to handle data of message type.

This modular approach allows for splitting up the overall processing task into individual operational entities. The computation needs of these modules are distributed over time and over CPUs to optimally exploit the system’s computation capacity.

In the case of WFS audio rendering, the signal processing task can be distributed to a number of processing modules, while a separate MessageBlock module conducts all driving coefficient calculation tasks. An instance of ExternalBlock encapsulates the interface to the multi-channel audio devices. Furthermore a dedicated MessageBlock module provides a socket based control interface for incoming external control instructions. Finally a processing module can conduct all decoding and timing processing that is needed to allow synchronised audio scene rendering within the DIOMEDES system.

DIOMEDES audio rendering cluster: module implementation status

The modules used within the DIOMEDES audio cluster software framework and their final state of implementation will be indicated in the following list:

Table 1: Audio cluster and encoder modules: state of progress

Module category	State of implementation
WFS coefficient calculation WFS rendering	Implemented
Low resolution 2D / 3D driving coefficient calculation Low resolution 2D / 3D rendering	Implemented
Driving coefficient convolution matrix	Implemented
Audio scene decoder for stream reception, decoding, clock recovery and synchronisation, high quality audio resampling: "NetDecoder"	Implemented
Universal audio scene decoder for file and network stream decoding "StreamPlayer", to replace "NetDecoder" module	Implementation will continue after DIOMEDES
Scene adaptation extensions (viewpoint changes)	Implemented
Real-time audio scene encoder (includes audio scene capturing, coding, packaging)	Implemented

Module example: Universal audio scene decoder "StreamPlayer"

A planned module that has entered the implementation phase is the "StreamPlayer" module. This processing module will allow the decoding of object based audio scenes. The module will have the following features:

- flexible, extensible component for playback of file- and stream-based media
- support of different types of synchronisation references (e.g. internal time reference, PCR timing information of MPEG2-TS based formats, SMPTE longitudinal time code)
- implements generic interface for buffering and decoding (support of diverse codecs shall be possible)
- interface for synchronization timers
- optional interface for file transport control

This module is under development and will replace the current "NetDecoder" module of the DIOMEDES architecture in a later stage of the audio renderer.

Audio-visual synchronisation, clock recovery, timing and signal processing strategies within the audio scene decoder

A crucial part of the DIOMEDES audio cluster structure is the audio scene decoder module which is embedded into the audio rendering software framework. It

- receives streaming data from the terminal PC,
- parses the Information contained in the MPEG-2 TS container (mainly signalling information of PMT and PAT, and timing data - PCR),
- selects the packetized elementary streams (PES) fed into the core audio scene decoder structures,
- parses the information contained in the PES packet headers (mainly timing information: PTS),

- feeds the PES payload to the audio codec component and the object description generator,
- conducts system clock recovery (to adapt the internal sampling rate conversion),
- resamples the decoded audio signals according to the recovered system clock,
- provides the audio scene data (audio signal and object description data) to the following modules of the rendering framework in a synchronized way.

Furthermore it provides following capabilities

- handle network stream adaptation (permanent check of format availability in input streams, choice of the decoded stream)
- support switching between decoded streams,

Regarding the output audio sample rate, the DIOMEDES audio cluster is a stand-alone module. It is not synchronized with other devices of the transmission chain via direct audio sample clock connections (e.g. word clock).

For this reason, the clock of the stream input system is regenerated within the audio cluster to conduct a resampling of the coded audio signal to the (typically fixed) sampling rate of the cluster's audio device. This prevents the audio cluster from clock drifts and buffer overflows/underruns. The resampling operation provides a link between the external system clock (broadcast system clock and terminal PC clock) and the cluster's audio hardware sample rate clock. This approach prevents the software structure from relying on hardware specific software interfaces that allow sample rate adjustments.

In performing a clock recovery, the decoder module follows the recommendations of standard [2]. Following figure shows the proposed clock recovery structure of this standard, as it is applied in common DVB receiver systems.

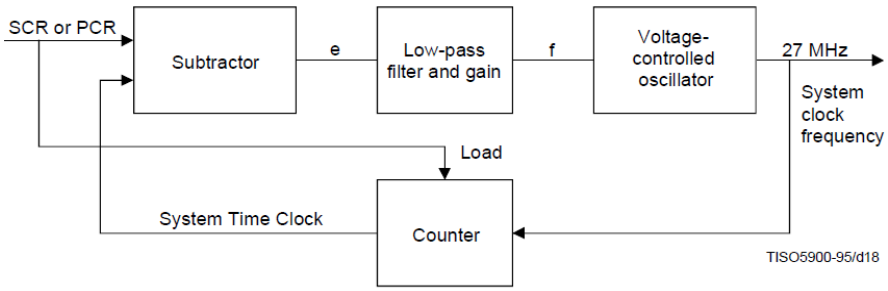

Figure 18: Example clock recovery structure (from [2])

The regenerated clock is represented by a counter that operates at a frequency generated by a VCO. The VCO frequency is adapted according to a low-pass filtered time difference signal between regenerated clock and incoming timing information that is carried by transport stream packets.

Based on the structure of this control loop, the clock recovery structure of the DIOMEDES audio cluster was derived. It has the structure shown in the following figure.

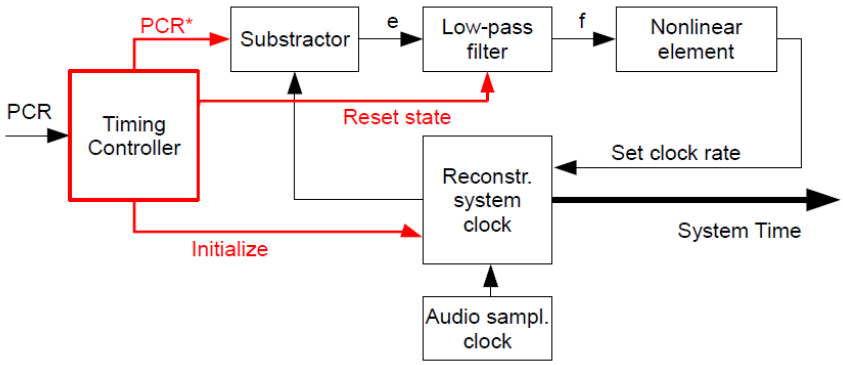

Figure 19: Audio cluster: timing and clock recovery in audio scene decoder module

Incoming PCR information is used for timing control by generating a difference signal from reconstructed clock value and latest incoming stream time stamp. Like in the standard's example structure, the difference signal is low pass filtered. After this operation, a nonlinear element derives setting values that correct the rate of the reconstructed clock to minimize the resulting time error. The design of this non-linear element has a significant influence on the duration of the convergence process, on stability of the control loop and on the resulting rate changes. Thus, its parameters are adjusted to keep resulting pitch changes due to sample rate conversion within the range of inaudibility.

The structure shows that the reconstructed system clock is derived by integrating the audio hardware sample rate. The nonlinear element delivers a correction factor affecting the integration.

Having regenerated the system clock indicated by the incoming stream's timestamp values, the audio decoder module performs a conversion of the transmit audio signal sampling rate as described above. As an object based audio scene typically is represented by a set of multiple audio object signals, the efficiency of the sample rate conversion algorithms has a significant impact on the overall computational effort of the audio rendering cluster.

Using existing methods for arbitrary sample sample rate conversion (ASRC), it is difficult to meet these efficiency requirements. For this reason, efficient algorithms for ASRC have been investigated and developed within this project. [13] considers the the use of an optimized continuous-time resampling filter, termed optimized image band attenuation design (OIB), in combination with oversampling. This approach uses an overall optimization method originally proposed in [12] that adapts the coefficients of the oversampling component to the characteristics of the resampling filter. In [14], the computational efficiency of the proposed algorithm is compared to established methods. It is shown that the OIB design achieves a significant reduction of the computational complexity.

2.3.5 **Audio Reproduction Setups During Development and Experiments**

Two main loudspeaker setups were built up at Fraunhofer IDMT for audio rendering development, demonstrator testing and conducting perception experiments in the DIOMEDES context. Both setups are part of one dome system of approximately semispherical shape. The following section describes both loudspeaker setups.

