

Deliverable

Project Acronym:	ImmersionTV
Grant Agreement number:	688619
Project Title:	<i>Immersive Experiences around TV, an integrated toolset for the production and distribution of immersive and interactive content across devices.</i>

D3.1 Design Architecture

Revision: 1.7

Authors:

Maciej Glowiak (PSNC) – main editor

Delivery date: M11

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement 688619

Dissemination Level

P	Public	x
C	Confidential, only for members of the consortium and the Commission Services	

Abstract: This deliverable defines the core technical concepts of ImmersionTV project, it lays out the global architecture of the different software and hardware modules as well as the standards that will be used in the scope of the project. It establishes the architecture and the interfaces of the various functional blocks that comprises the immersive system. Functionality of the ImmersionTV project is based on several modules that implement particular functions required for video acquisition, video processing and encoding, video production, content delivery and reception and display. All the main components are described in details in order to provide the reference for implementation and integration activities in the project. Finally, the Quality of Experience methodology and test scenarios are pointed out to support and formalize project results validation procedure.

The document provides overall system architecture but it is focused on Pilot 1 and Pilot 2 requirements collected and analysed in Deliverable D2.3

REVISION HISTORY

Revision	Date	Author	Organi sation	Description
0.9	20 Jul 2016	Maciej Glowiak	PSNC	Previous version of D3.1 (M4) as an input for this document
1.0	16 Dec 2017	Maciej Glowiak	PSNC	Collected contribution from all partners
1.1	30 Dec 2017	Joan Llobera	I2CAT	First review
1.2	17 Jan 2017	Luk Overmeire	VRT	Second review
1.3	19 Jan 2017	Joan Llobera	I2cat	Technical review
1.4	3 Feb 2017	Stephane Valente, David McNally, Philippe Bekaert	All	Updates regarding technical review
1.5	6 Feb 2017	Maciej Glowiak	PSNC	Integration of partners contribution
1.6	15 Feb 2017	Luk Overmeire	VRT	Final review
1.7	23 Feb 2017	Maciej Glowiak	PSNC	Integration of partners corrections, final corrections and styling

Disclaimer

The information, documentation and figures available in this deliverable, is written by the **ImmersiaTV** (*Immersive Experiences around TV, an integrated toolset for the production and distribution of immersive and interactive content across devices*) – project consortium under EC grant agreement H2020 - ICT15 688619 and does not necessarily reflect the views of the European Commission. The European Commission is not liable for any use that may be made of the information contained herein.

Statement of originality:

This document contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

EXECUTIVE SUMMARY

This deliverable defines the core technical concepts and general architecture of the ImmersiaTV project for implementation and development purposes. Based on the state-of-the-art analysis, scope of the project pilots and technical requirements and specifications defined in D2.3 all partners of WP3 worked together and performed various actions to define the different modules and functionalities as well as interactions between these modules. This deliverable was created incrementally, building further upon the previous version of D3.1. The previous version focused mostly on designing the architecture for Pilot 1. The current version was modified and enhanced by issues related to Pilot 2, which means the document describes both off-line and live systems.

Chapter 1 presents the background of the project, including the purpose of this document and relation between all corresponding Work Packages.

Chapter 2 contains all the requirements on content production and content consumption taken from WP2 (D2.3) as well as some considerations on Pilot execution (from WP4).

The general architecture and workflow, which are presented in Chapter 3, take into account both off-line and live scenarios for Pilot 1 and 2 and makes the system ready for further extensions (Pilot 3). However, the document includes already some details regarding Pilot 3 and advanced live scenario with elements of Phase 3. After the global architecture is defined and depicted, all the modules are characterised, defined and elaborated.

Chapter 4 defines Phase 1 platform architecture towards Pilot 1. **Section 4.1** contains architecture overview for Pilot 1 and introduces all the components described in next sections. In **Section 4.2** the capture and stitching process is described, containing selected camera systems, their capabilities and possible usage in the different Pilots. This chapter also contains requirements for stitching software input and output as well as the architecture and functionality of the omnidirectional video acquisition and processing.

Section 4.3 provides information on production tools. It concentrates on an off-line post production process for synchronized and interactive multi-platform 360° content across multiple devices using Adobe After Effects and Adobe Premiere.

Section 4.4 describes the encoding process and contains a list of H.264 codec parameters agreed between modules and tools for pilot 1. The codec definition is a result of discussions between all partners involved in content creation, editing, encoding, streaming and playback.

Section 4.5 is dedicated to content distribution and describes the DASH concept that will be used in the ImmersiaTV distribution system for off-line workflows.

Section 4.6 provides information on the reception and display side, where multiple streams are synchronously transported to end-users' devices such as Head Mounted Display, tablets and TVs. The solution of synchronisation relies on HbbTV 2.0 concepts and describes how the different types of content (omnidirectional and directive) can be synchronized across devices and how interaction between both omnidirectional and directive content can be realised in an immersive display.

Chapter 4.7 describes the Quality of Experience methodology and test scenarios. They are intended to support and formalize project results validation procedure.

Chapter 5 extends the global architecture towards Pilot 2. The contents is focused on live scenario and defines new capabilities and functionalities of hardware and software components.

In **section 5.1** the general architecture for Phase 2 is presented as well as all the components are introduced. This gives good overview of the live ImmersiaTV system workflow.

Section 5.2 describes new hardware omnidirectional cameras and stitching boxes (Orah 4i and EDM cameras). Architecture and workflow of these components are defined here.

Section 5.3 describes Live Production Tools, which are new components of the ImmersiaTV system architecture for Pilot 2. Live Production Tools are the central point of the workflow for a live scenario connecting capturing, editing and streaming functionalities.

Section 5.4 is dedicated to encoding issues, and contains considerations for AVC and HEVC usage in the project. Besides requirements for encoding components, the omnidirectional image mappings are described as well as a comparison of the mentioned codecs is presented.

Sections 5.5 and **5.6** explain the Distribution and Reception and Display components, but the architecture of them don't differ from Phase 1. During the first phase of the project, these components were designed and implemented to enable live scenarios in the same way as off-line ones.

In **Section 5.7** Quality of Experience considerations are addressed and the architecture of the component is presented. The section also contains metrics to be computed and collected during QoE assessment.

Finally, **Chapter 6** contains the conclusions from the document.

CONTRIBUTORS

First Name	Last Name	Company	e-Mail
Adriaan	Barri	IMEC-ETRO	abarri@etro.vub.ac.be
Philippe	Bekaert	IMEC-EDM	philippe.bekaert@iminds.be
Touradj	Ebrahimi	EPFL	touradj.ebrahimi@epfl.ch
Sergi	Fernandez	I2CAT	sergi.fernandez@i2cat.net
Maciej	Glowiak	PSNC	mac@man.poznan.pl
Alexandr	Kelembet	CINEGY	kelembet@cinegy.com
Joan	Llobera	I2CAT	joan.llobera@i2cat.net
Saeed	Mahmoudpour	IMEC-ETRO	smahmoud@etro.vub.ac.be
Szymon	Malewski	PSNC	szymonm@man.poznan.pl
David	Mc Nally	EPFL	david.mcnally@epfl.ch
Juan	Nunez	I2CAT	juan.antonio.nunez@i2cat.net
Luk	Overmeire	VRT	Luk.Overmeire@VRT.BE
Maciej	Strozyk	PSNC	mackostr@man.poznan.pl
Stéphane	Valente	VIDEOSTITCH	stephane@video-stitch.com
Wendy	Van den Broeck	IMEC-SMIT	wvdbroec@vub.ac.be

CONTENTS

Revision History.....	1
Executive Summary.....	2
Contributors.....	4
Table of Figures.....	8
List of acronyms.....	11
1. Introduction.....	12
1.1. Purpose of this document.....	12
1.2. Scope of this document.....	12
1.3. Status of this document.....	12
1.4. Relation with other ImmersiaTV activities.....	12
2. Requirements Analysis.....	13
2.1. Requirements on Content Creation.....	13
2.1.1. Pilot 1.....	13
2.1.2. Pilot 2.....	15
2.2. Requirements on Content Consumption.....	19
2.2.1. Pilot 1.....	19
2.2.2. Pilot 2.....	20
2.1. Requirements on Pilot execution – WP4.....	22
3. General Workflow and Platform Overview.....	22
3.1. ImmersiaTV platform.....	22
3.1.1. Capture.....	22
3.1.2. Stitching.....	23
3.1.3. Encoding.....	23
3.1.4. Off-Line production.....	23
3.1.5. Live production.....	23
3.1.6. Distribution.....	23
3.1.7. Reception and interaction.....	24
3.1.8. Quality of Experience.....	24
3.2. Workflow.....	24
4. Phase 1 Platform and Architecture.....	24
4.1. Architecture Overview.....	24
4.2. Capture and Stitching.....	25
4.2.1. Description.....	25

4.2.2.	Architecture.....	26
4.2.3.	Workflow.....	28
4.3.	Off-line production Tools	29
4.3.1.	Description	29
4.3.2.	Software architecture	30
4.3.3.	Workflow.....	33
4.4.	Encoding.....	33
4.4.1.	Description	33
4.4.2.	Architecture.....	34
4.4.3.	Workflow.....	35
4.5.	Content Distribution.....	35
4.5.1.	Description	35
4.5.2.	Software architecture	35
4.5.3.	Workflow.....	37
4.6.	Reception, Interaction and Display	37
4.6.1.	Description	37
4.6.2.	Software architecture	38
4.6.3.	Session management device.....	39
4.6.4.	Receiver devices.....	41
4.7.	Quality of Experience	47
4.7.1.	Description	47
4.7.1.	Subjective metrics for quality assessment.....	47
5.	Phase 2 Platform and Architecture	49
5.1.	Architecture overview.....	49
5.2.	Live Capture and Stitching.....	50
5.2.1.	Overview	50
5.2.2.	Omnidirectional camera systems.....	51
5.2.3.	Live stitching system	56
5.2.4.	Studio.One stitching.....	59
5.3.	Live production Tools.....	62
5.3.1.	Description	62
5.3.2.	Architecture.....	63
5.3.3.	Workflow.....	64
5.4.	Encoding.....	65
5.4.1.	Overview	65

5.4.2.	Requirements	65
5.4.3.	Real-time encoding architecture.....	66
5.4.4.	Comparison of H.264 vs. H.265.....	68
5.4.5.	Re-mapping	68
5.5.	Distribution.....	70
5.6.	Reception, Interaction and Display	71
5.7.	Quality of Experience	71
5.7.1.	Description	71
5.7.2.	Software Architecture	71
5.7.3.	Workflow.....	73
6.	Conclusions	74

TABLE OF FIGURES

List of Figures

Figure 1: Relationship between different tasks	13
Figure 2: General ImmersiaTV architecture with all the components.....	22
Figure 3: General workflow for ImmersiaTV system	24
Figure 4: Architecture for ImmersiaTV system for Pilot 1	25
Figure 5: Architecture of Capture component.....	26
Figure 6: Blackmagic Micro StudioCamera 4K	28
Figure 7: Details of capture and stitching workflow using VideoStitch Studio and Vahana VR .	29
Figure 8: Schema of proposed editor workflow	30
Figure 9: Composition of the off-line production tools.....	33
Figure 10: Connectivity of devices in home network.....	38
Figure 11: Software architecture of a player	39
Figure 12: Sample ImmersiaTV metadata file.....	44
Figure 13: Logging module architecture and interactions	46
Figure 14: General architecture for Pilot 2	49
Figure 15: Architecture of the capture modules for Pilot 2.....	51
Figure 16: Orah 4i Camera	52
Figure 17: AZilPix Studio.One 360 camera rig (bottom left and middle) and semi-omnidirectional camera with canon fish eye lens (top left). Stage box on the right.	55
Figure 18: AZilPix 360 camera centre front Dranouter music festival main stage (ICoSOLE EU project, www.icosole.eu)	55
Figure 19: Example of semi-omnidirectional video.	56
Figure 22: Vahana VR software screenshot	57
Figure 21: Orah Stitching Box.....	58
Figure 22: Studio.One server in rack mount case with time code generator and audio interface and robust connectors to stage box, cameras and control room.....	60
Figure 23: Screen shot of the interface of the Studio.One software. Concerns a medical surgery procedure video capture for training purposes. The session integrated 2 HD-SDI sources, a 360 rig and 4 semi-omnidirectional rigs - all real time on one server. (with dlive – eSurgie)	61
Figure 24: Equirectangular video produced simultaneously with previous. (In fact, a semi-omnidirectional camera would have been better for this point of view, as the action is concentrated in less than 170 degrees field of view.)	61
Figure 25: Sample user interface for Cinegy Live VR	62
Figure 26: Sample WEB configuration interface for Cinegy Transport.....	63
Figure 27: Architecture of Live Production Tools.....	63
Figure 28: Real-time video codec within the ImmersiaTV work flow	66

Figure 29: The re-mapping process for an input YUV image 69

Figure 30: Pixel loss during remapping from equirectangular to cubic projection 70

Figure 31: Architecture of QoE component 72

List of tables

Table 1: Requirements from D2.3 – WP2 on Content Creation for Pilot 1.....	15
Table 2: Requirements from D2.3 – WP2 on Content Creation for Pilot 2.....	19
Table 3: Requirements from D2.3 – WP2 on Content Consumption for Pilot 1	20
Table 4: Requirements from WP2 on Content Consumption for Pilot 2	22
Table 5: Portal video effect parameters	32
Table 6: Preview mode parameters.....	32
Table 7: Off-line recording configuration	34
Table 8: Accepted values for call back	44
Table 9: Subjective metrics for QoE	48
Table 10: Specification of Orah Camera	53
Table 11: Specification of Vahana VR software	58
Table 12: Specification of Orah Stitching Box.....	59
Table 13: List of HEVC implementations on the market.....	68
Table 14: List of statistics required for QoE assessment	73

LIST OF ACRONYMS

Acronym	Description
HMD	Head Mounted Display
DASH	Dynamic Adaptive Streaming over HTTP
WP	Work Package
MPD	Media Presentation Description
AVC	Advanced Video Coding
HEVC	High Efficiency Video Coding
QoE	Quality of Experience

1. INTRODUCTION

1.1. Purpose of this document

This deliverable documents in detail the architecture design for omnidirectional content creation, processing, transmitting and displaying in ImmersiaTV as investigated in “Task 3.1 – Platform design and architecture”. The outcome of this task is a complete structured description of all the different components, the global architecture with diagram and workflows that will be implemented by all the tasks of WP3 (T3.2-T3.8).

1.2. Scope of this document

The objective of task T3.1 is to design an architecture for the overall ImmersiaTV system that will form the basis for the implementation and integration of all the software and hardware components. This document focuses on delivery of an architecture for both pilot 1 and pilot 2 and defines the full process chain of capturing, processing, encoding, content distribution up to users’ display following the objectives of WP3, which consist of:

- To design a reliable and robust system architecture of the hardware and software platform and facilitate a smooth integration of all the project technical components.
- To design, set up and deploy an omnidirectional camera system capable of capturing off-line video (Pilot 1) and support live scenarios (Pilot 2).
- To design and implement a real-time process to effectively encode multiple images from cameras into full omnidirectional video
- To design and implement the required functionalities to adapt the existing production tools to omnidirectional inputs and cross-device visualization and interaction.
- To design and implement the servers required to distribute omnidirectional content (incl. live stream) to remote users through existing and next generation access networks efficiently
- To design and implement the clients and libraries required to display omnidirectional video-based productions across devices (TV, second screen and HMD) maintaining coherence, synchronization, and responsivity in LAN environments.
- To integrate and test the different components in an end-to-end pilot and validate it in lab conditions.
- To document the overall process for other researchers and developers as well as produce lab demonstrations.

1.3. Status of this document

This is an updated version of previous D3.1 report (submitted in M3). The previous version focused on Pilot 1 architecture. This version extends the previous iteration with an architecture for Pilot 2 (M11). Considerations for Pilot 3 will be delivered in M20 by updating this document.

1.4. Relation with other ImmersiaTV activities

The relationship between this task and the other WP3 tasks and relevant WP2 and WP4 tasks is shown below on Figure 1.

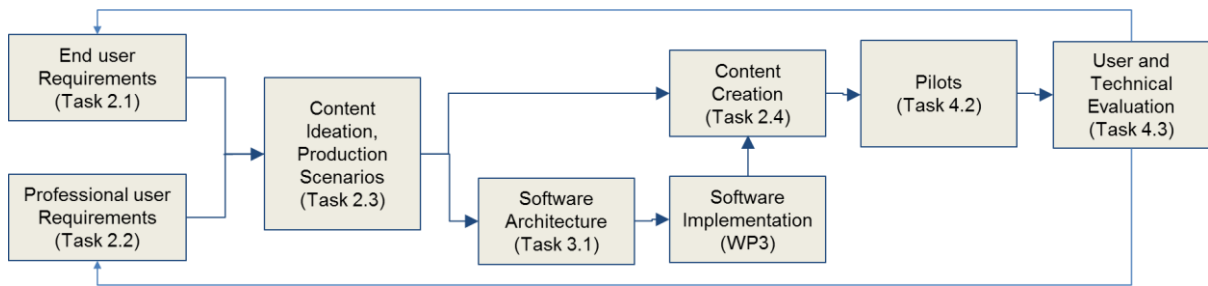


Figure 1: Relationship between different tasks

2. REQUIREMENTS ANALYSIS

2.1. Requirements on Content Creation

Deliverable 2.3 delivered in WP2 defines several structured requirements for content creation. They are collected and summarized in Table 1 and Table 2.

2.1.1. Pilot 1

Requirement	Comment
R-EDIT-1 The content creator can visualize the raw material across the different end-user devices.	For pilot 2 There will be a stream preview player based on Unity that will playback the rtsp streams of the raw material before being converted to MPEG-DASH to reduce latency.
R-EDIT-2 The content creator can use a standard edition software (Adobe Premiere, Final Cut, or other), e.g. by using predefined transitions, and avoid, for simple projects, using advanced compositing software.	Off-line production tools will be designed and implemented as Adobe plugin.
R-EDIT-3 In the editor software, the content creator can edit content for TV and for omnidirectional video in such a way that the timings of the content for the different targeted devices are continuously visible	The editor will use a standard Adobe Premiere workflow using plugins for omnidirectional video and metadata generation.
R-EDIT-4 The content creator can use Windows and OS X.	Off-line production tools will be available on Windows and OS X.
R-EDIT-5 The content creator can make use of an advanced mode in a compositing software (Nuke, Adobe After Effects)	Production tools plugin can be used in combination with advanced functionalities of Adobe Premiere
R-EDIT-6 The content creator can introduce interactivity within the editor timeline through <i>conditional transitions</i> between shots and scenes	Interactivity features such as enabling and disabling portals may be created directly by using the plugin
R-EDIT-7 The content creator can select, within the editor timeline, which video assets are visible within the TV, the tablet and the HMD	Timelines are tagged for the particular output device. Video assets may be assigned to such particular timeline.

R-EDIT-8 The content creator can also create ImmersiaTV scene typologies, i.e. interaction between devices, through conditional transitions within the editor timeline	This will be available for pilot 3.
R-EDIT-9 In pilot 1, the end user will experience the content with a common timing across devices (HMD, TV, tablet), it will be continuous and have no time jumps	All videos are synchronized and have common timings
R-EDIT-10 The content editor, using either a classic video editor or an advanced one, will easily define transitions and interactive transitions between omnidirectional videos using black and white video matte	Adobe Premiere plugins implements transition video filters corresponding with the shaders implemented in the player (FadeIn, FadeOut, Wipe, Slide, etc.)
R-EDIT-11 The content editor will be able to add a beauty layer to the interactive transition. This beauty layer will unfold synchronously with the video MATTE. It will be used to add borders and eventually other visual content needed for the transition	First the luma matte were implemented and tested, but finally the implementation was changed and the requirement was fulfilled by solution based on shaders.
R-EDIT-12 The content creator will be able to see omnidirectional content both in projected and non-projected views by using Previsualisation tools integrated in the content editor	Applied for off-line plugin
R-EDIT-13 The content creator will be able to pre-visualize transitions and interactive transitions within the editing software	Pre-visualisation of plugin effects is available in standard Adobe Premiere preview
R-EDIT-14 The content creator will be able to pre-visualize synchronized playout between different devices, for example, to see how TV and HMD content fit in timing	Standard functionality of Adobe Premiere supports this, as all the video assets are put on common timeline and are synchronized.
R-EDIT-15 An <i>export button</i> will generate a set of videos and metadata that is ready to distribute content across devices	Export button is available in auxiliary window that contains advanced exporting parameters. It generates all the videos and metadata in one pass.
R-EDIT-16 The export functionality will accept sequences involving different aspect ratios, due to differences in omnidirectional and traditional video formats (most likely solved through <i>nested sequences</i>).	Export functionality determines the resolution of each output sequence and accepts various aspect ratios.
R-EDIT-17 The common cutting points between devices will be visualized putting the content for the different devices in two sequences, one on top of the other	All sequences are in one common timeline
R-EDIT-18 It will be possible to define a label specifying the destination for each sequence	All videos are tagged / labelled for different devices
R-EDIT-19 The outcome should be: 1) A set of videos in the highest resolution possible. The videos should have a shared timestamp. This means that the	Output of the plugin contains a set of synchronized videos with common timestamps (1) and XML metadata file is compatible with DASH Manifest - the Media Presentation

<p>timestamp introduced at the frame level is common to all the different video streams. For example, the first frame of a video introduced exactly at second 12 of the broadcast should have its first frame with a timestamp set at 00:00:12:00.</p> <p>2) A metadata file detailing how the different videos have to be organised to compose an omnidirectional scene. This file should be compatible with broadband distribution standards</p>	<p>Description (2)</p>
--	------------------------

Table 1: Requirements from D2.3 – WP2 on Content Creation for Pilot 1

2.1.2. Pilot 2

Requirement	Comment
<p>R-CAP-1 Omnidirectional and directional cameras: Orah 4i, GoPro rigs and EDM Studio. One cameras (both 360 rigs and up to 8x 170 degree semi-omnidirectional high resolution cameras). The directional cameras used for the cyclocross pilot are Grass Valley LDK 8000 cameras (HD 1080i).</p>	<p>All mentioned cameras will be supported, but served in different ways. While Orah 4i produces RTMP stream, the EDM camera needs a special Azilpix server, that can also produce an RTSP stream or provide stitched content over SDI.</p>
<p>R-CAP-2 Camera distances: to deal with a variety of distance between cameras – from a few tens of meters (e.g. when filming a music contest) up to 1 kilometer (cyclocross competition). Fibre optics routing supported with adapter can cover the large distances.</p>	<p>If the maximum length of cables is exceeded, additional converters (into fiber) or amplifiers will be used.</p>
<p>R-CAP-3 Synchronisation of camera streams: the various streams resulting from different camera set-ups may have different latencies, each omnidirectional system typically has its own latency constraint. These streams need to be resynchronised, this will happen in the Cinegy server (live production tool)</p>	<p>Camera's live streams will be re-synced by Cinegy Transport with common timecode.</p>
<p>R-CAP-4 Cabling infrastructure:</p> <ul style="list-style-type: none"> ● Power: Power generator and power cables ● Ethernet cables: for larger distances fiber optics routing can be used with adapters ● Fibre sets: Connection cameras/capture server to control car: nevia fibre sets with break-out boxes (one in the field, one in the control car) carrying 4x HD-SDI and 1x Ethernet. The Ethernet connection can also be used to remotely control the Studio. One cameras via RESTful APIs. Different Nevia fibre sets can connect different cameras, such that cameras can be sufficiently distributed. ● Triax cables: to connect the directive (tv) cameras with the respective CCU's in the control car. ● BNC HD-SDI cables: HD-SDI cabling of up to 130m from Studio. One semi-omnidirectional cameras to their capture and processing server. (depending on 	<p>An example cabling scheme is illustrated in Figure 15.</p>

<p>cable quality - here supposed Belden 1694A ¹, with thicker high quality HD-SDI cables, or multiple cables per camera, larger distances can be covered)</p>	
<p>R-STI-1 Stitching output formats: Be able to output equirectangular real-time 4K 30 fps video streams, either compressed (AVC RTMP) or uncompressed (provided by EDM's stitching server, or Orah's Stitching Box HDMI out, or Vahana VR output ports)</p>	<p>These are the minimum requirements for Pilot 2. Cubic projection mappings, and AVC (Phase2) / HEVC (Phase 3) encoding, will be tested during the pilots.</p>
<p>R-STI-2 Vahana VR stitching: one Vahana VR instance per GoPro/Elmo rig</p>	<p>If the stitched output is uncompressed and provided on an HDMI or SDI port, the latency is in the order of 2 video frames.</p> <p>If the stitched output is compressed to an RTMP network stream, the latency is below 1 second.</p>
<p>R-STI-3 Orah stitching: One Orah Stitching Box per Orah 4i camera</p>	<p>If the stitched output is uncompressed and provided on the Stitching Box HDMI port, the latency is below 1 second.</p> <p>If the stitched output is compressed to a network RTMP stream, the latency is below 5 seconds.</p>
<p>R-STI-4 Studio.One stitching: the AZilPix Studio.One server performs 360 stitching in real-time with UHD 30fps output tested for live 360 streaming before.</p>	<p>Camera to screen latency is in the order of 2 frames (100 milliseconds).</p>
<p>R-PROD-1 Ingest: this describes the ingest side of the Live Production block. The possible sources include streams from omnidirectional cameras, streams from directive cameras, and optionally video files.</p> <ul style="list-style-type: none"> ● HD SDI signals to be converted to RTMP ● Compatibility with VahanaVR and Orah4i RTMP streams ● RESTful API to control the Studio.One camera ● Configurable number of sources (RTP, RTMP, RTSP, files) ● RTMP to RTP (Cinegy Live format) and RTSP (live preview) rewrapping support 	<p>Vahana VR is used as an encoding server for the uncompressed camera outputs to transform to RTMP AVC. Cinegy Transport is used to transform RTMP sources to other representations (RTP/RTSP/etc). Cinegy Live VR will display the real-time preview for all the source streams defined in the configuration.</p> <p>Support for file sources and control of Studio.One cameras via API is moved to the Phase 3.</p>
<p>R-PROD-2 Synchronisation of streams (Cinegy Transport): configurable service that lists the RTMP streams to be processed and specifies the required delay. Cinegy Transport can add a per-stream delay by rewriting the time stamp with a common value, based on UTC added with a preconfigured (measured) delay. Optionally the stream can also be physically delayed – i.e. re-broadcasted with defined delay to ensure both logical (metadata) and physical (delivery) streams are in sync.</p>	<p>Cinegy Transport will only change the stream representation (wrapper and reference timecode), no advanced processing (like re-encoding, scaling, etc.) is done. Cinegy Transport will also physically delay the incoming stream according to the settings using the RAM buffer.</p>

¹ <http://www.belden.com/techdatas/english/1694a.pdf>

<p>R-PROD-3 Live content edition (Cinegy Live): the live production operator workstation will have Cinegy Live VR application running that is Windows OS-based and provides the following functionality:</p> <ul style="list-style-type: none"> ● Sources: display of all incoming streams ● Preset 360 scene composition: One or more preconfigured 360 scene compositions (i.e. one or more omnidirectional videos and directive shots with interaction points) that are prepared beforehand and are available as ImmersiaTV scene XML format. This file initializes the live content delivery. The format is defined in D3.1. The operator can at least: <ul style="list-style-type: none"> ○ define a set of graphical icons and portals in the scene ○ select portions of omnidirectional and directive video for use in portals ○ assign position, size, shape and user/world reference of the insert ○ assign input streams to graphical icons representing cameras ○ define user interaction patterns e.g. switch from one scene composition to another one, triggered by user input ● Scene changes: the director can define required scene changes which activates an automatic scene update action, e.g. removing a camera icon, changing the actual source for a portal, ... An updated ImmersiaTV scene XML is generated in real-time and in sync with the media streams. ● Transitions: the director can activate a transition between omnidirectional and/or directive streams. A set of quick access scene transitions is available. Transitions typically apply for the <i>director's view</i> mode, but can also relate to the entry view of the <i>user exploration</i> mode. ● Live Preview: the director can locally preview scene modifications in real-time (or sufficiently low latency). ● Content for second screen (tablet): the director can preconfigure and change the second screen experience, e.g. a mosaic consisting of a number of camera viewpoints that can be activated 	<p>The focus for Pilot 2 is on sync delivery of director's view representation while advanced features (like user interactivity in exploration mode) are moved to Phase 3.</p> <p>Cinegy Live VR will provide operator with real-time preview of the source streams, list of available portals configurations (including position, shape, reference, etc.) and possible transitions between sources.</p> <p>Cinegy Live VR operator will be able to reassign the source for the scene and/or portals, add/remove portals from the scene, initiate complete scene change.</p> <p>Changes can be previewed with low latency using optional local ImmersiaTV player instance. For this purpose source streams will be optionally converted into RTSP versions to be fed into local preview player.</p>
<p>R-PROD-5 Distribution: collection and packaging of all streams and metadata to send towards MPEG-DASH streaming server that</p> <ul style="list-style-type: none"> ● accepts RTMP streams as live sources ● supports MPEG-DASH events mechanism ● allows scene profile modifications and scene updates 	<p>Cinegy Transport will generate segmented MP4 versions of the stream to be published on Distribution machine running Universal Streaming Platform (USP) software. USP is responsible for the final live MPEG-DASH configuration (availability, caching, load balancing, etc.).</p>

<ul style="list-style-type: none"> (optionally) performs external encoding to package the incoming streams as segmented MP4 files. However this packaging can be performed by Cinegy Transport in the final version. 	<p>The scene description file in ImmersiaTV XML format will be published on the same machine and be available for reference. The file will be updated with the changes made by the Cinegy Live VR operator. Scene changes will be aligned to the timecode provided by Cinegy Transport. Scene will reference MPEG-DASH versions of the streams for the global Internet based consumption.</p>
<p>R-ENC-1 Upstream external requirements for AVC-based solution</p>	<p>Fully integrated with VahanaVR platform, requirements are those specified within VahanaVR</p>
<p>R-ENC-2 Upstream external requirements for HEVC-based solution: to be realized on a standalone platform</p>	<p>No HEVC support in Phase 2. Will be designed for Phase 3.</p>
<p>R-ENC-2 Downstream external requirements: determined by capabilities of reception device</p> <p>Supported codec: H.264</p> <p>Codec parameters: bit rate, frame rate, frame size, chroma sub-sampling</p> <p>Natively supported projections: equirectangular, cubic or similar</p>	<p>Codec parameters to be adopted to those supported by the play-out device and rendering system.</p> <p>Rotated cube equirectangular project to be supported in Pilot 2</p>
<p>R-ENC-2 Encoding design requirements</p> <p>Real-time hardware (GPU: NV NVIDIA GeForce GTX 1080 or similar) supporting encoding of single video stream</p> <p>Ubuntu Linux Operating System</p> <p>YUV-based encoder input</p> <p>Input interface: HDMI or SDI card allowing for real-time acquisition and buffering</p> <p>Output interface: RTMP over Ethernet</p> <p>Per encoding session configuration</p>	<p>Related to HEVC real-time encoding. Will be designed for Phase 3</p>
<p>R-DIST-1 Live Preview (operator): different RTSP streams needs to be provided to preview contents in ImmersiaTV preview player.</p>	<p>Cinegy Transport will optionally generate RTSP versions of the streams to be referenced by the local preview. Cinegy Live VR will optionally generate alternative version of the Scene description file that references local low latency RTSP versions for the preview.</p>
<p>R-DIST-2 MPEG-DASH streams: live streams should be available on a server with metadata signaling the different contents to show at specific times.</p>	<p>Segmented MP4 versions of the stream to be published on Distribution machine running Universal Streaming Platform (USP) software. USP is responsible for the final live MPEG-DASH</p>

	configuration (availability, caching, load balancing, etc.).
R-DIST-3 Metadata updates: the metadata XML file should be accessible on the same server with the media content. The files are pointed by a relative path to itself. The metadata XML file will be refreshed in a timely manner in order to be consumed by the ImmersiaTV player such that the changes are reflected in the xml before they happen. This means that the metadata refresh interval should be less than the delay introduced by the distribution pipeline.	Cinegy Live VR will create and update Scene description file with new events based on the common timecode provided to all streams by Cinegy Transport.
R-DIST-4 Synchronisation: Input streams should be synchronized beforehand in the live production tooling, in origin to avoid desynchronization and problems in playback associated to bad alignment and poor streaming experience.	Synchronization will be handled by Cinegy Live VR

Table 2: Requirements from D2.3 – WP2 on Content Creation for Pilot 2

2.2. Requirements on Content Consumption

Deliverable 2.3 delivered in WP2 defines several structured requirements for content consumption. They are collected and summarized in Table 3 and Table 4.