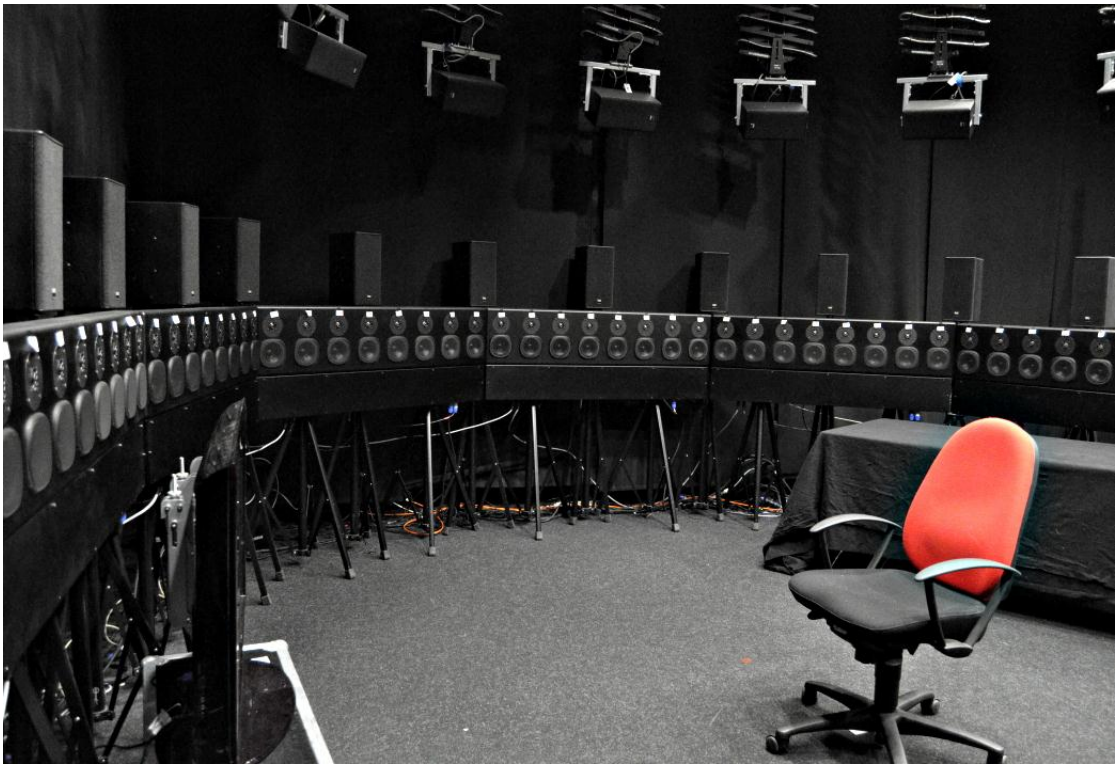

Figure 20: 3D audio lab at Fraunhofer IDMT: Horizontal 2D audio systems

Wave Field Synthesis setup

88 Loudspeakers enclose a listener area in the horizontal plane. The loudspeakers (2 band systems) are grouped to 11 loudspeaker panels by 8 loudspeakers each. The panels are specially designed for WFS reproduction and offer built-in 8-channel D/A converters and power amplifiers. All eight individually driven loudspeakers of one panel are controlled by one optical ADAT connection transmitting 8 audio signals. All 11 ADAT signal chains are fed by two optical MADI connections, each able to carry up to 64 audio signals simultaneously. One multi-CPU PC system supplies both MADI signal bundles of the WFS sub system by conducting efficient audio signal processing.

Low resolution and 3D loudspeaker setup

23 individual loudspeakers in the horizontal plane are arranged within the same ring as the WFS loudspeaker system above the WFS loudspeaker panels. Additional 29 loudspeakers are arranged above both loudspeaker rings and form a dome of approximately semispheric shape. The 52 individual ring and dome loudspeakers are passive 2-band loudspeaker systems of higher quality (model: Kling & Freitag “CA106”) than the panel loudspeakers. A power amplifier array and a 64 channel D/A converter are used to drive the dome loudspeaker system. 1 optical MADI connection is used to transmit the individual speaker signals from one multi-CPU PC system to the converter.

In addition to both loudspeaker setups, a group of four subwoofers surround the listening area. Like the WFS and dome loudspeaker systems, each subwoofer is driven by an individually generated driving signal. The signal processing software of each audio processing PC is able to generate 4 subwoofer signals.

The loudspeaker rings enclose an area of approximately elliptic shape (half axis of 2.40m and 2.80m) within the horizontal listener plane. An inner part of this area is used as listening area.

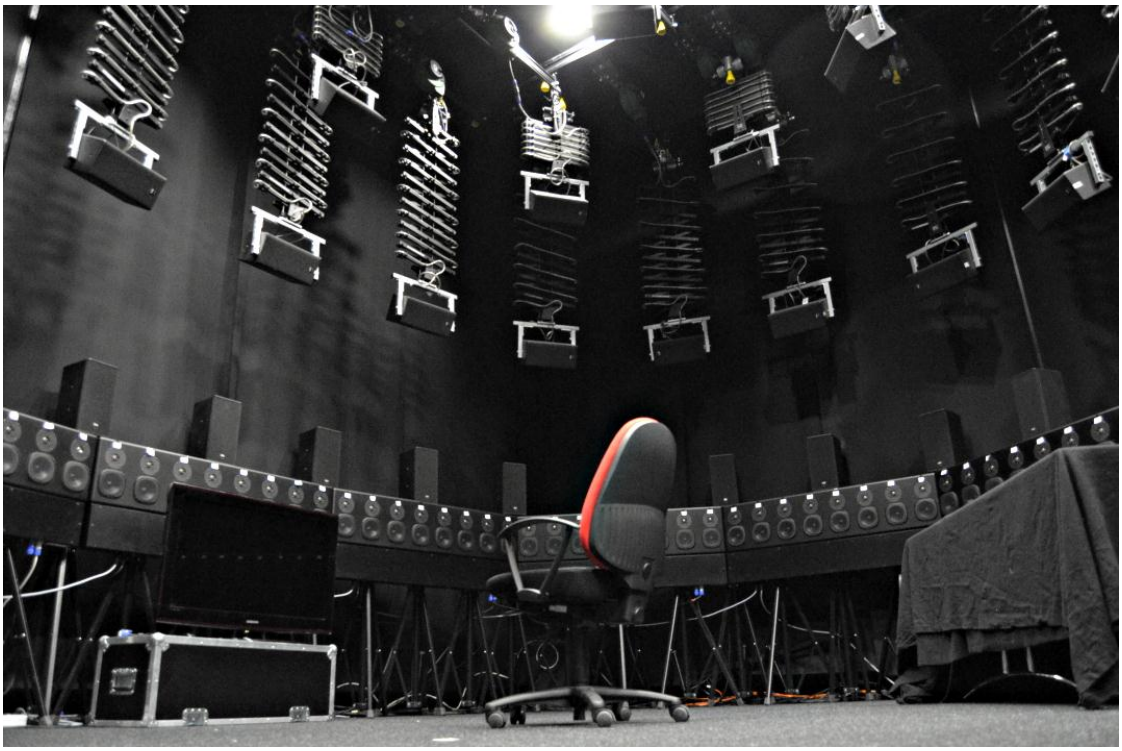

Figure 21: 3D audio lab at Fraunhofer IDMT: 3D low resolution setup (dome)

Both loudspeaker setups are placed in a dedicated 3D audio laboratory of Fraunhofer IDMT. To prevent cooling noises of power amplifiers and PC systems, the 3D audio lab is separated from the machine room. The facilities of the lab allow for preparing flexible signal connections between both rooms. While the loudspeakers and panels of the two speaker rings are set up on stands, the dome loudspeakers are mounted on pantographs that are designed to allow almost arbitrary speaker positioning and rotation. A broad variety of loudspeaker setups can thus be realised using the facilities of IDMT's 3D lab.

2.3.6 **Generation of Channel Based Audio Streams: Audio Scene Downmix**

Within the DIOMEDES architecture, channel based audio transmission of low channel number is used as basic audio format of DVB-T2 broadcast mode. P2P mode will be used to transmit a full object based audio scene.

Established channel based audio formats were chosen for basic audio transmission via DVB. AC-3 coding of 5.1 channels was chosen to be the main option for DVB-T2 audio coding in DIOMDES. As the audio cluster supports decoding of other established formats (e.g. 2.0 audio / MPEG-1 Layer II), the creation of such formats during the production process is a reasonable part of the broadcast side design. The channel based formats have a fixed channel layout (mapping of signals to speaker positions).

The creation of a complex object based audio scene is a process of higher complexity than creating a 2.0 or 5.1 mix. To reduce the overall effort of creating audio content for DIOMEDES transmission, the generation of 5.1 and 2.0 audio mixes is possible by automatically conducting a downmix of an object based audio scene to the desired channel based formats. To realize these steps, a two-stage procedure is applied based on the object based audio scene content:

1. Automatic downmix from complex object based audio scene to basic channel based surround format (5.1), applying dedicated downmix approaches of Fraunhofer IDMT.

2. Downmix from channel based surround format of step 1 to basic channel based stereo format (2.0), by applying dedicated downmix approaches of IRT.

As a result of this processing chain, three instances of the audio scene are produced (see audio architecture block diagram).

The following downmix approaches are be considered for use within DIOMEDES context:

Downmix from object based to channel based surround format using adaptive audio scene rendering:

An object based audio scene consisting of a set of virtual source objects is processed by an audio scene rendering tool for controlling low-resolution loudspeaker setups (same approach as described above). The loudspeaker setup is configured to represent a virtual surround (5.1) loudspeaker setup. The resulting driving signals for these loudspeakers are the 5.1 channel set that will be used for further coding and downmixing. By varying the virtual speaker setup in its parameters, the downmix result can be influenced and adjusted. The downmix signals are be generated by using the IDMT rendering implementation.

Downmix from channel based surround sound to channel based 2.0 format

The common 5.1 setup, suggested in ITU-R BS.775-2 [11], is provided as a multichannel configuration. Beyond that and due to the continuing importance of two-channel sound with respect to domestic television broadcast, 2.0 audio is transmitted as an ancillary format.