2.2.1. Pilot 1

Requirement	Comment
<p>R-PLAY-1 Basic controls. The basic controls of the player will be:</p> <p>Select media source: which is likely to be a list of available content, located in public servers.</p> <p>Play: Starts to process the selected source.</p> <p>Stop: Stops the current reproduction and allows you to select a content once again.</p> <p>Select tablet or HMD mode: switch from tablet to HMD behaviour and rendering</p>	Client application provides basic functionality of selecting the source, controlling the stream and selecting output device
<p>R-PLAY-2 Metadata to describe and define the scene: The scene composition information has to be distributed to the player. This includes information like which videos are visible and where they placed or how video scenes are composed. This metadata may be transmitted in a multiplex or signalled within the stream itself, or it might be transmitted using a parallel communication channel</p>	Metadata XML is generated by production tools and are sent to the client by the distribution server. XML metadata describes the scene, visual parameters as well as portal location and transitions.
<p>R-PLAY-3 The scene is device dependent. Each type of device will have to render a different scene, as the interaction with</p>	Production tools create independent streams for various types of devices (TV, tablet, HMD). Scene

the user will be different. This implies there is a scene description for each type of device	description is provided for each type of device.
R-PLAY-4 Render multimedia content over textures and 3D objects. One or several videos will be displayed in different positions over the 3D scene (over a spherical surface, as a regular 360° video, or over plain surface in a portal-like effect).	Client uses Unity 3D engine and the multimedia content is rendered in 3D space.
R-PLAY-5 Apply video masks in videos. A mask is needed to overlay more than one video over the same texture forming an overlay of an arbitrary shape (i.e. to render a portal as a circle over the 360° sphere)	Will be delivered in Pilot 2 and 3
R-PLAY-6 Interaction management. The player needs to be able to process a systematic way to define interaction mechanisms in the end-user devices, and the methods implementing such interaction mechanisms need to be made available to the content creator	The interactivity is implemented as on click actions for tablet and smartphone devices and staring at a point for HMD.
R-PLAY-7 Achieve a frame level precise synchronisation: This is relevant as devices can display different omnidirectional and directional contents that were shot together, so any sort of desynchronization is going to be noticeable by the user	Client application guarantees synchronisation between devices. It is done by Session Manager and DVB-CSS clock synchronisation mechanisms.
R-PLAY-8 The devices may need to synchronize to any base media time at start up: A device can be turned on when there is already the reproduction going on in another device, so the one joining the group must get synchronized without affecting the other ongoing playbacks	Running a new instance of the client forces the stream synchronisation to the playback of central device (usually a TV)
R-PLAY-9 Basic audio control in the end-user devices	The audio is only available for TV device in pilot 1.
R-PLAY-10 Real time communication channel between devices: sending messages from one device to another	Will be delivered in Pilot 2 and 3
R-PLAY-11 Second screen scene definition: The definition of the second screen view (mosaic layout) in the tablet must be defined within the content production process.	The screen view in the tablet is defined in the post production tool.
R-PLAY-12 The end-user can capture screen casts and share them with other devices	Will be delivered in Pilot 2 and 3
R-PLAY-13 The end-user can capture screen casts and share them through social media	Will be delivered in Pilot 2 and 3

Table 3: Requirements from D2.3 – WP2 on Content Consumption for Pilot 1

2.2.2. Pilot 2

Requirement	Comment
-------------	---------

<p>R-DIST-5 Live transitions: transitions will be selected from a list, as indicated in the XML metadata file. The transitions will be produced in real time in the ImmersiaTV player.</p>	<p>Transitions will be integrated in the live production tool.</p>
<p>R-REC-1 Master device: TV should be a master device for synchronization in LAN.</p>	<p>Will be implemented for pilot 2, the TV will be the device which distributes the synchronization signal.</p>
<p>R-REC-2 Adaptive streaming: the ImmersiaTV player should adaptively select stream resolution and bitrate provided by the MPEG-DASH server, taking into account the hardware capabilities of playout device and available network bandwidth.</p>	<p>The player will be aware of the device and network capabilities and act in consequence.</p>
<p>R-REC-3 Refresh: Player should refresh scene layout (XML metadata refreshing), taking into account changes on the server side.</p>	<p>The player will refresh the metadata XML and apply the changes to the scene.</p>
<p>R-REC-1 Transition implementation: Player should implement a list of possible transitions for opening and closing of portals or graphical icons</p>	<p>The player will implement the same transitions that were available for pilot 1.</p>
<p>R-QoE-1 Logging information: the QoE module requires a number of logging information as input to evaluate quality (a list of QoE parameters has to be provided).</p>	<p>Events occurring in the application will be logged, allowing to extract necessary information.</p>
<p>R-QoE-1 Server implementation: the QoE module will be implemented on a server. This can be the same server as where the Encoder is running or a separate server close to the Encoder.</p>	<p>The implementation of the QoE module will be independent from other modules, ensuring compatibility with interfaces for input and output data.</p>
<p>R-QoE-1 Timing signals: at the client side, access to timing signals (timestamps, net clock data) is required (before and after delay) to extract start and length of the delay (stalling).</p>	<p>Logging of events will include time related information.</p>
<p>R-QoE-1 Side channel: a side channel between clients and the server is required to deliver necessary data (e.g. delay data) for QoE optimisation</p>	<p>Client will send updates of logged set of parameters to a server in XML files.</p>
<p>R-QoE-1 Feedback communication: a communication channel between QoE and Encoder is required to send QoE feedback for steering encoding parameters.</p>	<p>Feedback channel and steering parameters will be defined in pilot 3, after analysing outcomes of pilot 2.</p>

--	--

Table 4: Requirements from WP2 on Content Consumption for Pilot 2

2.1. Requirements on Pilot execution – WP4

WP4 Demonstration Pilots defines requirements on logging functionality needed for QoE process and Pilots evaluation. Logging functionality was taken into consideration and is defined in Section 4.6.4.5. Quality of Experience for Phase 1 is described in Chapter 4.7 and extended for Phase 2 in Chapter 5.7.

3. GENERAL WORKFLOW AND PLATFORM OVERVIEW

3.1. ImmersiaTV platform

ImmersiaTV aims to distribute omnidirectional and directive audiovisual content simultaneously to head mounted displays (HMD), companion screens and the traditional TV.

The content distributed is constituted of one or more omnidirectional videos, complemented with several directive shots, and metadata detailing how to merge these streams in an immersive display, as well as how to select portions of the omnidirectional stream for traditional TVs and tablets.

All the components of the system with mutual relationships are depicted on Figure 2 and shortly introduced in next sections.

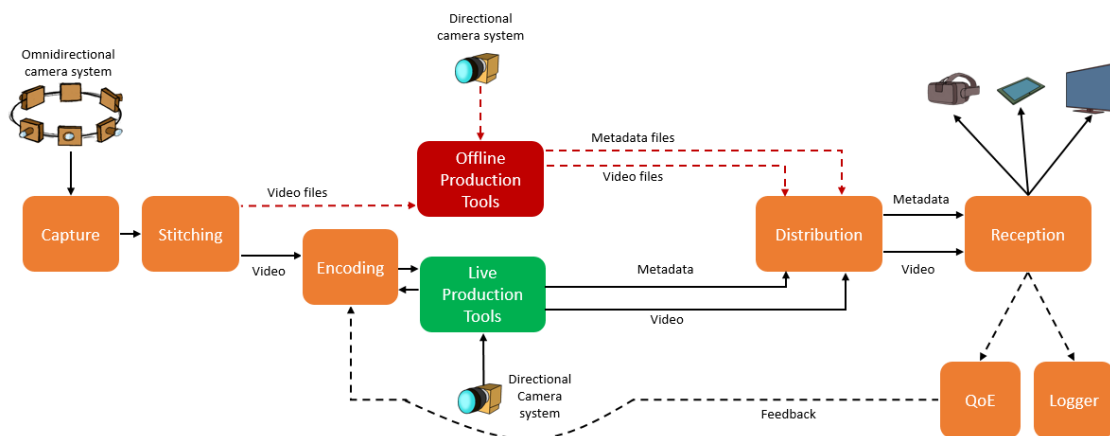


Figure 2: General ImmersiaTV architecture with all the components

3.1.1. Capture

This block is responsible for the physical capture of several video streams coming from 360 omnidirectional camera systems as well as other sources such as high resolution directive cameras, video clips, textual information and other metadata required for generating omnidirectional video enriched with audiovisual and auxiliary information in further stages.

3.1.2. Stitching

This block is responsible for grabbing and processing video images from 360 omnidirectional camera systems (constructed using multiple physical cameras). The main task of this block is to combine several video streams into one omnidirectional video stream (stitching process). Omnidirectional video from Capture component is the input for the Stitching module and other auxiliary data and video clips are used by production tools. While in Pilot 1 video files are recorded off-line and delivered to Off-line production tools (red dashed arrow), in the live scenario of Pilot 2, live streams are provided to Live Production Tools (black arrow).

3.1.3. Encoding

This block implements the video codec used by other modules of the ImmersiaTV system such as Capture and Stitching and Production Tools. For Pilot 1 standard encoders will be used, however encoding stream parameters needs to be defined between components. For Pilot 2 the Encoding block is executed between Stitching and Production Tools, as the second one will process already encoded streams.

3.1.4. Off-Line production

This block encompasses a set of tools and plugins for offline video editing with functionality of synchronization and combining multiple 2D and omnidirectional video sources and auxiliary data together. These data come from Capture and Stitching blocks. Off-line production tools operate on video files from omnidirectional and directional cameras and allows the editor to select, modify and combine streams using various transition effects and produce metadata. Off-line production tools produce video files as well as XML metadata that needs to be uploaded to Content Distribution server.

3.1.5. Live production

This block consists of a set of tools and plugins for live video editing with functionality of synchronization and combining multiple 2D and omnidirectional video sources and auxiliary data together. These data come from Capture and Stitching blocks by Encoding block. Live production tools operate on already encoded and optimized video streams from various types of cameras (directional and omnidirectional) and allows the editor to select, modify and combine streams using various transition effects in real time.

Live production tools produce target-specific representations of AVC or HEVC encoded video streams together coming together with XML metadata that both are passed to the Content Distribution server. All effects and transitions are described in metadata, so it is important this metadata is refreshed after each modification applied by Editor.

3.1.6. Distribution

This block handles the communication between off-line encoded contents or live streams and the end-user's player. It encapsulates selected video streams into network protocols and provides synchronized video and auxiliary streams to the player. Content Distribution server enables to select streams and handles the adaptation of resolution, codec, etc. in negotiation process with the client (Reception and interaction).

3.1.7. Reception and interaction

This block handles the end-user's reception side and display. It takes care of selecting proper video streams, receiving, decoding and displaying them. It also handles the synchronization of multiple received streams in order to present them to the end-user. The user can interact with the received content by selecting streams, choosing the device and performing basic playback actions. The client application provides logging functionality of playback parameters, that can be analysed off-line.

3.1.8. Quality of Experience

This module assesses the quality of content produced by the ImmersiaTV system by means of subjective and objective metrics and gives the feedback and recommendations of changes to the Capture and Stitching, Encoding and Reception modules. For Pilot 1 and 2 subjective metrics are used as well as taking advantage from logs generated by client application some QoE algorithms can be applied in order to assess and test the quality. In Pilot 3, real-time assessment of the quality will be introduced. These algorithms working real-time in the background will enable giving particular feedback to the encoder, which is able to adapt the quality, bitrate and resolution to current needs or limitations.

3.2. Workflow

The general workflow of the ImmersiaTV system covers all areas of production, distribution and reception of the omnidirectional video. The workflow is depicted on Figure 3.



Figure 3: General workflow for ImmersiaTV system

Physical captured streams are processed by the Capture component and then passed to advanced stitching algorithms (Stitching). Then omnidirectional video files or streams are encoded and can be modified and edited using Production Tools. The result of this are encoded videos with additional description metadata which are passed to the Distribution component which communicates with the client application and sends video streams through the network (Reception). The decoded images can be input for QoE and Logging modules for further analysis and quality assessment. This generic workflow of the system applies to both: off-line (Pilot 1 & 3) and live (Pilot 2 & 3) scenarios.

4. PHASE 1 PLATFORM AND ARCHITECTURE

4.1. Architecture Overview

The development plan in Phase 1 focuses on the implementation of the tools and modules required to demonstrate Pilot 1. In this offline scenario omnidirectional and directive streams will be captured, processed and aligned by the editor using Adobe Premiere Pro Plugin in order to prepare multi-platform omnidirectional views containing embedded two-dimensional video streams. Parts of the omnidirectional scene captured by several cameras will be processed and

stitched together using VideoStitch Studio. Other stitching software may be also used in order to meet specific requirements of the editor.

As a result of the offline production action, several H.264 video files with MP4 format wrapper and one metadata file will be generated, transferred to the DASH server, transcoded, and served in MPEG-DASH streamable format to be consumed by the end user's devices. The user will need to run a dedicated player in order to display omnidirectional content on their HMD and/or tablet device as well as a directive view on a TV set. All components required to achieve the goals of Pilot 1 are depicted in Figure 4.

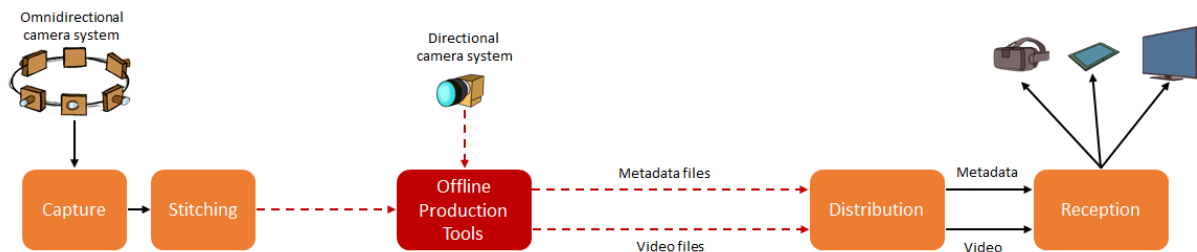


Figure 4: Architecture for ImmersiaTV system for Pilot 1

In the Pilot 1 workflow, directional and omnidirectional videos are passed between components as video files (red dashed arrows). Off-line production tools eases the process of editing and combining multiple video files into one and produces output video in various resolutions and versions for different devices together with description metadata. Videos and XML metadata files are finally passed to the distribution chain (MPEG-DASH server).

4.2. Capture and Stitching

4.2.1. Description

For capturing and stitching off-line visual content the ImmersiaTV system will use existing 360° camera rigs that are available on the market, such as GoPro and Elmo rigs.

For Pilot 1 we will rely on hardware available on the market in order to test all other components of the workflow:

- **Using off-the-shelf and professional cameras in conjunction with VideoStitch commercial products** (VideoStitch Studio and VahanaVR), that will be adapted to answer the constraints and requirements of broadcast-quality omnidirectional production workflows; VideoStitch Studio is dedicated to offline post-production, importing video files, stitching them together, and producing a 360° video file, while VahanaVR stitches live video streams. **These solutions will be deployed in pilot 1** (offline production). Although more oriented towards live productions, VahanaVR will be also tested in pilot 1, allowing the directors to have a live preview and direct feedback of the content during the shooting.
- **Using off-the-shelf high resolution professional traditional cameras for creating directional content in 4K.**

4.2.2. Architecture

There will be several camera systems tested and used in order to achieve the different goals of Pilots 1,2 and 3. For Pilot 1 and off-line content capturing the GoPro3 and Elmo QBIC rigs will be deployed. They will store several video streams on SD cards, and this content will be used in VideoStitch Studio which will do the stitching processing. Then the omnidirectional content will be available in off-line processing tools.

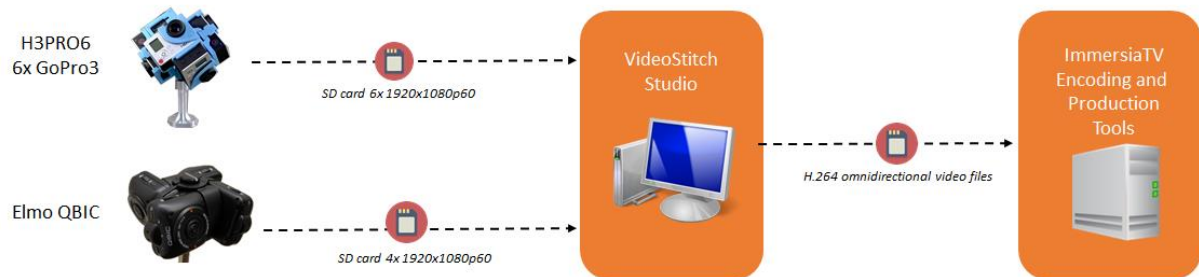



Figure 5: Architecture of Capture component

4.2.2.1. Omnidirectional camera systems


The most important parameters of the described camera systems with output capabilities are described as follows:

- **GoPro Hero 3 Black** camera rigs (3 or more sensors)². The H3PRO6 rig enables to combine 6 GoPro Hero 3 Black cameras together for capturing omnidirectional video streams. Each piece of the camera has 12MPix CMOS sensor and produces H.264 encoded stream or provides HDMI live output. The cameras support storing on microSD/microSDHC cards in resolution up to 4K, although the frame rate in UHD resolutions is rather poor (12 or 15 fps). Each camera handles Full HD resolution in 60 fps (recording) or 30 fps (HDMI output)

² <http://www.cnet.com/products/gopro-hero3/specs/>

Camera Rig	Number of sensors	Output resolution	How it is used
 <p>H3PRO6 Rig with 6 GoPro 3 Black cameras</p>	6	6x 1920x1080p/60 fps when recording on SD card 6x 1920x1080p/30 fps on HDMI output	Outputs video files on SD cards for VideoStitch Studio, or HDMI video for Vahana

- **Elmo QBIC rigs (4 sensors)³.** QBIC Panorama X camera rig enables to combine 4 QBIC cameras for capturing omnidirectional video streams. Camera is equipped with CMOS sensor and supports resolution up to Full HD in 60p. (recording) or 30 fps (HDMI output). The camera has WiFi output.

Camera Rig	Number of sensors	Output resolution	How it is used
 <p>Elmo QBIC</p>	4	4x 1920x1080p/60 fps when recording on SD card 4x 1920x1080p/30 fps on HDMI output	Outputs video files on SD cards for VideoStitch Studio, or HDMI video for Vahana

4.2.2.2. Off-line stitching system

For the stitching process, the “VideoStitch Studio” and “Autopano” software products will be used.

VideoStitch Studio for off-line stitching and production accepts input video files from cameras encoded with AVC, Baseline/Main/High profiles, 8 bits, progressive format, or Apple ProRes 10 bits (progressive). It can produce output stream encoded in MPEG4/AVC/Apple ProRes 10 bits or TIFF/JPEG/PNG image sequences. The requirements for VideoStitch Studio are: Windows 7 or later, 64 bits, Linux Ubuntu 12.04.4 64 bits, Mac OSX 10.9 or later. Regardless of the OS, an nVidia graphics card with 4GB of GPU memory is needed.

For the production of omnidirectional movie by Lightbox and regarding compatibility with existing workflow, AutoPano Video can be also used. Lightbox uses full Mac environment workflow with no hardware support for VideoStitch. Both products VideoStitch Studio and AutoPano will be tested and used for production purposes.

³ <http://www.video-stitch.com/360-camera-rigs-elmo/>

The workflow for AutoPano is quite simple. It recognises the camera rig automatically and all parameters are determined based on imported footage, including the number of cameras and the angle of the lens, however for fine tuning of all stitching videos, some control points must be introduced (marking the same point in different cameras).

4.2.2.3. Directional cameras

As directional cameras, the production team may use Blackmagic Micro Studio Camera 4K. For the traditional footage these cameras were selected because there was a limitation on camera size. It is important to choose the smallest cameras with great detail. Blackmagic Micro Studio Camera 4K cameras have small footprint, high definition (4K), remote control (all parameters and configuration can be remotely change). Cameras use Micro Four Thirds Lens Mount, so it is easy to use a wide range of lens, already available in Lightbox. The size of cameras is also important to hide directional cameras from omnidirectional movie. Small 4K cameras can be mounted in the 360 rig.



Figure 6: Blackmagic Micro StudioCamera 4K

Blackmagic Micro Studio Camera 4K supports resolutions up to 3840x2160 (UHD) up to 30 fps. In lower resolutions - 1080p supports wide range of frame rate up to 60 fps. The camera has focus control and iris control and can be used for both off-line recording and live streaming (SDI or HDMI output).

4.2.3. Workflow

The input and output formats are closely related to the cameras selected for content recording on the one hand, and VideoStitch software capabilities on the other hand.

The overall functionality consists of stitching various off-line input video sources. VideoStitch Studio uses following workflow:

- **read the input sources** coming from the multiple cameras on the 360° camera rig
- **temporally synchronize** them (VideoStitch Studio only, Vahana VR assumes the frames on its physical or network input ports are already synchronized)
- **calibrate the camera rig geometry** (through self-calibration of intrinsics (camera focal lengths and distortion parameters) and extrinsics (camera position and orientation) parameters, but an offline calibration template can be imported)
- **calibrate the camera rig photometry** (to make up for various exposures, colour temperatures if the cameras are not properly controlled, and for lens vignetting)
- **map each camera view onto a 360° equirectangular frame**
- **adjust the equirectangular frame orientation for horizon levelling** (making sure the scene horizon maps to an horizontal line in the stitched output)

- **export the stitched content.** In Studio, it can take the form of individual picture files, or a compressed video stream. In Vahana VR, the output format can be a compressed video file stored on disk, or and uncompressed output on an HDMI or SDI port, or a compressed RTMP stream which can be sent to a video server.

The workflow described above is depicted on **Figure 7**.

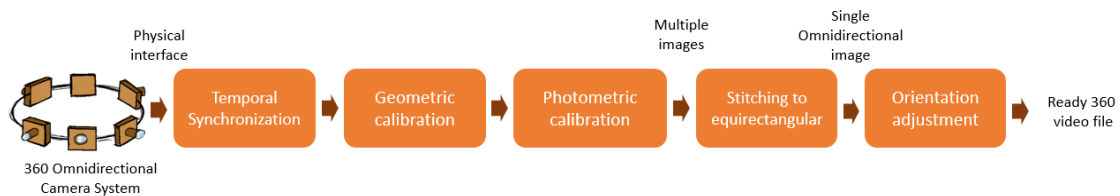


Figure 7: Details of capture and stitching workflow using VideoStitch Studio and Vahana VR

Only progressive video is supported by the stitching software, interlaced video cannot be stitched correctly without a deinterlacer.

Resolution is only limited by the computing power of the stitching device and available input cards.

4.3. Off-line production Tools

4.3.1. Description

Video editing in general is a complex process with many stages. In the project we extend typical media creation by adding new technical possibilities, however they also impose some restrictions for the content creation process.

The aim of this part of the project is to propose a workflow, elaborate techniques in existing and create missing tools that will allow users to create, in an easy and intuitive way, a wide variety of content suitable for the ImmersiaTV player.

The architecture of a set of tools for content creation is based on three elements:

- software requirements from the user scenarios (defined in Deliverable 2.3),
- output format and player capabilities (defined in 4.6 Reception, Interaction and Display),
- evolution of environmental capabilities.

While the first one defines requirements, the two others impose mostly limitations that have to be taken into account.

The proposed workflow should not differ much from the typical process of video editing. Editors should be able to follow their standard routine supplemented with just a few easy additional steps and having in mind that sequences for three synchronized output destinations have to be prepared. All of the required additional functionality will be implemented and added to Adobe Premiere Pro as a set of plugins.

There are three main stages added to the standard editing workflow, as depicted in Figure 6:

- synchronization of media for different output destinations
- defining portals/transitions/interactive points

- export to different output formats

By adding three stages (depicted as yellow blocks) to the typical content edition workflow (dark blue), separate processes for each device (TV, tablet, HMD) are merged into one bigger process.

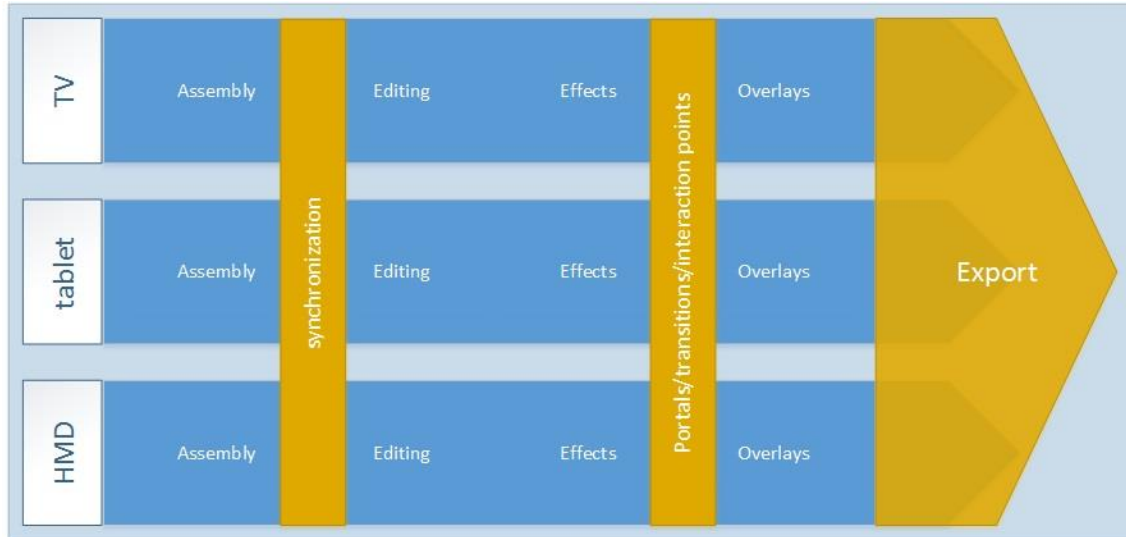


Figure 8: Schema of proposed editor workflow

4.3.2. Software architecture

4.3.2.1. Synchronization

The first challenge is assembly and editing of content for three devices in one project. It is up to editors, if they create them in parallel or one by one, but the objectives of the project put strict requirements for the synchronization of output media. To achieve this goal, at this point of the workflow editors have to follow basic rules allowing to use the ImmersiaTV plugin. In particular, clips for all types of devices should be edited together in a common sequence space, however on separate video tracks (layers). Each track will have to be labelled, indicating which device is the target. Tracks can be shared by more devices, if the same clips are intended for them.

This approach from editor perspective makes assembly, synchronization and editing of content for three devices similar to preparing picture-in-picture content for a single device. Generation of three separate contents will be done automatically in the export stage.

After tests editors may ask for additional tools to support labelling and managing tracks or enhancing preview capabilities, but these will be optional improvements, which will not influence general architecture.

4.3.2.2. Portal video effect

Portals are overlays (video or graphical) in omnidirectional content potentially with interactive options. In our approach inside of a portal there will be a separate video stream and the player will blend them dynamically. It can be used as a (conditional) transition, when the appearing portal covers the whole sphere.

To allow editors to create ImmersiaTV project with portals, there should be a Portal video effect implemented. It should be applied to the video clip that will be visible in the portal. It should

describe portal parameters and visualize them in a preview. We assume that background space is omnidirectional with equirectangular projection. During the export portals parameters will be used to generate proper video files and metadata.

This part of the plugin will be created with After Effects CC 2015 Plug-in SDK (C++)⁴.

Portal parameters should be possible to be modified from standard Effects Control panel and on a preview. Table 3 lists all parameters of the *Portal video effect*.

Parameters	Control type	Animatable (Variable)	Details
Projection	list	no	none - Directive shots equirectangular - Omnidirectional shots
Shape	implicit	no	rectangle - Directive shots spherical cap - Omnidirectional shots
Reference	list	no	world/user
Longitude	360° (-180°-180°)	yes	position of a centre of a portal relative to 'Reference'
Latitude	180° (-90° - 90°)	yes	position of a centre of a portal relative to 'Reference'
Distance	implicit	no	Track number defines order of portals
Size	slider / implicit	yes	Directive shots - scales width of video to a defined size (keeping aspect ratio). 1.0 -> width of a background sphere Omnidirectional shots - always 1.0 -> 360°
Luma matte	layer list	no	Layer defining transparency of a portal video (NOT related to the transition)
Transition:			
Visible	checkbox	yes	Possibility to switch on/off portal at keyframes.
Transition luma matte	layer list	no	Layer defining portal opening/closing transition
Additional action	list	yes	Action at a keyframe (currently only vibration)
Interactivity:			
condition on appearance	list	no	List of callbacks (click on, look at portal, shake the tablet, etc.)
condition on transition pause	list	no	List of callbacks (click on, look at portal, shake the tablet, etc.). Only affects the luma playout, not the content playout
condition on completion	list	no	List of callbacks (click on, look at portal, shake the tablet, etc.).
Separate switch	checkbox	no	Interactive area is different from portal content area (for tablets)

⁴ <http://www.adobe.com/devnet/aftereffects/sdk/cc2015.html>

Switch longitude	-180° - 180°	yes	Position of interactive area
Switch latitude	-90° - 90°	yes	Position of interactive area
Switch width	slider	yes	Width of interactive area
Switch height	slider	yes	Height of interactive area
Switch reference	list	no	world/user

Table 5: Portal video effect parameters

4.3.2.3. Visualisation

A preview of an edited scene can be observed in the Program Monitor window. Selecting the Portal effect in the Effect Control window enables overlay in the preview, that visualizes parameters of a portal and allows their direct modification.

Additionally the Preview mode parameter of the Portal effect allows to change how video is rendered. There are 5 options in the Table 4:

Preview mode	Description
Outside	portal is not visible, only background
Inside	portal video is visible, without any modifications (projection, luma matte),
Luma matte	luma matte layer of a portal is visible
Combine	portal is composed into the background

Table 6: Preview mode parameters

4.3.2.4. ImmersiaTV package export

The final step of content creation is the generation of an ImmersiaTV package containing a set of media files and metadata describing their relations. When the user follows the workflow described earlier in this chapter, export to the ImmersiaTV format should be done automatically.

It will be accomplished by a Javascript script run from a ImmersiaTV panel plugin (based on Adobe Premiere Pro CC 2015 panel SDK⁵). A panel is a HTML document, which can use standard Javascript, additional libraries (e.g jQuery) and can interact with Premiere Pro API. It will include a form to specify export parameters and to launch the export process.

The export process will have three main stages:

1. **project analysis** - tracks' labels and portal video effects' parameters should allow to determine the structure of the final package.
2. **metadata generation** - from the results of the project analysis an XML file with metadata described in chapter 4.6.4.2 Metadata should be created.

⁵ <https://github.com/Adobe-CEP/Samples/tree/master/PProPanel>

3. **media files rendering and encoding** - an export of each of the media streams as defined in the metadata should be started.

The only required export parameter will be the device type (output path).

Additionally encoding parameters for each target device could be parameterized, however in first versions of a plugin they will be fixed.

4.3.3. Workflow

Data acquired in previous modules are the input for production tools. Adobe Premiere Pro natively supports the formats described in the previous chapters.

Tools in this module will produce a package of files (media files and metadata) ready for distribution and compatible with ImmersiaTV player as depicted in Figure 9 .

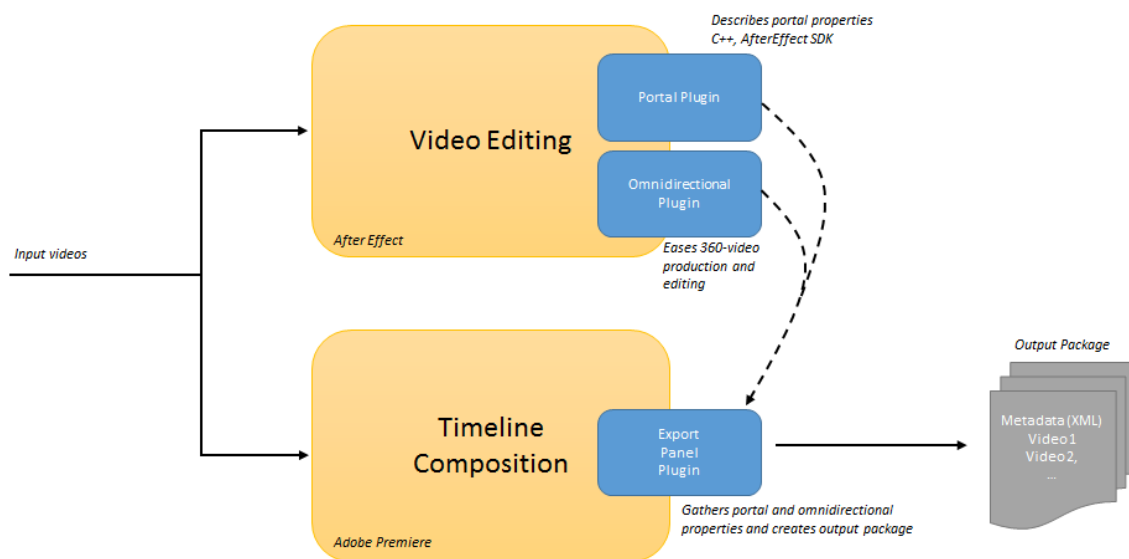


Figure 9: Composition of the off-line production tools.

4.4. Encoding

4.4.1. Description

The main functionality of the module is to provide an efficient and high quality encoding solution compatible with the general purpose equipment (TV, HMD, tablets) available on the market. At the moment, the most popular and widely supported codec in the consumer area is H.264/AVC and it was chosen as a main solution for the Pilot 1 demonstration.

The Encoding module is responsible for implementation of video encoder and decoder functionality used by other modules of the ImmersiaTV system. As the general standard selected for **first iteration and pilot is H.264/AVC**, this section provides the guidelines regarding the video coding format and underlying parameters to be used within ImmersiaTV from capture to rendering and display, as well as for all other processing and streaming related matters such as corrections, stitching, transcoding, etc.

The main objective of this specification is to ensure that the coding format in the complete chain of processing from creation to consumption is clear and as much as possible harmonized in order to reduce the number of transcoding and format conversions needed. Not only the quality of the content but also complexity issues arise if transcoding should be performed between different components in the processing chain.

4.4.2. Architecture

4.4.2.1. Codec definition

In principle, the highest quality content should be produced when generating content for off line configurations (pilot 1). Therefore, the highest possible bit rate, frame rate, frame resolution, and colour sampling must be achieved. Where possible, the influence of compression on the quality should be minimized. It is however a fact that several existing devices and hardware do not allow for uncompressed content. Based on the feedback received from ImmersiaTV partners, for off-line recording of content, the following guidelines are followed as far as the video format is concerned. Parameters of the codec are summarized in Table 7 below.

Parameter	Value	Comments
Encoder	H.264/AVC	Uncompressed where possible
Decoder	H.264/AVC	Compressed bitstream should be decodable by any widely used and H.264/AVC compliant decoder such as ffmpeg.
Profile	At least HiProfile (HiP)	If the capture device does not allow, Main Profile is acceptable but not recommended.
Bit rate control	Variable rate	If constraints do not allow, constant rate is acceptable but not recommended
Color Space	RGB, YCbCr, YUV	YCbCr recommended
Fame/Field sampling	Progressive	Interlaced should be avoided
Color sampling	4:2:2	4:2:0 is acceptable if 4:2:2 not possible
Frame resolution	4K, UHD, HD	The higher the resolution, the better
Frame rate	At least 30Hz	If capture device does not allow, then 25Hz is also acceptable but not recommended.
Bit depth per component	10bit (preferable) 8bits (acceptable)	The higher the better
Bit rate (total)	At least 10 Mbps per camera	The total bitrate corresponding to the number of cameras may arise various issues regarding their capture and could affect the 10Mbps lower limit.

Table 7: Off-line recording configuration

4.4.3. Workflow

The input to the Encoding modules is defined by the output formats supported by video acquisition equipment and Production tools, it can be raw data or pre-encoded video files. As the output the H.264 encoded video files or streams are generated.

The Encoding module has direct interaction with Capture block (Input side), Production tools (Input/Output side) and Content distribution (Output side) modules but the format and results of the encoding procedure influence all the stages in the ImmersiaTV processing chain.