For a reduction of complexity of the object based audio scenes, an automatic downmix processes is be used. This helps to provide a high production efficiency and safes costs towards manual mixing. A two-stage process is applied to the object based content, as described above.

In order to perform a WFS to 5.1 downmix a special rendering engine for low resolution speaker configurations is used. For this purpose there are known alternative technologies like the redistribution of virtual source objects from a audio scene to the channel based format with the help of the Vector Base Amplitude Panning (VBAP) approach [8]. This technique is easy to implement, but ignores among other things the frequency dependence of human localization and therefore doesn't reproduce the original perception of a WFS-playback system [10]. With VBAP, a virtual sound source at a certain position is created by applying the tangent panning law between the closest pair of loudspeakers. Another loudspeaker spatialization technique, which enables a more realistic representation of an object based audio scene with a 5.1 speaker setup, is the virtual microphone approach (ViMiC) [9]. At ViMiC, the source audio objects are placed into a simulated room equipped with virtual microphones. The driving signal for every speaker is typically generated by placing an artificial microphone in this virtual room at exactly the same position that ITU-R BS.775-2 suggests for the loudspeakers of the playback system. Besides the positioning and orientation of microphones, the ViMiC model also simulates their basic directivity pattern. Furthermore a shoe-box room model is used in order to increase the perception of sound source distance and the acoustic environment.

The virtual microphone technique however can, inherent to the system, lead to a higher amount of crosstalk and therewith to more correlation between the loudspeaker's driving signals compared to a conventional 5.1 mix.

Also the 5.1 downmix using the IDMT object based audio scene processing leads to inter-channel correlations in the 5.1 downmix. During multichannel playback this won't lead to any perceivable difference in audio quality, whereas an automatic 5.1 to 2.0 downmix will usually react very sensitive on highly correlated signals. Therefore the dedicated 5.1 to 2.0 downmix system by IRT had to be adapted to the DIOMEDES specific demands.

- **Adaption work**

With established methods of downmixing 5.1 audio content to 2.0 stereo, coherent signals between center channel resp. surround channels and left / right- channels usually lead to the

occurrence of comb-filter effects, which can heavily degrade audio quality in terms of timbre, localization and volume balance. The IRT downmix on the other hand tries to identify comb-filters in the frequency domain by comparing the result of the electro acoustic peak level summation (IST) with the desired value (SOLL). This value is deduced from the level perception of our hearing within the sound field: In the case of several sound sources playing at the same time within a room, it is assumed, that the sensed overall level is corresponding to a sound power summation of the individual levels. Following equations are illustrating the basic operating principle for two signals (A and B) summed together:

$$A_{IST} = |A| + |B|$$

$$A_{SOLL} = \sqrt{|A|^2 + |B|^2}$$

If the resulting level of the electrical summation A_{IST} doesn't correspond to the calculated A_{SOLL}, thus at this specific frequency index a comb-filter can be assumed. In consequence the amplitude value A is automatically corrected to A_{SOLL}.

To preserve the original level of effective power after downmixing with the IRT system, the level of the center channel, as well as the surround channels have to be lowered by -3dB, as can be seen in the following figure. This value is related to the assumption of a sheer sound power summation and doesn't completely represent the complex processes within a natural sound field. As a result, the overall level, the timbre and the direct- / diffuse-sound ratio can change compared to the multichannel playback and in dependence of the input signal characteristics.

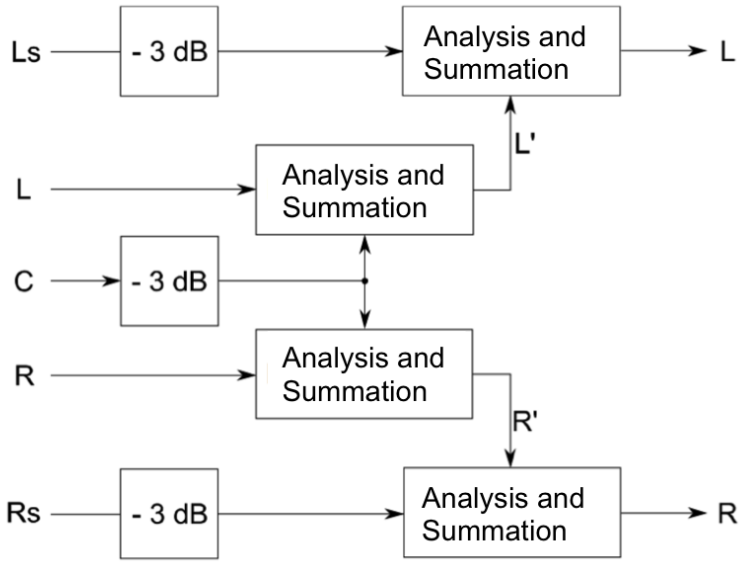

Figure 22 Functional schematic and attenuation coefficients used with the IRT downmix system by default

This effect observed, also increases with the amount of correlation between the 5.1 audio channels and had to be compensated for the use in DIOMEDES. Subsequently an enhanced model of the downmix process was developed and implemented, which controls the correction of the surround channels dependent on the degree of correlation between the summands. Therefore the correlation is separately determined for different bands within the frequency spectrum. For each of this bands, lower than 1,5kHz, the correction target is derived from the following rule:

$$A_{KORR} = (1 - c) * \sqrt{|A|^2 + |B|^2} + (|A| + |B|) * c)$$

In this equation c is representing the correlation coefficient. With this approach, a better preservation of the multichannel signal properties can be achieved.

- **Implementation**

For the use in DIOMEDES a hardware demonstrator was created (see following figure), as not to stress the computing power of the audio cluster. For this purpose an already existing embedded linux system, based on a x86 platform, had been chosen. The system is connected to the audio renderer either by ADAT or MADI. Beyond it supports a file-based workflow employing “Watch Folders”. The synchronization can be solved tethered via wordclock, because of the assumed proximity between the downmix and the audio cluster.

Figure 23 DIOMEDES hardware demonstrator used for the processing of the 5.1 to 2.0 downmix

The demonstrator enables the user to freely set the downmix coefficients and other important parameters electively by the use of an internal display or a web interface connected with the browser. The integrated JSON interface would furthermore allow an automatic control directly by the rendering engine.

2.3.7 **Rendering of channel based audio formats (Upmix)**

Due to the limited transmission capabilities of the DVB transmission channel, only conventional channel based audio formats are chosen to be distributed in DVB-mode.

Regardless, the audio rendering software structures and the loudspeaker setups of the DIOMEDES audio cluster are prepared for the rendering of object based audio scenes.

To reproduce the channel based formats of DVB transmission (usually 5.1 or 2.0 formats) with the audio cluster, these formats are transformed to a static object based audio scene during decoding. As this conversion makes use of the static channel layouts of the conventional audio formats, the conversion can be described with following steps:

- audio decoding
- generation of object description data: information retrieved from MPEG2-TS signalling and/or audio stream signalling is used to reconstruct a channel layout using “virtual loudspeakers” – virtual sound objects are positioned at the standard positions of the current channel layout. Influence on this positioning will be allowed via the configuration interface of the decoder module.

This virtual loudspeaker approach will in most cases lead to the reproduction of up to 6 virtual sources on a loudspeaker setup of a higher number of loudspeakers. Thus, this process can be referred to as an automatic upmixing process.

3 AUDIO-VISUAL ADAPTATION TECHNIQUES

3.1 Overview

Within the DIOMEDES architecture, the video and audio clusters are connected to the adaptation decision engine that triggers adaptation especially in video cluster and terminal modules to varying reception and reproduction parameters of the terminal system. The adaptation of both audio and video clusters to varying viewpoint positions is one example for the system's adaptation.

The following chapter pays special attention to the system's adaptation based on adaptation decisions.

3.2 Audio-Visual Adaptation

3.2.1 Adaptation Decision

The Adaptation Decision Engine (ADE) Module considered in DIOMEDES project is the block for deciding the media stream priorities, which are used by the P2P Software Module and Video Cluster for performing their respective content adaptation operations. The proposed adaptation decision taking algorithm responds to the "user terminal initialisation" and "context change" messages received from the Control Module. The operations of the adaptation decision taking algorithm are described in the following subsections.

3.2.1.1 ADE Module operation at the user terminal initialisation phase

In this phase, it is assumed that no *KPI metadata* and *user requested viewpoint ID* have been received by the ADE module. Therefore, the ADE module assigns default priorities to the views. Under the default priority settings, the middle cameras are given the highest priority, while priority ranking reduces as views are further away from the centre. It is also assumed that the middle-left view takes higher priority over the middle-right view.

Each view contains three components, which are uniquely identified by Program ID (PID) as used in MPEG-2 Transport Stream format. Thus, once the view priorities are assigned, the priority for each PID associated to individual views has to be set. Base layer PID of a view always has the highest priority over the enhancement layer PID and depth map PID for a given video stream. Assuming there is a total of 3 viewpoints in the multi-view video stream, and view priority order is calculated as V1>V2>V3, the priority order for the PIDs is shown in Table 2. It should be noted that P=1 refers to highest priority, whereas P=7 refers to lowest priority.