The streaming of video is largely dependent on the available HD and infrastructure capabilities. A transcoding step will be necessary to convert the recorded content and to match it to what is possible.

On the reception site the decoder is responsible for decoding of H.264 encoded media streams received by the client player and sending raw data to the rendering module. Regarding decoding functionality of Reception, Interaction and Display module the rendering and display of video is largely dependent on the available HD and display capabilities. The final parameters of the encoding procedure will depend on the capabilities of both the streaming solution (see below) and display devices. As a general rule, the video format, bit rate, frame and rate will depend on the capabilities of a general purpose display as available on a typical HMD, a tablet or a smartphone.

4.5. Content Distribution

4.5.1. Description

This section specifies Content distribution mechanisms and deals with the transmission of all data (media and metadata) from the main server where the content is stored, through the Wide Area Network (Internet), and up to the user's Local Area Network, where synchronized playback among devices will take place. For multimedia delivery, or the ImmersiaTV content delivery, the streaming technique that will be used is the MPEG-DASH standard. MPEG-DASH is a recent standard, officially published in 2012⁶, and reviewed in 2014⁷. DASH is the acronym of Dynamic Adaptive Streaming over HTTP so which clearly denotes two of its main goals: being adaptive and use of HTTP as the network protocol.

4.5.2. Software architecture

This functional block handles communication between offline encoded content or live streams and the end-user's player. It encapsulates selected video streams into network protocols and provides synchronized video and auxiliary streams to the player.

MPEG-DASH emerged aiming to be reference standard for Segmented HTTP techniques, as before MPEG-DASH there were only proprietary or private approaches, like HLS (HTTP Live

⁶ http://standards.iso.org/ittf/PubliclyAvailableStandards/c057623_ISO_IEC_23009-1_2012.zip

⁷ http://standards.iso.org/ittf/PubliclyAvailableStandards/c065274_ISO_IEC_23009-1_2014.zip

Streaming) from Apple (it has also being published as an IETF draft⁸), HDS (HTTP Dynamic Streaming) from Adobe or MSS (Microsoft Smooth Streaming) from Microsoft. All of them are HTTP based and adaptive solutions, but MPEG-DASH appears to be the only option that might get a wide adoption in the industry, as many of the main industrial actors already announced support to it (Microsoft, Adobe, Netflix, Google, etc.⁹). There are three main reasons to choose MPEG-DASH as the standard to follow in the ImmersiaTV project:

1. MPEG-DASH is getting adopted by the major players. This is a very important point in order to get ImmersiaTV close to the market. Ideally, the content providers using mature MPEG-DASH services would not need to drastically update their distribution scheme in order to provide immersive experiences.
2. It is based on HTTP which means it is easily supported by many CDN services that operate over the top and by any platform or infrastructure adapted for web content (i.e. mobile networks).
3. It is an adaptive standard. Being adaptive might be of special interest in ImmersiaTV as the project will handle different devices, screens and resolutions. In 360° video resolutions up to 4K must be considered, however there might be some client devices (i.e. tablets or smartphones) that are not capable of handling these high resolutions; in that case adaptation is useful. If via MPEG-DASH the server is offering a simpler version of the same content, a limited client might use it and be able to provide a lower quality experience instead of failing to provide any service or experience at all. Although, the quality can't be too low to prevent side effects of omnidirectional video feeling and perceiving. The right trade-off between the quality and effectiveness will be a subject of QoE feedback.

MPEG-DASH has been already used and tested with 360° immersive video nearly out of the box (just providing a specific player for 360, see for example a demo by BitMovin¹⁰).

MPEG-DASH is not a protocol, format neither a codec. As stated before it is an streaming technique that makes use of HTTP protocol, however it is codec and format agnostic. MPEG-DASH does not solve the HTML5 codec issues and does not describe the details about how a specific codec and container could be used in a way the result is MPEG-DASH compliant. In order to fill the gap the DASH Industry Forum provided several guidelines¹¹ about how to use MPEG-DASH together with H264 or HEVC codecs and ISO-BMFF container format (ISO/IEC 14496-12)¹². ImmersiaTV will stay as close as possible to those specifications.

The project is something more than just providing a single 360° video, as it has been already said, the project aims to provide a transversal immersive experience across devices. The impact is that there will be several synchronized streams, and all of them will be contextualized with the metadata defined in 2.5.2.1 This metadata will be delivered using a simple HTTP download, to use the same mechanism as the media.

⁸ <https://tools.ietf.org/html/draft-pantos-http-live-streaming-13v>

⁹ <http://dashif.org/members/v>

¹⁰ <http://www.dash-player.com/demo/adaptive-vr-360-video-html5-demo/>

¹¹ <http://dashif.org/guidelines/>

¹² http://standards.iso.org/ittf/PubliclyAvailableStandards/c068960_ISO_IEC_14496-12_2015.zip

4.5.3. Workflow

The content distribution module as a communication service interacts mainly with two other modules. On the input side with Production tools including the encoding module which provides H.264 encoded content (video files or live streams) as well as the metadata. The data gathered from production tools are further encapsulated according to MPEG-DASH standard.

On the output site Content distribution module communicates with the receiver module included in the Reception, Interaction and Display functional block which receives H.264 encoded MPEG-DASH media streams (with metadata) distributed by the streaming server over the HTTP protocol. This module also deals with user feedback and passes requests generated by the player to the streaming servers in order to provide an adaptive approach and support the region of interests mechanism.

4.6. Reception, Interaction and Display

4.6.1. Description

This deals with the reception of the streams from the Wide Area Network (Internet), their redistribution in a local area network, and the integration with the interactive input of the end-user.

The player integrates the audio, video and data streams in a coherent omnidirectional scene, parses the user input and adapts the environment appropriately to the reactions expected. Examples of touch-based interaction include:

1. Browsing and selecting a specific content.
2. Starting/stopping the experience.
3. Selecting a region of an omnidirectional video to share through social media.
4. Zooming in or out.

Examples of interaction based on movements include:

1. Moving the head in an immersive display should update consistently the portion of the omnidirectional video being displayed, to reflect basic sensorimotor correlations.
2. Moving the tablet around should also enable the update of the field of view.

Audio is played consistently across different devices, either in stereo (TV), either in binaural format (tablet with headphones, as well as google cardboard and head mounted displays).

The chosen architecture involves two different kinds of connected devices, which synchronize and interact:

- Receiver devices (TV Set, HMD, Tablet).
- Session Management device.

The receiver devices run the ImmersiaTV interaction and display software (in short, the ImmersiaTV player). This software is a multi-platform player targeting the general consumer. Consistently, this player is designed to be compatible with emerging broadcast synchronisation

standards (like HbbTV 2.0¹³), and work on the main platforms available to deliver the ImmersiaTV experience.

The session manager is a device connected to the same local network as the players, which coordinates playback among them. It makes sure that all players synchronize to the same clock and watch the same content, among other things. All functions provided by the session manager can be integrated into the players, so any one of them can act as the session manager, thus removing the need for an additional device on the network.

The Figure 10 shows a diagram of the connectivity of all devices in the home network. Conceptual blocks inside each device are shown simplified. A more detailed view is given in the next section.

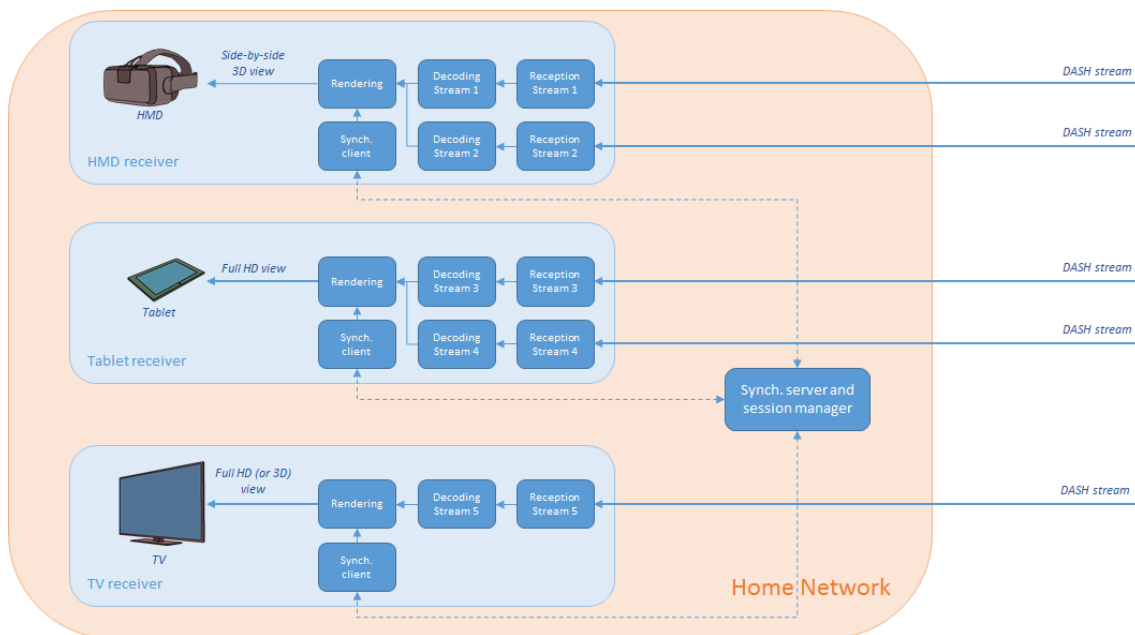


Figure 10: Connectivity of devices in home network

4.6.2. Software architecture

The ImmersiaTV player running on the receiver devices is based on the Unity3D¹⁴ engine. This greatly simplifies deployment on a wide variety of end-user devices and adapts the experience to the particular characteristics of each device.

The processing of the media streams is performed using the GStreamer¹⁵ open-source framework. It receives and decodes different audio and video streams and delivers resulting frames to Unity3D for rendering. The connection between GStreamer and Unity is performed by a plugin developed within the ImmersiaTV project, called GStreamer Unity Bridge (GUB for short)¹⁶ and publicly available.

¹³

http://www.etsi.org/deliver/etsi_ts%5C102700_102799%5C102796%5C01.03.01_60%5Cts_102796v010301p.pdf

¹⁴ <https://unity3d.com>

¹⁵ <http://gstreamer.freedesktop.org/>

¹⁶ <https://github.com/ua-i2cat/gst-unity-bridge>

The Figure 11 depicts the complete software architecture (including both kinds of devices), with the most important blocks detailed in next sections.

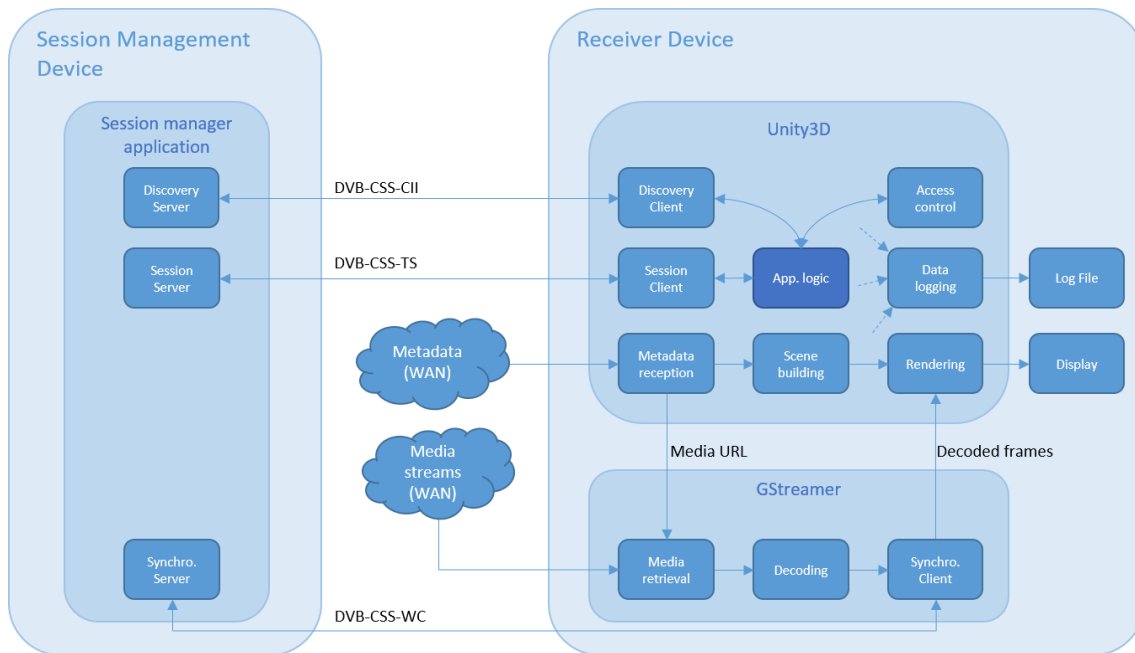


Figure 11: Software architecture of a player

4.6.3. Session management device

This is a device on the same local network as the players which coordinates the distributed playback experience. Initially it will be an application independent from the other players, running on a separate machine. The goal, however, is to integrate it with the player application, so any player can act as Session Manager, simplifying the setup for the user.

Its main functions are:

- Ensure all connected players see the same content
- When a player connects to a running session (there are previous players watching the same content already), it “catches up”, starting the playback at the point where the other players are.

More precisely, it provides:

- A master clock
- Session management: Distributing the BaseTime (the wall-clock time at which content playback started) and counting connected clients
- The media location (the remote MPD file URI)
- A discovery mechanism so the clients do not need to know the IP of the server.
- Optionally, a cache for the media files. Since many players might require the same content, huge bandwidth gains can be achieved by using a local media cache. The session manager, when running on a dedicated machine on the network, is the ideal location for this cache.

Communication between the Session Manager and the clients will be based on protocols from the DVB-CSS (Digital Video Broadcasting – Companion Screens & Streams) family to ease eventual interoperability with HbbTV 2.0 devices.

4.6.3.1. Discovery

To avoid having to provide each client with the server's address, the DVB-CSS-DA (Discovery and Association) discovery protocol will be used, in conjunction with the DVB-CSS-CII (Content Identification and Information) protocol. Combined, they provide the entry points for the other protocols and features (DVB-CSS-TS, DVB-CSS-WC and media location). DVB-CSS-DA uses the UPnP protocol, so there are plenty of available software libraries to aid its implementation.

The DVB-CSS-CII protocol will also be used to provide all players with the URL of the metadata file describing the scene, so this URL only needs to be stored in one place and can be easily changed.

Furthermore, it will easily allow caching, if this URL points to a local HTTP server in the same machine, for example.

The DVB-CSS-CII is very well suited for this task. However, it only provides a contentID which needs to be looked up in a Media Resolution Server (MRS) through HTTP to obtain the media URL. To ease implementation and reduce the requirements of the devices, we will embed the media URL in the CII response using private data, as already foreseen in the CII specification. This protocol uses JSON+WebSockets.

4.6.3.2. Session Server

This block will count the number of connected players and give each one the base time when they connect. When the first player connects, the base time is set to the current wall-clock time (so the clip starts from the beginning). Following players will see the clip has already started. When the last player disconnects base time is reset.

This functionality is very similar to the MSAS unit in DVB-CSS. Communication therefore will resemble the DVB-CSS-TS protocol, with the clients requesting a session through JSON+WebSockets, and the server replying with the current media time (base time). There are available C libraries to help the development:

- <https://libwebsockets.org/index.html>
- <http://www.json.org>

4.6.3.3. Synchronization

Multi-device synchronized playback should use standard protocols to achieve maximum interoperability. Particularly, to support [HbbTV 2.0](#)¹⁷ devices, the [DVB-CSS](#)¹⁸ (Digital Video Broadcast – Companion Screen & Streams) protocols family has been selected. The DVB-CSS-WC (Wall Clock) protocol is interesting, since it synchronizes the clock of all devices, so they all provide the same time.

GStreamer, though, lacks support for DVB-CSS-WC synchronization which needs to be added. This library is modular and plugin-based by design and could be used in implementation. Also,

¹⁷ www.hbbtv.org

¹⁸ www.dvb.org/resources/public/factsheets/dvb-css_factsheet.pdf

this allows contributing the work back to the GStreamer open source community, extending the project's dissemination.

There are elements already in GStreamer that allow inter-device synchronization, although they use a different protocol and therefore cannot be directly used. Their code, however, can serve as basis to implement support for DVB-CSS-WC. These elements are [GstNetTimeProvider](#)¹⁹ and [GstNetClientClock](#)²⁰. For a usage example, take a look at the [gst-rtsp-server](#)'s²¹ [test-netclock](#)²² and [test-netclock-client](#)²³.

This work will provide two new GStreamer libraries, a DVB-CSS-WC Server and a DVB-CSS-WC Client, which will perform the same functions as the already present [GstNetTimeProvider](#) and [GstNetClientClock](#), using the DVB-CSS-WC protocol.

The source code will be hosted in a GIT repository, forked from GStreamer, to ease contributing back to the original project. The library will be written in C and follow the GStreamer naming conventions. There is an open source Python implementation by the BBC of the DVB-CSS-WC protocol available [here](#)²⁴ which can also be used to help implementation.

These libraries will then be used by the Synchronization Server and Synchronization Client blocks.

4.6.4. Receiver devices

These are the players which display the immersive content to the user. They are programmed using the Unity3D game engine to allow interoperability on a wide range of devices. Therefore, most of the software modules are made in C#, with occasional calls to C when needed (for GStreamer operation, for example).

This is the normal operation of an ImmersiaTV player:

- Upon powering on, the player tries to discover a Session Manager on the network using the Discovery Client. It receives from the Session Manager entry points for the rest of protocols and the URL of the media being played.
- The obtained URL points to a metadata file describing the scene, which needs to be downloaded (by the Metadata reception module).
- The metadata is parsed and used to build the scene inside Unity3D (Scene building module). The parts of the scene which require displaying media will instantiate GUB objects as required (which, in turn, will instantiate GStreamer pipelines) and provide them with the appropriate media URL.
- The player starts synchronizing its internal clock to the remote master clock using the Synchronization Client.

¹⁹ gstreamer.freedesktop.org/data/doc/gstreamer/head/gstreamer-libraries/html/GstNetTimeProvider.html

²⁰ gstreamer.freedesktop.org/data/doc/gstreamer/head/gstreamer-libraries/html/GstNetClientClock.html

²¹ git.freedesktop.org/gstreamer/gst-rtsp-server/tree

²² git.freedesktop.org/gstreamer/gst-rtsp-server/tree/examples/test-netclock.c

²³ git.freedesktop.org/gstreamer/gst-rtsp-server/tree/examples/test-netclock-client.c

²⁴ bbc.github.io/pydvbcss/docs/latest/wc.html

- The player instantiates a Session Client which will inform the Session Manager that a new client is connected, and in return, it will obtain the Base Time (This is, the wall clock time at which playback of the current media started).
- Unity3D will take care of rendering the scene onto the display.
- During the whole session, the Data Logging block can retrieve information from any module and produce a log file, to monitor the Quality of Experience.
- For testing sessions in which a questionnaire must be filled in before the experience, the Access Control module ensures all information has been received before starting playback.

Modules whose function is not clear from previous descriptions are described next.

4.6.4.1. Session Client

From the client perspective, this module is only needed to retrieve the current Base Time so it knows at which point in the media playback has to start. This is done through the DVB-CSS-TS protocol (only a small subset of it will actually be required). This small DVB-CSS-TS interaction, though, is more interesting for the server, since it allows it to count the number of connected clients.

Session Clients will periodically poll the server, as the protocol states, and this will also allow the server to know when a client has been disconnected (via a timeout mechanism).

Also, each Session Client must have a unique ID (unique within the local network) so the server can keep track of them.

4.6.4.2. Metadata

The ImmersiaTV experience is based on 2 ideas: synchronized content across devices, and portals allowing interaction by blending different scenes, taken as traditional and omnidirectional shots in an immersive display.

All the metadata in ImmersiaTV will be sent in XML format, in a dedicated file. An event mechanism will be used, so the metadata can be added, removed or updated at pre-established times. Interaction is also defined in the XML file so metadata can be changed in response to user actions.

The ImmersiaTV Scene

The basic ImmersiaTV container will be called a Scene. Each Scene can contain the following elements:

- A unique string identifier (mandatory)
- A sequence of Shapes (defined below) showing some type of media
- A pointer to a CGI scene, containing additional geometry, textures, methods and other elements that may be involved in the scene rendering

The ImmersiaTV Shape

Each Shape can contain the following metadata:

- A unique string identifier (mandatory)
- The geometry describing this shape (rectangle, sphere, ...) and its size
- A list of Anchors (defined below) describing how this shape is to be situated in the scene, and a method to merge the different Anchors.

- A series of media file names, indicating the texture and optional transparency masks to render on the shape.
- A description of the projection used to turn omnidirectional media into a conventional flat video stream (if any).
- Cropping parameters, if desired

The ImmersiaTV Anchor

Anchors are points in the scene used to place Shapes. In the preparation towards pilot 1, each shape will have only one Anchor. Each anchor will contain the following metadata:

- A unique string identifier (mandatory)
- A reference frame (either the world or the camera)
- The polar coordinates (longitude, latitude and distance from the camera)
- A weight, used for merging different Anchors (not used in pilot 1)

The complete ImmersiaTV metadata specification

The above requirements have been implemented as an XML Schema Definition file (XSD), available at the following URL:

http://server.immersiatv.eu/public_http/metadata/ImmersiaTV.xsd

This definition is precise, allows checking for validity of XML files, and contains documentation, which can be processed to generate a human-friendly format, like the online pages available here:

http://server.immersiatv.eu/public_http/metadata/ImmersiaTV.html

The URL of the XSD file can be used in XML files so they can be validated automatically.

Metadata callback specification

The metadata format defined above specifies some methods on the player to be called after some user actions. The list of available methods is purposely left out of the definition of the metadata in order to render it more generally, and to ease expanding this list. The following table describes the accepted values for these callbacks (used, for example in the `onActivate` and `onDeactivate` attributes):

<code>toggleVisibility, shapeId</code>	Toggles the visibility of the shape with the indicated Id.
<code>setVisibility, shapeId</code>	Makes the shape with the indicated Id visible .
<code>unsetVisibility, shapeId</code>	Makes the shape with the indicated Id invisible .
<code>playTransition, shapeId</code>	Starts playing the transition indicated with the <code>transitionFile</code> attribute, on the shape with the indicated Id.
<code>pauseTransition, shapeId</code>	Pauses the transition indicated with the <code>transitionFile</code> attribute, on the shape with the indicated Id.
<code>playLimitedTransition, shapeId, seconds</code>	Starts playing the transition indicated with the <code>transitionFile</code> attribute, on the shape with the indicated

	Id, for the limited amount of time indicated in the mandatory <i>seconds</i> parameter.
--	---

Table 8: Accepted values for call back

The *shapeId* parameter is always optional. If no Id is given (the parameter is missing), the command affects the shape with the attribute.