Table 2. The priority order of the PIDs

View priority order (View-ID)	V1	V2	V3
Base PID	P=1	P=3	P=4
Enhancement PID	P=5	P=6	P=7
Depth PID	P=2	P=3	P=4

The base PID of view V1 is assigned the highest priority since it is assumed to be the PID for backward compatible 2D view for serving legacy 2D displays. The ADE Module subsequently checks whether the user terminal is receiving the DVB streams. If it receives them, they are excluded from priority list message, so that the P2P Software Module does not download the stereoscopic views over the Internet unnecessarily. Assuming V1 and V2 form the stereoscopic video pair delivered through DVB, and also assuming that these viewpoints do not have an enhancement layer (as defined in system requirements in D2.1), the revised priority list is shown in Table 3. Furthermore, any view not required by the renderer will be excluded from the priority list.

Table 3. The revised priority order of the PIDs excluding DVB streams

View priority order (View-ID)	V1	V2	V3
Base PID	N/A	N/A	P=3
Enhancement PID	N/A	N/A	P=4
Depth PID	P=1	P=2	P=3

Note that some PIDs listed are assigned equal priority level, which depicts that the existence of either without the other is useless in the rendering process. For example, an extra view without its depth map cannot be incorporated in Depth Image Based Rendering (DIBR) process.

3.2.1.2 ADE Module operation when a change in context is detected

When new KPI metadata or user requested viewpoint information is received, it is considered as a context change.

When user requested viewpoint information is received, the ADE Module assigns higher priority to the closest 3 real camera viewpoints required to render the user requested stereoscopic view pair, as shown in Figure 24. Assuming *user requested viewpoint ID = x.y*, the following steps describe the prioritisation algorithm.

1. Calculate the virtual view pair $x_i.y_i = \{x_1.y_1, x_2.y_2\}$ to be rendered, based on *x.y* and *physical distance* input parameter
2. Determine the closest 3 real camera views (x_i, x_{i+1}, x_{i+2}) for rendering $x_1.y_1$ and $x_2.y_2$. These camera views are selected amongst the existing view-IDs and they are referred to as core camera views.
3. Calculate fitness of real camera views (i.e., x_i, x_{i+1} and x_{i+2}) for synthesising the virtual view pair determined in step 1 (i.e, $x_1.y_1$ and $x_2.y_2$), using the *KPI metadata*.
4. Assign priority orders to (x_i, x_{i+1}, x_{i+2}) based on fitness computed in step 3.
5. Assign the priority order for the remaining view-IDs based on the decreasing fitness for synthesising *user requested viewpoint.* This is done as described in step 3.

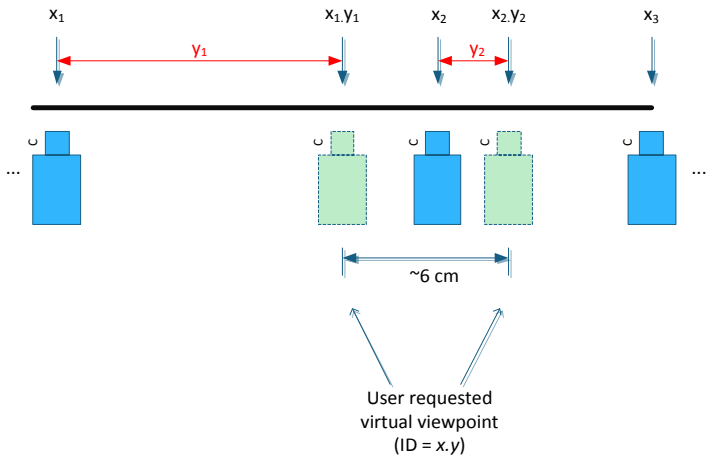

Figure 24. Real camera viewpoints required to render the user requested stereoscopic view pair

Calculation of the virtual view pair

The first step of the adaptation decision taking algorithm is to determine a virtual view pair, which forms the 3D view requested by the user. Since both the relative physical distances between each camera and the separation between a stereo-camera pair (i.e., ~6 cm) are

known to the ADE Module, it can calculate $x_1.y_1$ and $x_2.y_2$ accordingly. This is presented in Figure 24.

Determine the closest real camera views

In order to accurately render the virtual view pair calculated in the first step, the ADE Module needs to identify the real camera views that are in the closest proximity of the virtual views. These real camera views can be two three depending on the position of the user requested virtual viewpoint. Therefore, the second step focuses on determining which of the 2~3 real camera views out of all the available real camera views are necessary for rendering the virtual viewpoint. If both of the views constituting the user requested viewpoint fall in between two existing View IDs, then only 2 cameras are selected.

Calculation of fitness of real camera

In the third step, the ADE Module determines the fitness of each real camera view for synthesising the virtual view pair. This is done by assuming that the virtual view pair is synthesised using only one real camera view at a time. The fitness of a given real camera view is obtained by averaging the quality of the virtual views synthesised from this real camera. The method used for computing the quality of synthesised views is described in Section 3.2.1.3.

Assign the priority order

In the fourth step, the ADE Module determines the priority order for the 2~3 real camera views based on their fitness as defined in the previous step. Subsequently, the priorities for the base layer, enhancement layer and depth map of each view should be determined. For this purpose, the adaptation decision taking algorithm looks at the PIDs associated with each layer and depth map. The priority order amongst those is as follows:

- The base layer PID has always higher priority over the enhancement layer PID and depth map PID for the camera that has been assigned with the highest priority**.**
- For the remaining two cameras out of the selected three cameras, the base layer PID and depth map PID have equal priority.
- The enhancement layer PIDs of all three cameras have the lowest priority.
- The enhancement layer PID of the camera that has the highest priority, has the highest priority over the enhancement layer PIDs of the remaining cameras.

Assign the priority order for the remaining views

The fitness of each remaining camera view is also computed as described in step 3. The ranking of the PIDs of the remaining views (i.e., the cameras that are in the three-camera set found in step 2) are assigned with lower rankings than the lowest rank PID of the core camera views.

An example priority order is shown in Table 4, assuming there are 8 camera views. Here, V1, V2 and V3 are assumed to be the core camera views (based on the calculation in Step 2), and based on the camera fitness, the overall view priority order is calculated as V1>V2>V3>V4>V5>V6>V7>V8.

Table 4. An example priority order of the PIDs assuming V1, V2 and V3 are the core camera views

View priority order (View-ID)	V1	V2	V3	V4	V5	V6	V7	V8
Base PID	P=1	P=3	P=4	P=8	P=10	P=12	P=14	P=16
Enhancement PID	P=5	P=6	P=7	P=9	P=11	P=13	P=15	P=17
Depth PID	P=2	P=3	P=4	P=8	P=10	P=12	P=14	P=16

3.2.1.3 Fitness criterion for real camera views to synthesise virtual views

To synthesise virtual views requested by users, the most suitable real camera viewpoints need to be identified. For this purpose, a fitness criterion is proposed. Fitness refers to the projection of the objective quality of the original camera view to the location of the user requested virtual viewpoint as depicted in Figure 25. Projection was performed by a function derived through a subjective experiment, in which the quality of the rendered views at a set of selected target locations was assessed. For each target location, a stereoscopic view pair was rendered from a single real camera view and they are displayed on a stereoscopic display. The quality assessment was conducted using the non-categorical judgment method 0, where the observers rated the stereoscopic sequence. The experimental results are presented in Figure 26 for the Band and Music test sequences.

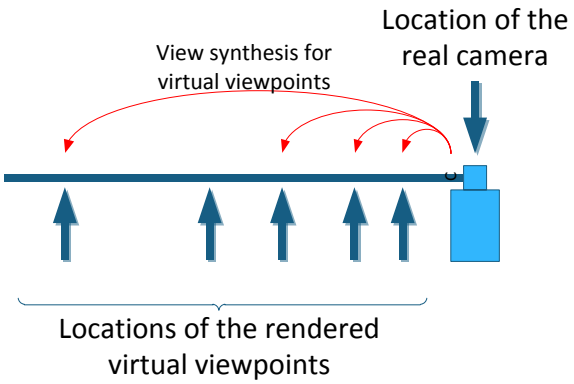

Figure 25. The concept of quality projection from real camera location to the user requested virtual viewpoint

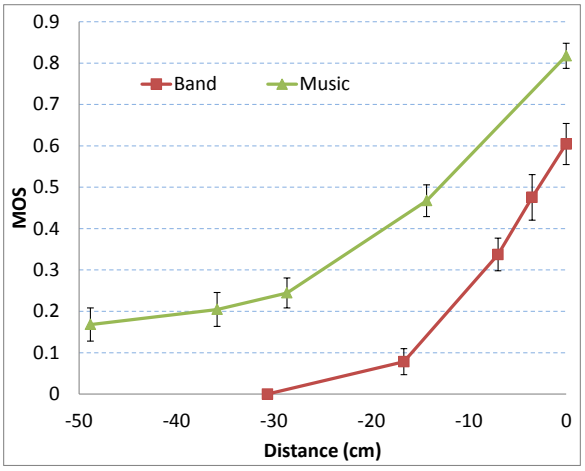

Figure 26. The average MOS results vs. the distance between the real and rendered (i.e., user requested) virtual view locations

As observed from the results, the visual quality of the rendered virtual views presents an exponential dropping behaviour with increasing distance from the real camera location. The approximated function for the above subjective MOS results is:

$$MOS = a \cdot e^{b \cdot x} \tag{1}$$

where x is the distance between the real camera location and the location of the rendered virtual view (in centimetres), and $a = 0.6663$ and $b = 0.0545$. This approximation is illustrated in Figure 27 for the combined results of the Band and Music sequences. The values for the constants a and b were determined using a curve fitting technique. The abovementioned function was adapted for the projection function as follows:

$$F = \max\left(0, Q_{ref} - a\left(1 - e^{b \cdot x}\right)\right) \tag{2}$$

where F is the fitness, and Q_{ref} is the quality of the real camera view. Q_{ref} is dependent on the perceptual quality of the coded camera frames as well as the quality of the corresponding depth map frames.

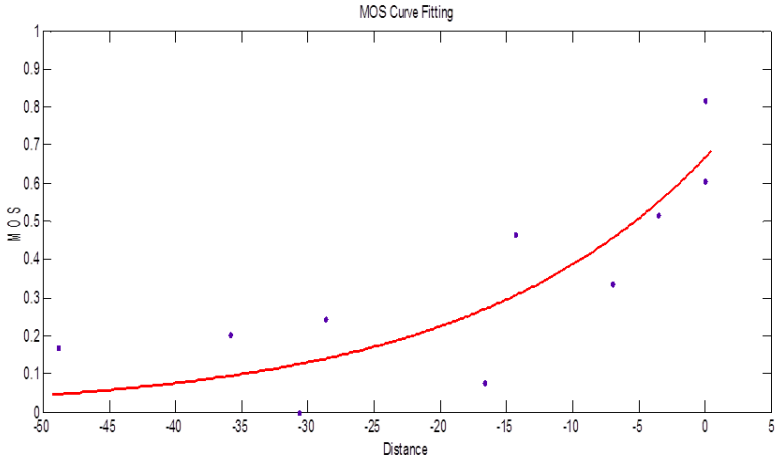

Figure 27. The MOS vs. distance plot for the Band and Music sequences showing the approximated function

3.2.2 Video Adaptation

DIOMEDES handles viewpoint adaptation and quality adaptation of individual viewpoints (through selective enhancement layer streaming) jointly, as described in Section 2.4.1. In this sense, view scalability corresponds to adjusting the number of views being downloaded via P2P Software Module and SNR scalability corresponds to adjusting the number of enhancement layers being downloaded via P2P Software Module.

ADE Module generates the priority descriptions incorporating both of the mentioned scalability options. The priority of base quality and enhancement quality layers, as well as depth maps of each viewpoint is ordered in the same list, where P2P Software Module makes use of these priority descriptions to perform the necessary video adaptation operations accordingly.

In the viewpoint adaptation scenario, the ADE Module excludes the viewpoints that are not required by the Video Cluster from its priority list before passing this information to the P2P Software Module (via the Control Module). In turn, the P2P Software Module does not download any streams associated to discarded viewpoints, which are not in the priority list.

In the network adaptation scenario (for adjusting quality), the P2P Software Module uses the same ADE Module-generated priority list to select the most prioritised enhancement layer streams considering their availability. In this way, the adequate number of enhancement layers, depth maps, and views can be adjusted by the P2P Software Module. P2P Software Module dynamically manages all downloading sessions based on the currently available bandwidth capacity, and chunk availability within a time window, and starts discarding streams starting from the least prioritised ones to match the current capacity. Otherwise, all requested streams to construct the required viewpoint are streamed.

The adaptation process in P2P SW Module has been implemented using a content-aware adaptive chunk scheduling mechanism. When the downloading session starts, the P2P module initializes one scheduling window per each base layer/ enhancement layer/ depth map in the priority list. Within this window, the most prioritised streams' chunks are requested by proper weighting (i.e. the chunk of the stream that is on the top of the ranking list has the most likelihood to be the requested chunk in the next slot). If the play out time approaches, which means that the downloading capacity is reached and/or stream availability is low, the weighting of the least prioritised streams' chunks in the ranking list are reduced gradually, while the weighting for more prioritised streams' chunks is increased. If a chunk can be

received before the play-out deadline, which is indicated by the AV Synch module, then the next scheduled chunk is requested for downloading. Using this pattern, it is possible to adapt the total content downloading rate to the available network rate, where the stream ranking is taken into consideration.

As mentioned in Section 2.4.1, video adaptation operations entail the user selection of specific viewpoints. For this purpose, the Control Module is invoked by the user viewpoint selection console (implemented as part of the user interface), where both the ADE Module and the Video Renderer Module are informed about the user requested View ID. ADE Module inherently incorporates the delivered user requested View ID information in prioritising the PIDs in the ranking list that is sent to the P2P SW module. At the same time, Video Renderer takes the user requested View ID to synthesise the view, or the view pair at exactly the camera coordinates of the user requested View ID. Hence, the viewpoint adaptation corresponds to both reconfiguring the video renderer and at the same time reconfiguring the P2P Software's download scheduling to help generate (render) the requested virtual camera viewpoint.

3.2.3 Audio Adaptation

The aspects of audio adaptation within the DIOMEDES context can be divided into different categories:

- Adaptation of object based audio scene to different reproduction systems. (Varying in loudspeaker setup)
- Adaptation of object based audio scene to different reproduction configurations (Varying in display and listener setup)
- Adaptation to varying video viewpoint of 3D multiview video renderer
- Adaptation to varying channel properties (Controlled by adaptation decision engine).

The following section gives a more detailed view on these categories.

3.2.4 **Adaptation to different reproduction systems**

The range of sound reproduction systems that should be supported within the DIOMEDES context spans from loudspeaker arrangements of 5 loudspeakers to dense loudspeaker setups like they are used for Wave Field Synthesis systems. The DIOMEDES system architecture uses an object based audio scene transmission approach to allow a scene transmission that is independent of a certain speaker layout, in contrast to the established channel based formats like 2.0 or 5.1. As the object based transmission allows the broadcast of numerous individually positioned sound objects, the scalable receiver system can be customized regarding reproduction precision. The sound reproduction precision increases with higher number of loudspeakers. In some cases, even 3D loudspeaker systems can be utilized on the receiver side. This variety of possible loudspeaker systems within the current context can be grouped in:

- 2D wave field synthesis loudspeaker setups (high resolution, high number of speakers)
- 2D low resolution loudspeaker setups (significantly lower loudspeaker number that WFS loudspeaker setup)
- 3D low resolution loudspeaker setup (includes loudspeakers above horizontal listener plane)

These types of setups are supposed to surround the listener area. For each type of setup, a dedicated rendering approach is implemented, as described above in the Audio Rendering chapter. One common audio scene description format (object description format) is shared among these approaches and allows adapting of one content stream to different speaker setups.

As the audio rendering cluster will be configured to drive one given loudspeaker setup that is not changed during operation, all incoming object based and channel based audio scenes are rendered to one loudspeaker system. Thus, this type audio scene adaptation will have a constant parameter set during operation.

3.2.5 **Adaptation to different reproduction configurations (listener & display setups)**

The DIOMEDES user terminal and rendering sub-system will meet varying conditions when it is configured at different user sites. Following parameters are expected to differ between terminal setups and to be the main aspects to be considered for an automatic audio scene adaptation:

- size and shape of the overall loudspeaker setup
- size and position of the listener area within the loudspeaker setup
- size and position of the video display in relation to loudspeaker system and listener area

As these aspects vary among different reproduction setups, a degradation of following perceptual aspects would to be expected, if an object based audio scene was reproduced without dedicated scene adaptation strategies.

- change in overall level and time of arrival proportions of the audio scene
- displacement of the focusing feature of virtual audio source
- displacement of virtual sources in relation to the listener position and change of perceived perspective
- displacement of virtual sound sources in relation to visual events of the video presentation

The aim of audio scene adaptation is to preserve the major aspects of the audio scene. Additional to the mentioned aspects, a list of architecture and system specific aspects could be named that also influence the design of an automatic scene adaptation. They are not presented in this document due to their close relation to internals of the implemented rendering system and production chain.

Not all of these aspects can be equally preserved using an automatic scene adaptation, as they concur with each other. For example, the position of virtual audio sources in relation to the listener position cannot be preserved when a congruent audio-visual presentation has higher priority in all system configurations. The following figure shows schematically the relative position and size of listener areas and displays belonging to 2 different reproduction setups.

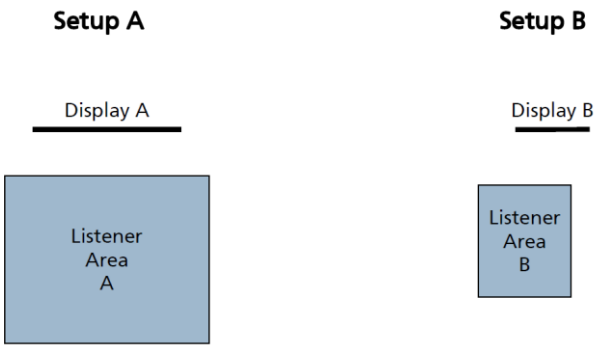

Figure 28: Example listener and display configurations: audio scene adaptation

Based on prioritising the audio scene aspects that should be preserved on different reproduction setups, a design of an automatic audio scene adaptation could be done. The resulting audio scene adaptation affects the overall object based audio production chain as follows:

- audio scene rendering is generally extended by a module conducting automatic scene adaptation; this module is applied before scene rendering during scene production (broadcast side) and during reproduction (receiver side)
- scene adaptation is generally a geometric transformation of the audio object arrangement
- an audio scene production is realised by positioning audio objects in relation to a standard listener and display setup; the resulting true positions of audio objects are hidden from the user
- to enable automatic scene adaptation, the data structure that describes an audio object is extended by data fields that control the scene adaptation
- the audio scene production tools are extended to support the audio scene adaptation features

The current rendering structure was extended by implementing modules that conduct automatic scene adaptation before rendering to display and listener arrangements. The description of these arrangements is part of the configuration data set that is evaluated by the audio cluster during start-up phase. Thus, this data accompanies the conventional rendering configuration data structures that describe the WFS or low resolution loudspeaker setup.