A sample XML file containing ImmersiaTV metadata

The following is a sample ImmersiaTV metadata file (Figure 12). Please note the usage of `xsi:schemaLocation` in the root node to import the XML Schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<ITVEvents xmlns="urn:immersiatv:immersiatv01:2016:xml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:immersiatv:immersiatv01:2016:xml
    http://server.immersiatv.eu/public_http/metadata/ImmersiaTV.xsd"
  type="static">
  <DefineScene id="0" device="hmd" externalScene="TestScene1.unity"
    time="0">
    <DefineShape id="0" type="rectangle" anchorMethod="simple"
      mediaFile="FILE1"
      mediaProjection="none" mediaCropX="0" mediaCropY="0"
      mediaCropWidth="1"
      mediaCropHeight="1" maskFile="FILE1MASK">
      <Anchor id="0" referenceFrame="world" longitude="0" latitude="0"
        distance="0.5" weight="0.8" maxAngularDeviation="45" />
      <Anchor id="1" referenceFrame="user" longitude="45" latitude="10"
        distance="0.5" weight="0.2" />
    </DefineShape>
    <DefineShape id="1" type="point" mediaFile="FILE2" />
    <DefineShape id="2" type="sphericalCap" size="1" mediaFile="FILE1"
      mediaProjection="equirectangular"
      onActivate="toggleVisibility,theatre_screen" />
    <DefineShape id="theatre_screen" type="mesh" mediaFile="FILE0"
      transitionFile="TRANSITION0" transitionState="paused"
      onActivate="playTransition" onDeactivate="pauseTransition" />
  </DefineScene>
  <DefineScene id="0" time="10">
    <DefineShape id="0">
      <Anchor id="0" latitude="10" />
    </DefineShape>
    <RemoveShape id="1" />
  </DefineScene>
</ITVEvents>
```

Figure 12: Sample ImmersiaTV metadata file.

This file defines two events: One creating a new scene and one updating it.

The Scene initially contains 4 Shapes: One rectangle with two anchors and a planar video with a mask, one point, one spherical cap with an equirectangular omnidirectional video and one external mesh with a planar video (because this is the default value). Moreover, the spherical shape (id "2") can toggle the visibility of the external mesh shape (id "theatre_screen") through user interaction. Also, the external mesh shape has a transition mask, which starts paused and can be played or paused through user interaction.

The second event triggers 10 seconds after the scene starts, and updates the latitude of one of the anchors of the first Shape and removes the second Shape.

4.6.4.3. Metadata Reception

The metadata describing the scene contents, including media and interactions, will be contained in an XML file hosted on a remote server. The Metadata Reception module has to retrieve this file, through a simple HTTP GET request and provide it to the rest of the player.

The only input to this module is the URL of the XML file, which will be provided by the application logic, after retrieving it from the Discovery Client.

4.6.4.4. Access control

Some of the tests require that the users fill in a questionnaire before the experience can start. This questionnaire will be online and, upon completion, will provide a token (an alphanumeric string, for example). The first screen in the player must ask for this token, which then needs to be validated against a remote database. Only tokens which have an associated questionnaire stored in the database will allow entering the experience.

4.6.4.5. Data logging

In order to allow integration testing and offline analysis of the events taking place in the system, there is a data logging module present in the system.

Logged data can include static information like device, session and user characteristics, or dynamic information like user view direction, network state, CPU usage, frame losses or the momentary value of bitrate.

The module enables storing events in the push-like manner, where interested parties call methods of the logging module's API. The module then writes the logs to a file or a database.

After the session is logged to a database, a post-processing phase may begin, where data is read, filtered, aggregated and analysed. The aim of the analysis is to identify and quantify any incorrect behaviour of the system, especially by catching transient events that are possible in any real-time system. Therefore it is vital for the logging module to provide all data necessary for offline analysis.

Architecture

The logging module architecture is composed of two main components

- *Logging Client Library*, which defines functions implementing a flexible logging interface available for other *ImmersiaTV* modules. The library is meant to be run on (and monitor) each device presenting stream.
- *Logging Server*, which is a central component storing the logs. The server can actually be any database server, relational or not, depending on the deployment requirements. The server is run in a network location that is accessible by all components of the system.

The general architecture of Logging module is depicted on Figure 13:

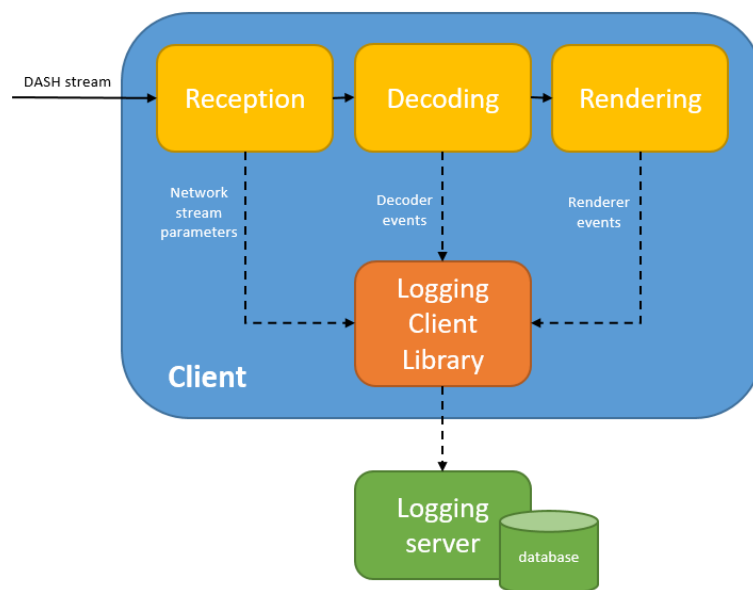


Figure 13: Logging module architecture and interactions

Workflow

All log data messages generated by different *ImmersiaTV* modules are stored in Log Database acting as a Logging Server component. Messages are collected in a database as a sequence of records.

Logged data includes static information such as device, session and user characteristics, or dynamic information such as user view direction, network state, CPU usage, frame losses or the selected adaptive bitrate.

All dynamic data is gathered periodically with frequencies adjusted to the dynamics of the observed processes. The data allows for offline analysis of all crucial processes. Thanks to this, the platform can be controlled and the proper presentation of streams can be ascertained.

Logging server

Logging server is a central element storing the logged data. It must be located on a network location that is accessible by all monitored devices. The server consists of a database along with its store, read and filter interfaces. The solution is flexible so any database implementation, relational or not, can be used, depending on the deployment requirements. The server is capable of handling several simultaneous log store operations.

Client library

Client library provides interface which supports logging mechanisms for all *ImmersiaTV* modules. The API is implemented as a shared libraries for C# (integrated with Unity 3D). Each class willing to log events can obtain an individual log pushing instance, which then can be used to issue the actual log commands. Each message has an individual level of importance assigned. The messages are supplemented by global parameters identifying a session and a device.

As the library runs on devices presenting video, it is especially important not to influence negatively the performance of the player, so to preserve the QoE. Therefore, the client logging library uses own threads to communicate with the database, not to block other threads of the application.

Logging data structure

The logging data have an open structure in order to ensure flexibility and extensibility. Each module can log messages of custom content, depending on the specific context.

Messages are collected in a database as a sequence of records and are identified by parameters like:

1. Parameters constant between sessions on a single device:
 - *user name*
 - *device ID*
2. Parameters constant only throughout single session:
 - *session ID*
3. Parameters dynamic within each session:
 - *local timestamp*
 - *network timestamp*
 - *level of importance*
 - *module name*
 - *message content*

As the log data messages are strongly related to the timestamps and time correlation between logged events is very important, it is required to ensure the same frame of time reference for all modules using logging mechanisms. Therefore the presence of a network timestamp, which is precisely synchronised between devices.

4.7. Quality of Experience

4.7.1. Description

Quality of Experience (QoE) represents the degree of delight or annoyance of the immersive visualization at the end-user's side. The immersiaTV project makes a distinction between QoE evaluations of the professional and residential end-users based on the acquisition of subjective data (T4.3), and QoE estimations using objective metrics that assign quality scores to audiovisual content by mimicking perceptual mechanisms based on training data (T3.7). The QoE module in the ImmersiaTV platform will be a piece of software that provides QoE estimations of the audiovisual content shown on the primary display device (tv screen) as well as the immersive display device (head-mounted display, smartphone, or tablet). These QoE estimations will be made available to other components in the ImmersiaTV platform, and can for example be used to steer parameters inside the codec.

4.7.1. Subjective metrics for quality assessment

To prepare the QoE module, the Phase 1 platform includes a data logging module (Section 4.6.4.6) and subjective data gathering procedure.

As part of the pilot evaluation and execution plan (D4.1), the following metrics were identified for measurement in pilot 1 and pilot 2 and will be gathered and processed during Phase 2.

Data type	Measurement of	Logging	Observation
Head-movement while wearing HMD	Frequency direction / angle	X	
Switching viewing angle in HMD	Frequency Which content	X	
Switching viewing angle on tablet/smartphone	Frequency Which content	X	
Multi-device usage	Which devices are used in combination How often do people switch between devices Length of each interaction		X
Use of portals	Which portals are being shown What user input has triggered some interactive behaviour of portals	X	

Table 9: Subjective metrics for QoE

5. PHASE 2 PLATFORM AND ARCHITECTURE

5.1. Architecture overview

The development plan in Phase 2 focuses on the implementation of the tools and modules required to demonstrate Pilot 2 and adapt the ImmersiaTV pipeline to a live scenario. The Pilot 2 architecture differs from Pilot 1 in technical aspects, so in this use case the omnidirectional and directive streams will be captured real-time by cameras capable to deliver a live video signal. Then the video will be processed and assembled in real-time by dedicated live stitching software (camera-specific stitching software, Vahana VR, AZilPix Studio.One). Additionally, semi-omnidirectional cameras are added, providing equirectangular views for use in HMDs or other VR viewing devices, but not offering a full spherical view. This allows to spend the capture pixel budget more intelligently, focusing better on where the action is, while significantly increasing the number of points of view. Finally all stitched omnidirectional, semi-omnidirectional and directional streams will be combined in the live production tool. Similar as in Pilot 1, the result of the live production will be several H.264 video files with MP4 format wrapper and one metadata file, transferred to the DASH server, transcoded, and served in MPEG-DASH streamable format to be consumed by the end user's devices. The architecture of the content distribution and content reception sides remains the same as in Phase 1. Therefore, we do not detail them again.

In the Pilot 2 workflow directional and omnidirectional videos are passed between components as live streams. Live production tools enables the process of editing and combining multiple video files in real-time. After live editing videos and XML metadata files are passed to distribution chain (MPEG-DASH server).

The general architecture of Pilot 2 is depicted on Figure 14.

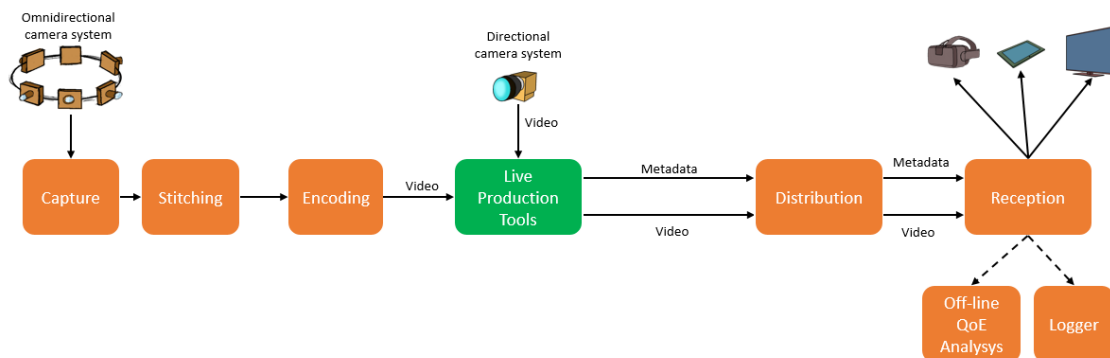


Figure 14: General architecture for Pilot 2

5.2. Live Capture and Stitching

5.2.1. Overview

While Pilot 1 used only “off-the-shelf” cameras that were available on the market at the time of production, Pilot 2 will use a variety of live camera systems in addition to off-the-shelf ones:

- Go Pro/Elmo rigs combined with Vahana VR;
- Integrated “Orah 4i” cameras developed by VideoStitch;
- a new distributed camera system developed by iMinds/EDM, commercialized in their AZilPix spin-off under the name Studio.One²⁵;
- and directional broadcast cameras: Grass Valley LDK 8000 cameras.

The GoPro/Elmo rigs have been introduced in pilot 1 already. Live stitching with these rigs is provided by Vahana VR, resulting in AVC video streams.

The integrated “Orah 4i” camera is a compact omnidirectional video rig developed by VideoStitch, with integrated motion sensor and ambisonics 3D spatial sound microphone. Stitching and encoding of the resulting equirectangular video stream is performed in the VideoStitch Orah stitching unit producing an AVC video stream.

The AZilPix Studio.One system is a video capture and production system that accommodates at the same time various models of 360 and panoramic camera rigs developed by iMinds/EDM and custom 12-megapixel block cameras equipped with a fish eye lens, providing a 170 degrees horizontal “semi-omnidirectional” field of view. These cameras have been found to be very useful in providing a multitude of points of views, while focusing pixel budget more intelligently on where the action is. They are typically used at the sides or the back of a concert stage. They produce equirectangular video, handled in the same way as that of full omnidirectional video. In short: they offer a better balance between the number of VR points of view from which action is captured, and the field of view at each point. All video processing including stitching is performed on an AZilPix Studio.One server, to which the cameras are connected. The output of the server consists of 4 HDMI, DVI or DisplayPort outputs, that can be fed into Vahana VR for encoding the already stitched video, via a SDI convertor at the Studio.One server and SDI capture board with Vahana VR.

These capture systems and their stitching solutions are presented in more detail in the next paragraphs.

The directional cameras used in pilot 2 (cyclocross) will be Grass Valley LDK 8000 tv cameras, which provide 1080i HD signals. Transport from the camera to the CCU in the central control room (OB truck) is based on triax cables.

Nevion fibre sets with break-out boxes having 4 SDI channels and 1 Ethernet channel, can transport the output streams to the live production tooling in the OB-VAN cabinet, either via IP (Orah, VahanaVR) or SDI (EDM camera system).

They will all be connected to the live production servers of section 5.3, either by RTMP network streams, or HDMI/SDI physical interfaces. Live Production Tool expect all streams to be RTMP and compressed, so interface can be through Encoding block/Vahana VR for EDM camera.

²⁵ See www.azilpix.com for description and pictures

The architecture of Capture modules are depicted on Figure 15

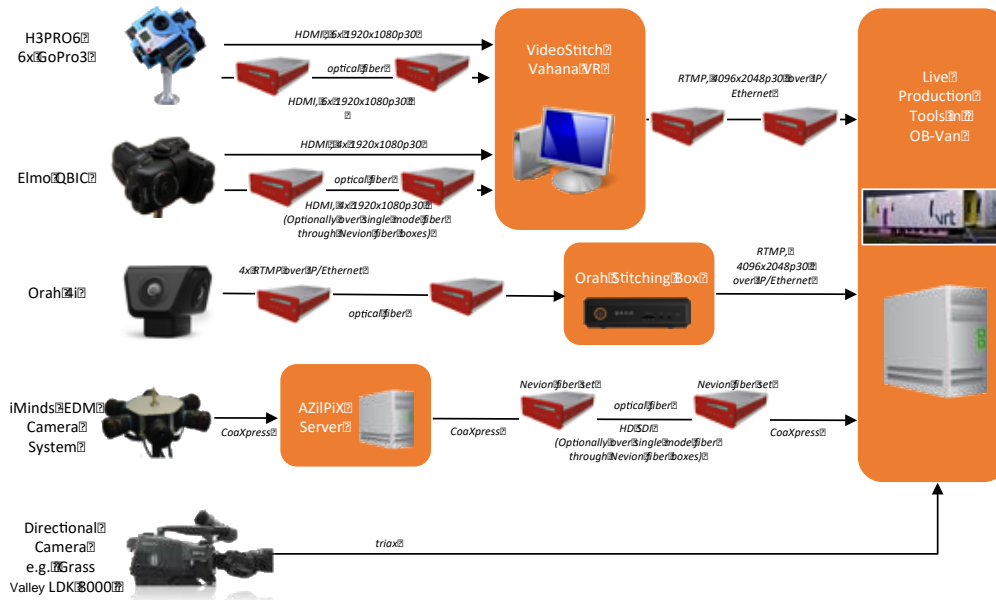


Figure 15: Architecture of the capture modules for Pilot 2

5.2.2. Omnidirectional camera systems

5.2.2.1. Orah 4i

The Orah 4i camera, depicted on Figure 16, is equipped with 4 sensors. Contrary to previous camera rigs used in Pilot 1, Orah is an integrated camera in a single housing and does not require an additional rig for mounting several camera modules. It is equipped with 4 integrated lenses and SONY EXMOR sensors as well as Ethernet output, and does not require additional stitching software such as Vahana VR or Videostitch Studio: stitching is done by the Stitching Box connected to it (i.e. the camera control unit, which takes the form of a mini PC). Maximum resolution for stitched output is 4K in 30p. The output AVC video bit rate transmitted by RTMP varies between 5 and 25 Mbit/s.



Figure 16: Orah 4i Camera

The Orah 4i needs a single ethernet cable, which takes power in and streams videos out to the Stitching Box. The Orah 4i was designed for live immersive events, and has features such as an embedded Inertial Measurement Unit which allows to have automatic horizon levelling of the output spherical video (i.e. the horizon remains horizontal whatever the position of the camera is) and automatic stabilization (if the camera is not static). The camera is equipped with 4 microphones to capture a 3D ambisonics sound field, which will ease the process of recording omnidirectional audio for ImmersiaTV system. The main characteristics are presented in Table 10.

Sensors	4 x Sony EXMOR	Hardware synchronization between sensors Associated to a Stitching Box, delivers 4K/30 fps to an SD card or an RTMP streamed AVC output.
Video resolution per lens	1920 * 1440	Pixels
Lenses	Fisheye 4x f2.0	Horizontal field of view close to 180°
Camera exposure	Automatic exposure	
White balance	Automatic white balance	
Inertial Measurement Unit	Detecting the motion and orientation of the camera	horizon-levelling of the output video

		stabilization of the output video
Microphone	4x high dynamic range microphones	ambisonics sound field
Output	4x H.264 RTMP streams	Streamed to the Orah Stitching Box
Power	48V over PowerOverEthernet	
Dimensions & Weight	8 x 7 x 6.5 cm 0.5 kg	

Table 10: Specification of Orah Camera

5.2.2.2. Studio.One Cameras

AZilPix, a recently founded spin off of Hasselt University and iMinds (now IMEC), brings to the market a video capture system named Studio.One, that integrates high resolution block cameras and custom omnidirectional and panoramic capture rigs designed by iMinds-EDM. It aims at simultaneous live conventional + VR broadcasting.

The Studio.One system consists of cameras and a custom server equipped with the Studio.One software. The server is described further in this document. Here, we concentrate on the cameras.

For pilot 2, one or two full omnidirectional 360 video rigs will be used, depending on availability, as well as up to 8 block cameras equipped with a 170-degree field of view fish eye lens. We call these latter “semi-omnidirectional” cameras: they are used exactly in the same way as omnidirectional rigs, but do not offer a full spherical coverage.

In our experience, the 360 camera rigs make most sense providing overview to a spectator, making him/her feel as being central in the event, as well as conveying the ambiance of the event. Due to several factors, technical as well as non-technical, 360 video is not suited for conveying detail in our experience. The semi-omnidirectional cameras are well suited for conveying detail. They could be placed along important sections of a cyclocross track for instance, allowing to follow the action by cutting between them, for instance.

The 360 cameras are connected to a stage box, at up to 20 meters from the camera. The stage box is connected to the AZilPix Studio.One server, over optical fiber which can be hundreds of meters or more. The Semi-omnidirectional cameras are connected directly to a AZilPix Studio.One server unit, at up to 100m.

360 camera rigs:

- 6 full HD sony imx249 sensors
- 12-bit color resolution
- Up to 40 fps
- 2.7mm fujinon C-mount fish eye lens, offering 185 degrees field of view on each sensor

- Sensor shutters are synchronised to microseconds precision.
- Double redundant 360 spherical coverage allowing control of location of stitching seams in addition to stitching sharpness and distance. Enables advanced stereoscopic and 3D omnidirectional reconstruction algorithms (in research).
- Precalibrated
- Connected to a stage box via 6 gigabit Ethernet cat5e UTP network cables and one neutrik XLR 5-pin connector cable for synchronisation signals and power.
- Rigid black anodized aluminium housing.
- Mounts physically on a microphone tripod or manfrotto magic arm.