3.2.6 Adaptation to varying video viewpoint of 3D multiview video renderer

The DIOMEDES architecture allows for viewpoint changes by the terminal user that affect the video rendering by shifting the rendered camera view.

To adapt the the rendered audio scene to a varying viewpoint, a module for geometrical transforming the incoming object or channel based audio scene was implemented in the audio cluster. This module “SceneScaler” offers a range of geometrical transformations and modifications that can be applied to a given audio scene description:

- scaling, translation, rotation of audio object positions
- modification of audio object properties (source type, source properties, level and delay properties)

These modifications can be configured statically during configuration of the audio cluster or controlled during the realtime rendering via internal message commands within the audio cluster’s processing framework.

The viewpoint information sent to the audio cluster via JSON messages consists of the following parameters describing the video viewpoint in relation to the audio scene position:

- viewpoint positions (cartesian metrical coordinates, 3D)
- viewing direction (cartesian vector, 3D)

The SceneScaler module receives the adaptation properties after converting the JSON viewpoint description into a set of transformation parameters.

As the viewpoint adaptation is affecting the audio scene description before rendering, the incoming audio object signals remain unaffected. Besides the object based audio scene, also ncoming channel based audio formats can be processed by the viewpoint adaptation as they are rendered as set of audio objects at defined object positions.

3.2.7 **Adaptation to transmission channel properties**

Stream and channel configurations

The two media input channels of the DIOMEDES terminal structure are DVB-T2 and P2P (WAN). Each channel is capable of transporting multiple audio streams of different formats simultaneously beside the stereo video streams. The following table shows the useful plus the planned combinations of audio stream types and transmission channels within the DIOMEDES architecture.

Table 5: Overview of the transmission channels and usable audio stream types (includes estimates of the expected bitrate ranges)

	DVB-T2	P2P
Object based audio scene (constant max. number of simultaneous objects), >= 500kbit/s	Not to be implemented.	Included in minimal (default) configuration
Basic audio stream (Surround 5.1, e.g. AC-3) < 500kbit/s	Included in minimal (default) configuration	Optional (not used for DIOMEDES)
Basic audio stream (Stereo 2.0, e.g. AC-3, MPEG-1 Layer II) ,<=320kbit/s	Optional	Optional (not used for DIOMEDES)

The minimal configuration of broadcast streams is represented by one basic audio stream transmitted via DVB-T2 and one full object based audio scene stream transmitted via P2P. The basic audio stream will be a 5.1 surround stream that preferably will be generated by automatic downmix approaches directly from the object based audio scene as described above. The integration of automatic 5.1 to 2.0 downmix allows the generation of an additional audio stream that can be transmitted via DVB-T2 channel. This option was chosen to be implemented within DIOMEDES project.

If the terminal is running in P2P-only mode, only one object based audio stream is received. This audio stream is fed into the audio decoder and rendering modules. No decisions are required for this variant. If future modifications of the system architecture should allow multiple formats for P2P mode, a decision, which stream to decode and to render, would be necessary.

If the terminal is running in DVB-only mode, one or more basic, channel-based audio streams are received. These audio streams are fed into the audio decoder and rendering modules. A decision, which stream to decode and to render, is necessary.

If the terminal is running in combined DVB and P2P mode, multiple audio streams are received. These audio streams are fed into the audio decoder and rendering modules. A decision, which stream to decode and to render, is necessary.

The following sections describe the audio adaptation decisions that are implemented within the DIOMEDES terminal.

Stream switching structure: Audio cluster

The audio cluster is planned to be implemented with following features:

- Multiple audio streams can be streamed to the audio cluster simultaneously, using different PIDs. All streams are expected to be signalled using Program Association Table (PAT) and Program Map Table (PMT) (according to MPEG-2 Part 1: Systems [2]).
- The decoder module of the audio cluster will be able to support decoding a set of formats including the DIOMEDES object based audio scene format and basic-mode surround formats.
- If multiple streams of supported formats are present, an internal set of priority rules will determine, which stream is preferably decoded and rendered (e.g. the object based format is preferred to the basic format). The set of priority rules will be transformed into an internal stream priority list. The interruption of currently decoded streams and the reception of new streams of higher priority can trigger a stream switching.

The original plan of an external control of stream switching with a priority list sent to the audio cluster from the ADE was discarded, as the internal priority mechanism seemed to be sufficient for the audio cluster stream control.

The original plan of implementing double decoder structures for nearly seamless stream switching was deferred as seamless stream switching is a comfort feature and stream switching with a short decoding interruption seemed acceptable for the demonstrator implementation.

4 FUNCTIONALITY AND QUALITY EVALUATION

4.1 Overview

The implementation of the rendering modules within DIOMEDES architecture was evaluated regarding their functionality and rendering quality. Since the deliverable D3.4 focuses mainly on the rendering quality aspects, quality evaluation is only treated cursory in the following section.

4.2 Evaluation of Functionality

4.2.1 Video Rendering Functionality

Image quality

It has been shown that the MVD renderer is able to synthesize high quality images when supplied with high-quality images and depth maps, however it is very sensitive to errors in the depth maps and camera calibration information. Unfortunately, despite very careful preparations and lengthy depth estimation processes, practically all live multiview contents have some errors and imprecision in terms of camera calibration information (and thus rectification) and depth estimation. To make the renderer more robust against these inevitable artefacts in the input, we proposed and implemented several improvements to the algorithm, as well as implemented another view generation method in the renderer, which is more on the traditional side of view synthesis methods. Our subjective evaluation shows that that the point cloud based method performs better when synthesizing views from the contents used in the project, and gives plausible results when the virtual cameras stay inside the camera baseline. Of course, providing good quality images outside this area is clearly desirable, the real-time constraint poses a very strict restriction on the set of algorithms and improvements we can use. For this reason, implementation of sophisticated inpainting algorithms was out of scope. In the final system, when the viewer's movements are bounded with the area that has been captured with the cameras, the views generated are of good quality.

Performance

The main challenge in terms of performance was that most processing steps had to be done on the GPU, otherwise chances are very low that the renderer can achieve real-time performance generating multiple views on a single PC. Performance tests shown that one of the possible bottlenecks, uploading large textures to the GPU's memory with every frame indeed has a penalty, however it is possible to upload textures with sufficient speed. This can be reduced by uploading only selected cameras and associated depth maps, those with the highest contribution to the final image. The result is that view generation can work in real-time. Generating multiple views of course has a performance penalty, but this can be partially eliminated by reducing the resolution of the rendered images (something we can safely do, as the separate views have reduced resolution on current multi-view displays anyway, as discussed above).

Viewing freedom and adaptation

The proposed renderer is highly flexible in terms if input and output configurations, as it can basically operate with arbitrary camera arrangements on both sides, as well as using the decompressed images directly. Also, the renderer itself does not restrict the viewers from moving the camera anywhere in the scene, however, in the final system, it is desirable that viewers can see the recorded scene from viewpoints that have been captured by the multiview camera system, to maintain good visual quality ("less is more").

The renderer is also flexible enough to accept data via different data transmission methods (even simultaneously), maintaining synchronization, and to accommodate to varied amount of data available at its input according to changing network conditions.

4.2.2 Audio Rendering Functionality

Core audio rendering functionality

The core audio rendering modules were tested at Fraunhofer IDMT's various audio reproduction systems (rendering includes coefficient calculation and optimized signal convolution), rendering an audio scene of 32 simultaneous sources each (live signal input):

- WFS reproduction on a setup of 88 loudspeakers and 4 subwoofers
- Low resolution 2D reproduction on a setup of 23 loudspeakers and 4 subwoofers
- 3D low resolution reproduction on a setup of 52 loudspeakers and 4 subwoofers
- 3D low resolution reproduction on a setup of 60 loudspeakers and 4 subwoofers

In all situations, the audio processing was running on Linux x86 quad core PCs (2.8GHz). The rendering behaviour was stable for all tests.

Audio rendering functionality in combination with file and network decoding

For achieving a higher CPU usage, the pure audio rendering configurations were extended by existing object based audio scene file decoder and the DIOMEDES network audio scene decoder module. Each decoder module had to conduct audio decompression of 32 audio channels of an audio scene. The combinations were tested on the following setups:

- Low resolution 2D reproduction on a setup of 23 loudspeakers and 4 subwoofers
- 3D low resolution reproduction on a setup of 52 loudspeakers and 4 subwoofers
- 3D low resolution reproduction on a setup of 60 loudspeakers and 4 subwoofers

The audio rendering configurations were stable for all cases.

Audio rendering functionality with 2 rendering and 2 decoding instances on one PC

For the audio visual quality perception test described in D3.4, the audio rendering and decoding structures were doubled on one PC. An object based audio scene streaming configuration was created that allows the rendering of a scene on a high resolution loudspeaker setup when streamed to a first stream decoder, and the rendering on a subset of the loudspeaker setup when streamed to the second network decoder module running on the PC. In addition to that, 2 different sets of 32 equalisation filters were applied simultaneously to the object signals before rendering (compensation of the different loudspeaker densities for both rendering options).

Two rendering instances were configured to deliver the audio signals for different subsets of the connected 88 loudspeaker setup from 32 simultaneous audio objects:

- WFS reproduction on a setup of 88 loudspeakers (complete setup) and 4 subwoofers
- Low resolution 2D reproduction on a setup of 22 loudspeakers (subset of the 88 loudspeakers) and 4 subwoofers

The signal outputs of these rendering instances were added to be output to the same loudspeaker system.

Two spatial audio scene streaming decoders (“NetDecoder”) were receiving and decoding the incoming audio scene streams. The 32 output signals of the decoders were fed to the audio renderers via 2 arrays of 32 FIR convolvers each for frequency response equalization.

Only one decoder module received an audio scene stream at a time.

The system was fed with multiplexed stereo HD video and object based audio scene streams within a series of the 40min AV-perception tests. The audio rendering was reliable throughout the tests series.

Audio timing / AV synchronisation functionality

The synchronisation of audio rendering with an external video playback was controlled preparing different MPEG-2 TS example streams. Each stream contained object based audio scenes, mono or stereo video streams, and in some cases channel based audio streams to validate channel based format decoding. The following list shows some of the content examples that were used for the test streams:

- DIOMEDES “Music”: HD Stereo Video, object+channel based audio, ca. 20sec
- DIOMEDES “Lecture”: HD Stereo Video, object based audio, ca. 20sec
- DIOMEDES “Fencing”: HD Video, object+channel based audio, ca. 15sec
- DIOMEDES AV Sync Test video: HD video, object+channel based audio, >5min
- Trailer “The Settlers” PC game: PAL Video, object based audio, >3min
- Trailer “Creating Waves”: PAL Video, object based audio, >1.5min
- DVB-S example TV program streams (ASTRA satellite): PAL or HD Video, channel based audio
- DVD example streams: PAL video, channel based audio, streamed from DVD Video using VLC player software

All streams were sent from a dedicated streaming PC to both audio rendering PC and a video playback PC (using VLC player as video decoder and player). Streaming was done using DIOMEDES tools from IRT and OPTIBASE, a dedicated DIOMEDES streaming tool from Fraunhofer IDMT, VLC Player software and embedded streaming software of a DVB-S receiver.

All test streams showed very good synchronization properties. Audio rendering was stable even over long test periods (e.g. AV perception tests using long series of DIOMEDES lecture and music sequences), but VLC player video playback sometimes was unstable.

The tests of AV synchronicity between DIOMEDES audio rendering and DIOMEDES video cluster still have to be completed.

4.3 Quality Evaluation

4.3.1 Visual Quality Evaluation based on viewer tests and Quality of Experience Model

Detailed description of the conducted visual experiments can be found in Deliverable 3.4.

4.3.2 Audio Quality Evaluation

Evaluations of the perceived quality of the audio coding and audio rendering applied in DIOMEDES have done in the context on Quality of Experience (QoE) evaluation. Detailed descriptions of these experiments are included in D3.4.

5.1 to 2.0 Downmix quality evaluation measurement

For a proof of effectiveness of the proposed enhancements, a metrological verification was conducted. Therefore, a loudspeaker setup according to the recommendation ITU-R BS. 775-2 was placed in the anechoic chamber, as can be obtained from the following figure. The measurements had subsequently been verified under practical conditions in a common broadcast studio environment.

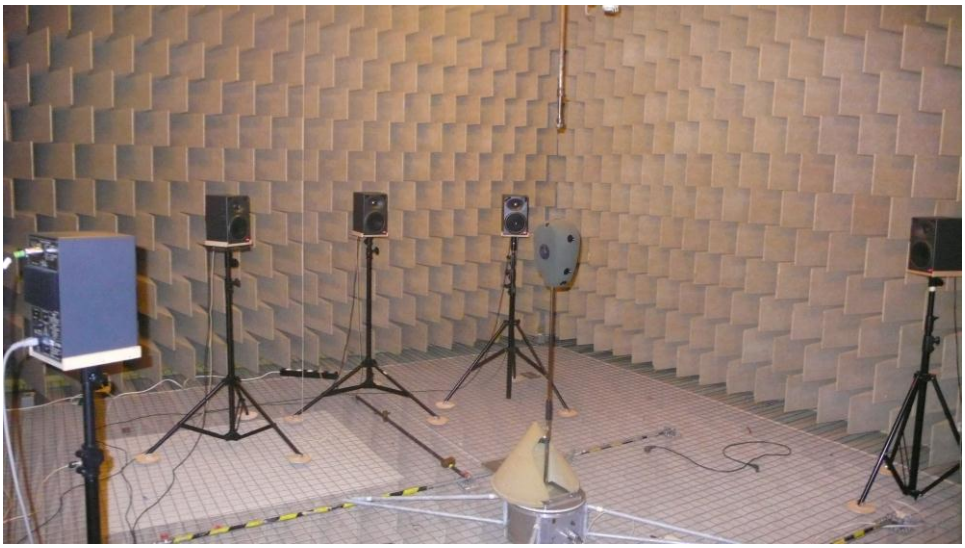

Figure 29 Measurement setup in the anechoic chamber

An artificial head, positioned at the sweet spot of the arrangement, served as a measurement device. Pink noise and natural music was used in order to excite the setup. To identify timbral and loudness changes within a time-variant audio signal, the measurement of integrated 1/3-octave-band levels had been utilized with an integration time of up to 120 seconds.

For test evaluation purposes the playback of the multichannel sound, of the ITU-, and the IRT downmix was measured. Furthermore the resulting graphs were shown in a common diagram. Thus it is possible to assess the level deviation between the different downmix systems and the changes made to the algorithm of the IRT downmix.

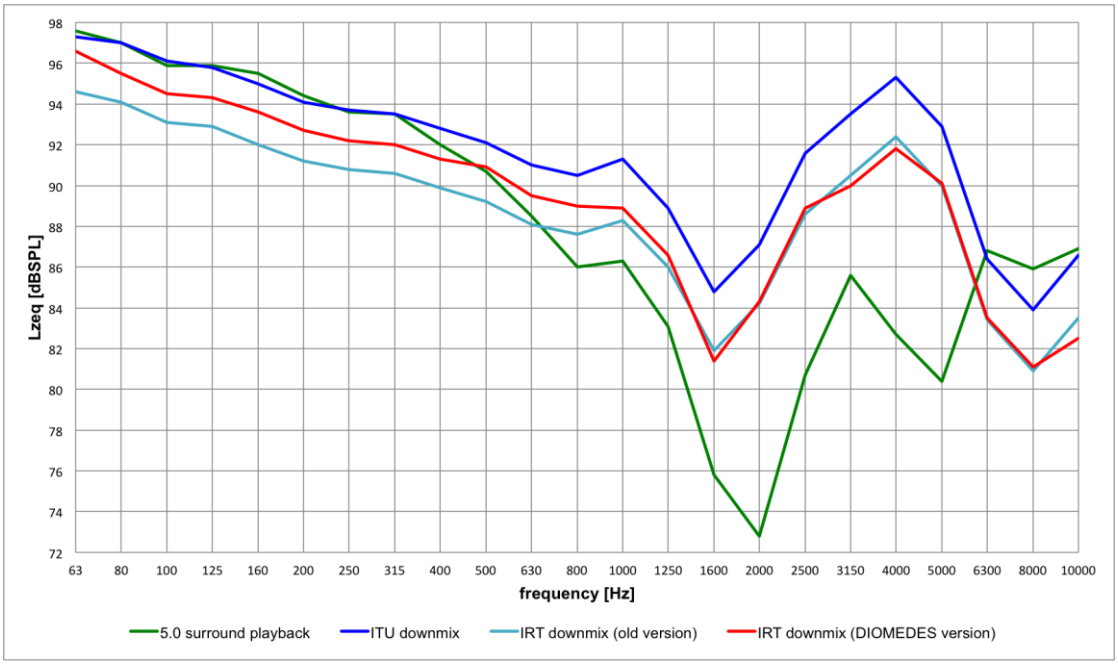

Figure 30 Measurement of coherent pink noise in the anechoic chamber

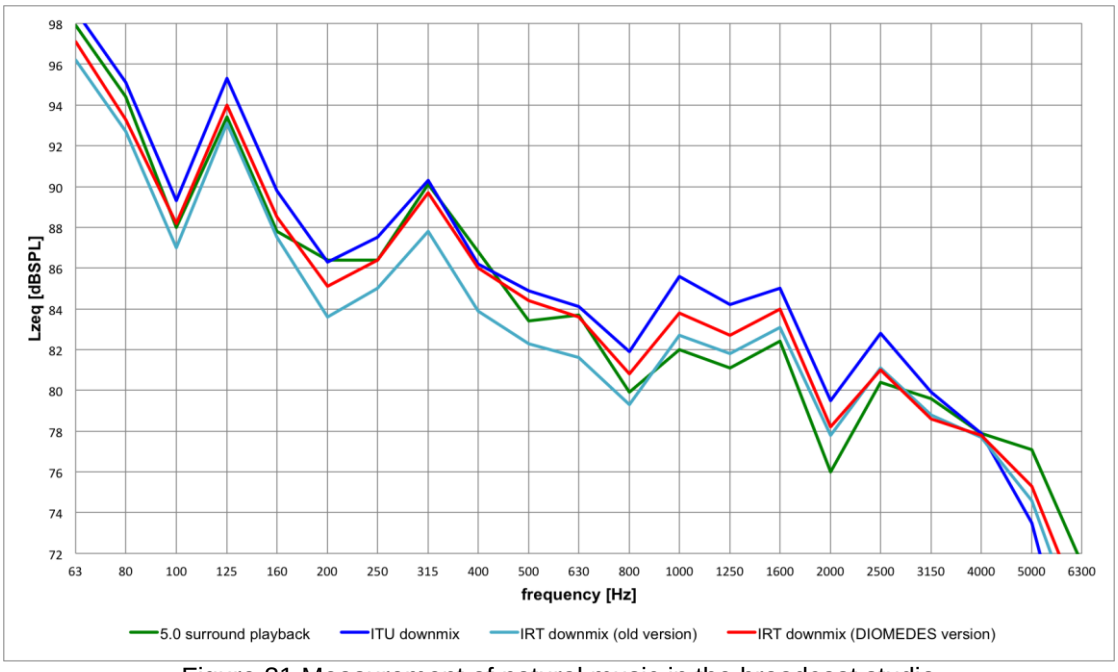

Figure 31 Measurement of natural music in the broadcast studio

As can be abstracted from the first figure above, the level of the improved IRT downmix at lower frequencies is averaged 2dB closer to the multichannel playback than the older downmix version. The remaining gap is almost consistent over the measured frequency range and mainly refers to the static surround attenuation of -3dB, which is set to the downmix algorithm by standard. Therefore it can be easily improved by adjusting this parameter in relation to the input signal and in this way achieved a high congruence to the multichannel playback.

The result of the ITU downmix also looks very good regarding the overall level deviation from the surround playback. In practice however, this observation has to be qualified, because this

downmix signal still contains the comb filter related distortions and moreover leads to a wrong level ratio between the center and surround channels.

The measurement under practical conditions, as shown in figure directly above, confirms the described findings. It also illustrates, that the progression of the adapted IRT algorithm is up to 2dB closer to the multichannel signal, which causes a distinctly perceivable improvement of the overall loudness preservation. Beyond that, the revised algorithm still effectively eliminates annoying comb filter effects and thus means the best possible compromise for the use in DIOMEDES.

5 SUMMARY

This deliverable described the concepts and structure of the stereo multi-view video and spatial audio rendering components within the DIOMEDES architecture.

The development of the final multiview video plus depth (MVD) rendering as an integration of an existing MVD renderer and a stereoscopic multiview renderer framework was described. The interconnection of the video rendering with the video decoding was emphasized, as the high video data rates in the DIOMDES multiview scenario require fast transmission designs and interfaces. The treatment of video rendering artifacts when utilizing the MVD renderer with depth map data of varying quality was discussed. Furthermore, attention was paid to the video rendering on multiview and light field displays.

For the DIOMEDES audio rendering, the object based audio scene approach was chosen and adapted to the broadcast and network transmission scenario. The rendering of scenes of audio objects, that are transferred in a streamable container format was described, highlighting the different audio rendering approaches utilized for DIOMDES. For home use, an audio rendering for less expensive loudspeaker systems than Wave Field Synthesis (WFS) was implemented. For the DIOMEDES use cases, the integration of audio rendering and decoding is shown. The simultaneous decoding of the audio scene and rendering onto a loudspeaker system have been implemented and performed within a modular multithreading software framework, that was presented. The automatic downmixing of the object based audio scenes to conventional channel based audio formats for broadcast transmission was described.

Special attention was paid to the aspect of synchronicity of audio and video rendering. The influence of the timing, clock regeneration from media stream time stamps and synchronisation mechanisms on the rendering was described.

The ability of the rendering architecture to adapt to different use cases and working conditions was presented. The Adaptation Decision Engine (ADE) was introduced as main instance for control module and video cluster, that defines the stream priorities for network transmission and processing. The handling of different video viewpoints and quality, of available base, enhancement and depth video streams was defined. Besides video rendering adaptation, the adaptation of the DIOMEDES audio cluster was described. The ability of the audio rendering to adapt flexibly to a wide range of loudspeaker setups, to different listener and display configurations, to video viewpoint changes and to the incoming audio scene streams was shown.

Finally, a set of functionality and quality evaluations was presented, that highlight some aspects of the achieved features of the DIOMDES rendering components.

6 REFERENCES

[1] ITU-T Recommendation BT.500–11, “Methodology for the Subjective Assessment of the Quality of Television Pictures,” Jun. 2002.

[2] MPEG-2 Part 1: Systems, ISO/IEC 13818-1

[3] Berkhout, A.J.: "A Holographic Approach to Acoustic Control", J.Audio Eng.Soc., vol. 36, December 1988, pp. 977–995

[4] Verheijen, E.N.G: "Sound Reproduction by Wave Field Synthesis", PhD thesis, Delft University of Technology, 1997

[5] E.W. Start: "Direct Sound Enhancement by Wave Field Synthesis", PhD thesis, Delft University of Technology, 1997

[6] S. Brix, T. Sporer, and J. Plogsties: "CARROUSO - An European approach to 3D-audio", 110th AES Convention. Audio Engineering Society (AES), May 2001

[7] Wittek, H.: "Perceptual differences between wavefield synthesis and stereophony", PhD thesis, University of Surrey, 2007

[8] V. Pulkki; “Virtual Sound Source Positioning Using Vector Base Amplitude Panning”, Journal of the Audio Engineering Society, 45/6:456– 466, 1997

[9] J. Braasch: "A loudspeaker-based 3D sound projection using Virtual Microphone Control (ViMiC)", Convention of the Audio-Eng. Soc. 118, Preprint 6430, Barcelona, 2005

[10] D. Griesinger, "Stereo and surround panning in practice," in Proceedings of the 112th Audio Engineering Society Convention, Munich, Germany, May 2002.

[11] Recommendation ITU-R BS.775-2, “Multichannel stereophonic sound system with and without accompanying picture”, International Telecommunication Union, Radiocommunication Sector, January 2006

[12] A. Franck and K. Brandenburg, “An overall optimization method for arbitrary sample rate converters based on integer rate SRC and Lagrange interpolation,” in Proc. IEEE Workshop Applications Signal Processing to Audio and Acoustics, New Paltz, NY, Oct. 2009.

[13] A. Franck, “Arbitrary sample rate conversion with resampling filters optimized for combination with oversampling,” in Proc. IEEE Workshop Applications Signal Processing to Audio and Acoustics, New Paltz, NY, Oct. 2011.

[14] A. Franck, “Performance evaluation of algorithms for arbitrary sample rate conversion,” in AES 131st Conference, New York, NY, Oct. 2011.

Appendix: Glossary of abbreviations

A/V	Audio/Video
ADAT	Alesis Digital Audio Tape
ADE	Adaptation Decision Engine
API	Application programming interface
ASRC	Arbitrary sample rate conversion
BSD	Berkeley Software Distribution
CPU	Central Processing Unit
D/A	Digital/analogue
Dx.x	Deliverable x.x
DVB	Digital Video Broadcasting
ES	Elementary Stream
FIR	Finite impulse response
GOP	Group of pictures
GPU	Graphics Processing Unit
HCA	Host Channel Adapter
HD	High Definition
IPoIB	IP over InfiniBand
ITU	International Telecommunication Union
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
LUT	Look-up table
MADI	Multichannel Audio Digital Interface
MOS	Mean Opinion Score
MPEG	Moving Picture Experts Group
MVD	MultiView plus Depth
NAL	Network Abstraction Layer
OFED	Open Fabrics Enterprise Distribution
P2P	Peer-to-peer

PAL	Phase alternate line
PAT	Program Association Table
PCR	Program Clock Reference
PDE	Partial differential equation
PES	Packetized Elementary Stream
PID	Packet Identifier
PMT	Program Map Table
PTS	Presentation Time Stamp
QoE	Quality of Experience
RC	Reliable Connection
RDMA	Remote Direct Memory Access
RGB	Red green blue
SDK	Software development kit
SMPTE	Society of Motion Picture and Television Engineers
TCP/IP	Transport control protocol / Internet protocol
TS	Transport stream
UD	Unreliable Datagram
UDP	User Datagram Protocol
VBAP	Vector Base Amplitude Panning
VCO	Voltage controlled oscillator
ViMiC	Virtual Microphone
WAN	Wide area network
WFS	Wave field synthesis
WP	Work Package