360 camera stage box:

- Located up to 20m from 360 camera rig.
- Connects camera via 6 cat5e UTP cables and one neutrik XLR 5-pin connector cable
- Connects to the AZilPix Studio. One server via an optical fiber cable, allowing placement at 300m with multimode optical fiber, or kilometres with single mode optical fiber.
- Optional SMPTE LTC time code signal input over balanced line level audio neutrik XLR 3-pin connector cable, for automatic synchronisation with broadcast video cameras and audio.

Semi-omnidirectional cameras:

- 12 megapixel sensor, 60 dB dynamic range, up to 180 fps
- Canon EF L 8-15 mm fish eye zoom lens, at 8 mm focal length
- Precalibrated
- Solid black anodized aluminium housing
- Mounts physically on 3/8" 16tpi microphone tripod or manfrotto magic arm, or 1/4" 20 tpi camera tripod (same as 360 rigs).
- Integrated Canon EF lens controller allows to set focus distance and aperture from AZilPix Studio. One software.
- Connected to AZilPix Studio. One server unit via a HD-SDI cable and a gigabit Ethernet cable of up to 100 meter:
 - Power, synchronisation signals, control and data over standard HD-SDI cable. Longer distances and higher frame rates with multiple links.
 - Lens control power and data over gigabit Ethernet cable (to Power Over Ethernet capable gigabit Ethernet switch or power injector).

All devices mentioned above are depicted on Figure 17 and 18.

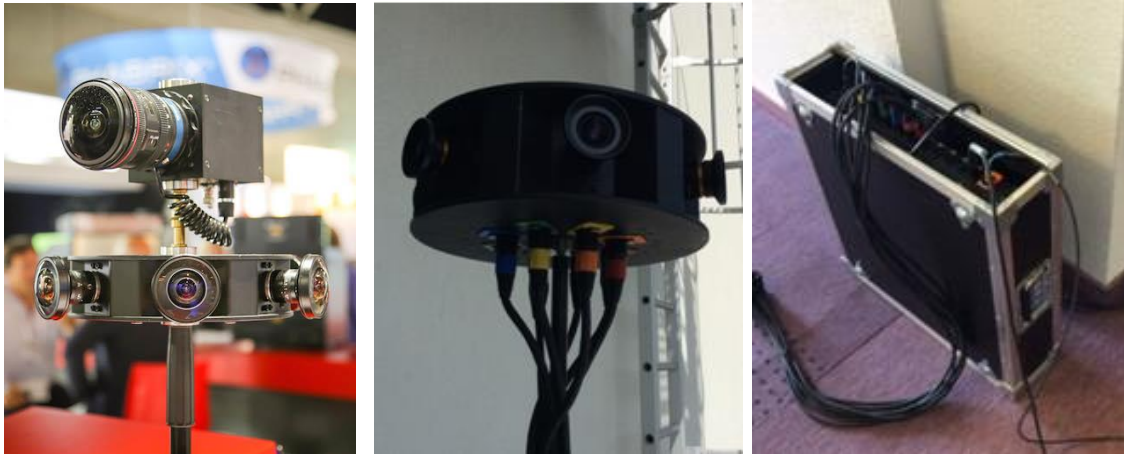


Figure 17: AZilPix Studio. One 360 camera rig (bottom left and middle) and semi-omnidirectional camera with canon fish eye lens (top left). Stage box on the right.



Figure 18: AZilPix 360 camera centre front Dranouter music festival main stage (ICoSOLE EU project, www.icosole.eu)

As shown on Figure 19, semi-omnidirectional video is sufficient for full 360 experience in this and many other cases: there's little to be seen behind, above or below the camera. Semi-omnidirectional cameras allow to concentrate pixel budget on where the action is, and obtaining a better balance between number of points of view and field of view per point. Full 360 cameras of course are required when the action covers more than 170 degrees.



Figure 19: Example of semi-omnidirectional video.

5.2.3. Live stitching system

Depending on the used cameras, several live stitching systems can be used for pilot 2:

- Vahana VR, for GoPro or Elmo rigs. Vahana VR can adapt to cameras available on the market, and can be used for stitching preview and adjustments. It may also be the preferred solution if we need to go to resolutions and frame rates larger than 4K 30 fps;
- Orah Stitching Box, to be used in conjunction with the Orah 4i camera;
- Studio.One distributed stitching system, to be used in conjunction with Studio.One camera.

5.2.3.1. Vahana VR

Vahana VR is VideoStitch's software application for live stitching, running on a computer with dedicated input and output ports. The main view is presented on Figure 20 and Table 11 contains parameters.



Figure 20: Vahana VR software screenshot

Video signal capture options	<p>Hardware adapters:</p> <ul style="list-style-type: none"> ● Magewell HDMI capture cards ● BlackMagic Deckling SDI cards <p>Network adapters:</p> <ul style="list-style-type: none"> ● RTP video streams 	Full list available on http://support.video-stitch.com/hc/en-us
Video output options	<p>Hardware adapters:</p> <ul style="list-style-type: none"> ● BlackMagic Deckling SDI cards <p>Network adapters:</p> <ul style="list-style-type: none"> ● RTMP AVC streams 	Full list available on http://support.video-stitch.com/hc/en-us
Computer requirements	<p>Windows 7 or later, 64 bits</p> <p>nVideo GeForce graphics card 900 series or better, at least 3 GB of graphics memory</p>	<p>Linux not supported officially, but Ubuntu 12.04.4 64 bits should work.</p> <p>No Mac version.</p>
Preview	Live preview on computer screen, or through Oculus Rift head mounted display	
Max output resolution	Depends on the graphics card	
Supported input projections	Rectilinear lenses, Fisheye lenses, Equirectangular inputs	

Supported output projections	Equirectangular	
-------------------------------------	-----------------	--

Table 11: Specification of Vahana VR software

Vahana VR’s workflow for live is the following one:

- read the input sources coming from an omnidirectional rig
- calibrate the camera rig geometry (through self-calibration, or an offline calibration template)
- calibrate the camera rig photometry (to make up for various exposures, colour temperatures and lens vignetting)
- map each camera view onto a 360° equirectangular frame
- adjust the frame orientation for horizon levelling
- output the equirectangular video, through an RTMP stream or the computer physical HDMI/SDI interfaces

For Pilot 2, other types of output projections will be supported, such as cubic mappings, as recommended by EPFL and the codec WP (see section 5.4.5).

Also for Pilot 2, common rig templates will be added, to have an easier calibration workflow, and first HEVC coding tests will be performed.

Since Vahana VR is capable to ingest an already stitched input such as Studio.One’s, it will be possible to use it in a stand-alone mode as a live projection remapper and encoder, before or after the live production tools in the pipeline, until the live HEVC encoder of 5.4 is ready.

5.2.3.2. Orah Stitching Box

The Orah Stitching Box (depicted on Figure 21) is the live processing unit connected to the Orah 4i camera through a routed network. Its main characteristics are in Table 12.



Figure 21: Orah Stitching Box

Live output resolution	4096 * 2048 pixels, 30 fps	
Output Projection	Equirectangular	

Output Field of View	360° full spherical	
Preview	Through a web app, accesible on the box Wifi network	
Encoded Video Format	AVC High, Main or Baseline profile	
Output video bitrate	From 5 to 25 Mbps	
Output protocol	RTMP streams File format to save the inputs/outputs to disk	

Table 12: Specification of Orah Stitching Box

As for Vahana VR, other types of output projections will be supported in Pilot 3, as recommended by EPFL and the Encoder component. The box has an HDMI output which will be activated by a software update.

5.2.4. Studio.One stitching

The Studio.One cameras are eventually connected to one or more Studio.One servers running software to control the cameras and process their raw video streams, including stitching.

The software implements a full videography pipeline, running at 4 billion pixels per second. The pipeline encompasses low level sensor related image corrections, state of the art denoising, an advanced Bayer demosaicking, color space conversion to sRGB or Rec709 color spaces for computer monitor and HDTV respectively, correction of lens vignetting and geometric distortions, warping into rectilinear, equirectangular or other projection views, with blending for panoramic and 360 video stitching, video image enhancements using advanced edge preserving filtering, unsharp masking, and several calibration and monitoring widgets including histograms, waveform monitors, digital vectorscope, overexposure indicator, and focus peaker.

The software implements camera, recording and playback control, as well as an elementary interface for creating conventional video or VR video editing. It also supports live editing, via presets and transitions, and view switching. A low level API is available, as well as a RESTful web API allowing control over the network. Live remote production was tested from IBC 2016 in Amsterdam, on the live video screens at the Leffinge-Leuren music festival near Ostend in Belgium, that took place at the same time. It was one of the main results of the ICoSOLE EU FP7 project. The interface is the key to building a distributed capture and production infrastructure.

The capture and processing server performs real-time stitching. With a HDMI 1.4 output, UHD stitched content is produced at max. 30 fps. This is a limitation of the video output on the GPU or the HD-SDI conversion or display connected. The stitching process is fast enough to allow 4K stitched output at over 100 fps from the here described cameras.

Transmission is realized via an encoder or conversion box attached to the HDMI, DisplayPort or DVI computer video output. Multiple simultaneous outputs are available, for instance one rectilinear conventional full HD output stream along a equirectangular UHD panoramic output.

The server can take a number of our semi-omnidirectional 170 degrees cameras along with a 360 rig, and produce multiple equirectangular outputs as well.

The server software supports GPU video encoding with nvidia CUDA (NVENC API). We use it typically for generating stitched video files. We will test live video encoding, and will offer it in the trial if tests are successful. Main risk concerns the fact that it introduces a synchronisation point in our otherwise highly asynchronous multi-threaded software.

The base plan is to convert at least one equirectangular output to 6G-SDI with a blackmagic HDMI mini convertor and feeding into a Vahana VR server with blackmagic declink studio 4K capture card. Vahana VR provides further processing and AVC video streaming.

Camera to screen latency is in the order of 2 to 3 frames, half of which is probably due to the screen tested (dell ultrasharp full HD display).

Physically, the server is mounted in a 4 units or 6 units 19 inch rack mount flight case, together with time code generator, audio recorder and/or power over Ethernet switch for lens control. The flight cases have a footprint of 60x60 cm and are 25 respectively 40 cm high. They require 220V power and consume typically around 500 Watts, including monitor screens if set up like that. We will remote control the servers at pilot 2, via a 100 megabit Ethernet connection and using the RESTful web API. Ethernet and SDI video stream for live editing and monitoring can be brought to the OBVan over fiber using a neviion breakout box.

The Studio.One server is depicted on Figure 22 and screenshots of software are on Figure 23 and Figure 24.



Figure 22: Studio.One server in rack mount case with time code generator and audio interface and robust connectors to stage box, cameras and control room.



Figure 23: Screen shot of the interface of the Studio.One software. Concerns a medical surgery procedure video capture for training purposes. The session integrated 2 HD-SDI sources, a 360 rig and 4 semi-omnidirectional rigs - all real time on one server. (with dive – eSurgie)



Figure 24: Equirectangular video produced simultaneously with previous. (In fact, a semi-omnidirectional camera would have been better for this point of view, as the action is concentrated in less than 170 degrees field of view.)

5.3. Live production Tools

5.3.1. Description

Live production tools design and architecture are based on the experience obtained from the Pilot 1 execution and requirements for the live workflows gathered during evaluation sessions. Taking into account the existing complexity and challenges of traditional live TV production workflow the aim is to provide the Live Production tools operator with functionality to produce spectacular immersive experience and at the same time avoid additional work load.

Live production tools package consists of the following software:

- Cinegy Live VR
- Cinegy Transport

Cinegy Live VR provides the operator with the user interface for live VR production while Cinegy Transport takes the complexity of all required media transformations to connect all sources and targets in the background. Cinegy Live VR software screenshots are presented on Figure 25 and Figure 26 below.

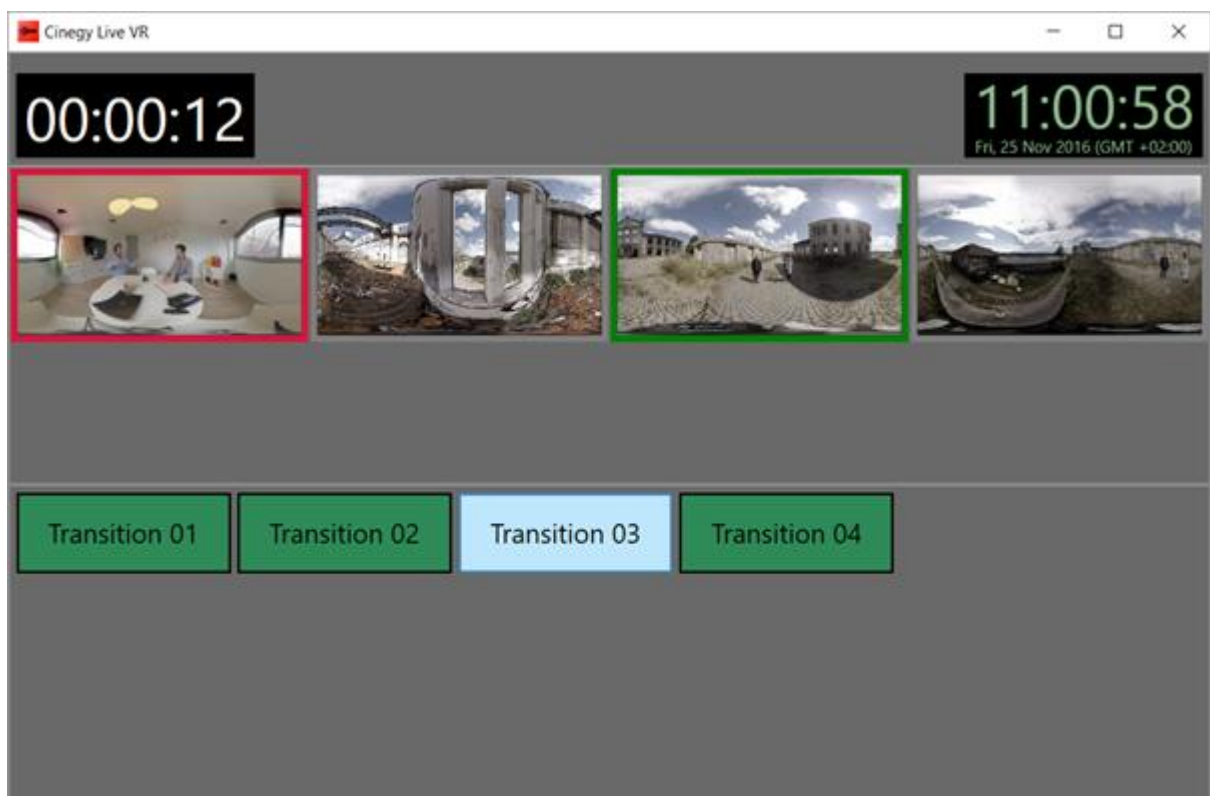


Figure 25: Sample user interface for Cinegy Live VR

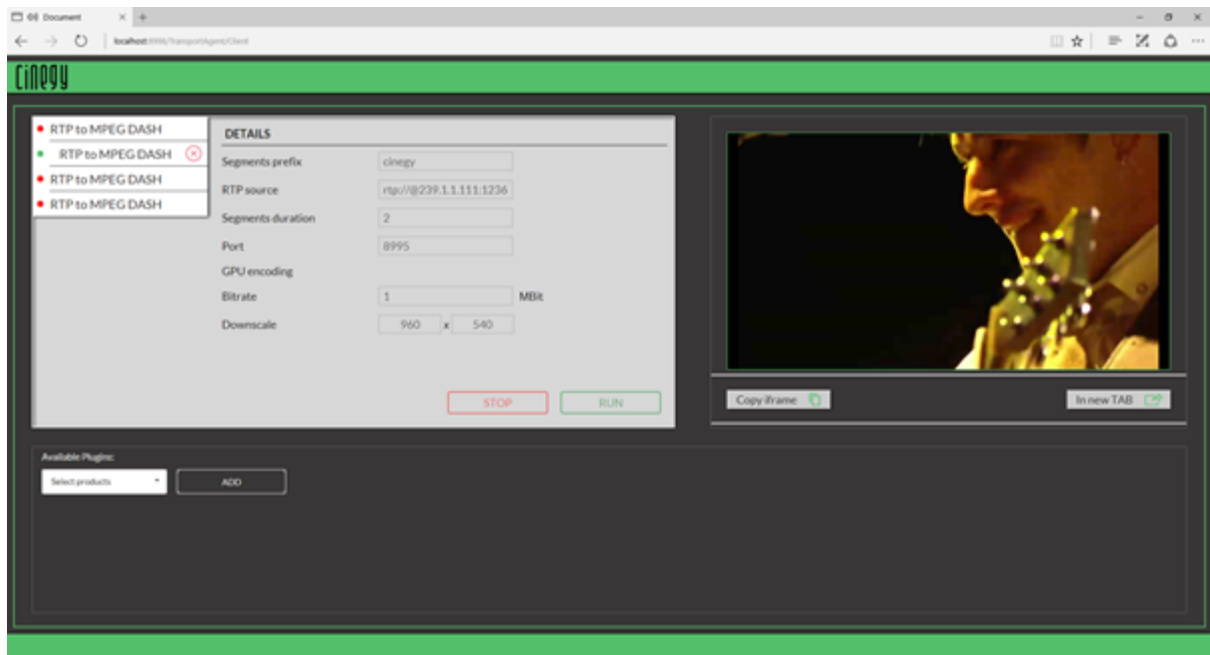


Figure 26: Sample WEB configuration interface for Cinegy Transport

5.3.2. Architecture

The following architecture will be used during the Pilot 2 execution (Figure 27):

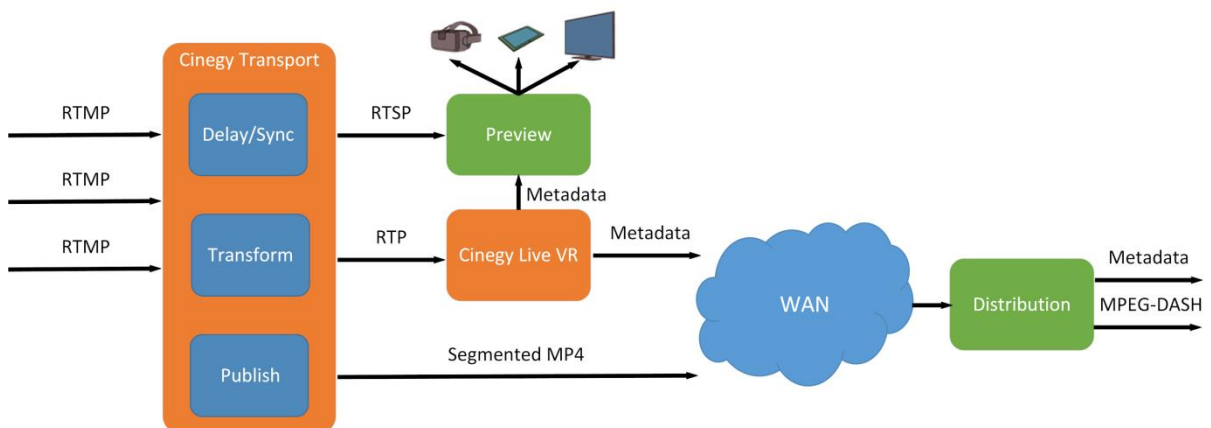


Figure 27: Architecture of Live Production Tools

All sources to be used in the Live Production are registered with Cinegy Transport server(s) to be explicitly synced both on metadata and physical level. The synced streams will be also optionally transformed into different versions (re-packaged) to be accepted by the corresponding target. For example, incoming RTMP stream from Oras 4i/Vahana VR will be synced with other streams in production (delayed) and emitted as RTP preview stream for Cinegy Live VR, emitted as RTSP preview stream for low latency operator preview, packaged as segmented MP4 for Delivery Module (for example, Unified Streaming Platform) or immediately published as MPEG-S-DASH package for custom delivery.

The Cinegy Live VR software provides the operator with the required tools to create exciting immersive experience by providing real-time preview of available source streams, ability to define the active source, dynamically switch between sources, define optional portal size and

position. All operator actions are timestamped and stored inside the live production metadata file. The updates of the metadata files expressed as events are to be displayed by ImmersiaTV players.

The primary goal of Pilot 2 for live production tools is to ensure synchronous delivery of all media coming from different sources and locations (directional cameras, omnidirectional cameras of different models, etc.).

The architecture will be extended and optimized during Pilot 3 to allow user interaction within live production (portals activation, choice between directors' choice and user driven experience, etc).

5.3.3. Workflow

5.3.3.1. Synchronization

As source streams are coming from different platforms (camera models, stitching modules, encoding modules) the delays or time desync between them will be notable. In order to remove desync and provide the same time base Cinegy Transport Delay module will be used. This module accepts incoming RTMP streams, patches stream metadata to insert the new common timecode, delays the transmission of the output to the defined number of seconds (milliseconds) and emits the modified re-packaged result as RTP stream.

This allows defining custom delays for each of the source stream in production to make them in sync after Cinegy Transport Delay stage is passed. As virtually no computations are done at this stage the only remarkable requirement is amount of available RAM to buffer all incoming stream. The amount of RAM depends on the number of streams and the largest required delay.

5.3.3.2. Live Production

Using Cinegy Live the interface operator is able initiating the scene changes by adding additional objects (for example, portals), changing the primary video source, displaying additional graphical objects (highlights), etc. All changes are stored as events with the defined format in the metadata XML file that is being updated and published via the Distribution module for the external viewers to consume.

At the same time the operator is able to preview the changes to be made using local low latency ImmersiaTV player and compatible device.

5.3.3.3. Preview

The low latency preview is achieved by consuming the local RTSP version of the stream produced by Cinegy Transport. The alternate version of the scene is available for the operator to try the changes before they are published for the external audience. Once operator is fine with the changes/transitions/etc. he is able to send the update to the public.

5.3.3.4. Publication

Cinegy Transport is responsible for producing the final version of the streams to be published on the external access point. MPEG-DASH or segmented MP4 version of the stream is created and made available to the Distribution module that can be located either locally or remotely (for example, on an AWS machine in the cloud).

5.4. Encoding

5.4.1. Overview

This section specifies the encoding process required to provide 360 video content within a live production workflow as set forth for Pilot 2 in ImmersiaTV Part B Technical Annex.

In contrast to ImmersiaTV Pilot 1 where the encoding step was performed off-line and therefore could be implemented as a stand-alone component of the production flow, Pilot 2 requires the encoding process to be performed in-line with the video output from the live production tools (see Figure 28).

Real-time encoding necessitates a series of trade-offs which ultimately affect the quality of experience delivered to the consumer. In the following we outline the proposed architecture to be adopted by ImmersiaTV for the Pilot 2, explore the particularities of video encoding for ImmersiaTV, discuss the key factors directly effecting encoded media quality, review the necessary compromises which need to be made to achieve real-time processing and explore the impact which above mentioned compromises will have on the service provided.

In view of codec choice (see discussions below) and for Pilot 2, AVC shall be adopted as the primary codec solution. This assures a realizable solution in compliance with the overall Phase 2 project requirements. In parallel and in view of Phase 3 project goals, support for HEVC shall be explored and implemented.

5.4.2. Requirements

5.4.2.1. AVC based solution

For an AVC based solution, the encoder will be fully integrated within the Vahana VR platform and Orah cameras. Upstream external requirements are those specified within the Vahana VR system.

External requirements are:

- Compatibility with H.264 High, Main or Baseline profile
- Frame size: $\leq 4K$
- Frame rate: ≤ 30 fps
- Chroma: 4:2:0 at 8 bit
- Projection: Equirectangular
- Output: RTMP stream at 5 ~ 25 Mbps

5.4.2.2. HEVC based solution

An HEVC based solution will be realized on a stand-alone platform which is co-located next to the Vahana VR system. Upstream external requirements are:

- Interface: uncompressed video frames over HDMI or SDI
- Frame size: $\leq 4K$
- Frame rate: ≤ 30 fps
- Chroma: 4:2:0 at 8 bit
- Projection: Cubic 2x3 rotated
- Signalling of input frame stream characteristics - once per live stream at initialization

Downstream external requirements are:

- Projection: native support of Cubic 2x3 rotated projection in player or integration of a re-mapping function from Cubic 2x3 rotated to supported projection at the reception system (post decoding)
- Frame size: determined by capabilities of reception system (max: same as input)
- Frame rate: ditto
- Chroma: ditto
- Output: RTMP stream (max 25 Mbps)
- Signalling of output encoded video stream configuration (frame rate, size etc) - once per live stream at initialization

Design and configuration requirements:

- Real-time hardware (GPU) supporting encoding of single video stream
- Encoder input: YUV frames at 8 bit color depth
- Operating system: Ubuntu Linux LTS 16.4 (64 bit)
- System: High end workstation (or rack mounted server).
- High speed (SSD) disk.
- GPU: NVIDIA GeForce GTX 1080 (“Pascal” architecture) or similar
- Framework: FFmpeg 2.6 or later (GNU license)
- Interface - input: HDMI or SDI card allowing for real-time frame acquisition and buffering. Requires SDK supporting a real-time API
- Interface - output: RTMP over ethernet
- Encoder configuration: Per encoding session - configuration based on up and downstream session information and based on pre-configured encoder configurations.

5.4.3. Real-time encoding architecture

The following block diagram illustrates the proposed encoding system for Pilot 2 and serves to illustrate both the required trade-offs and key design decisions:

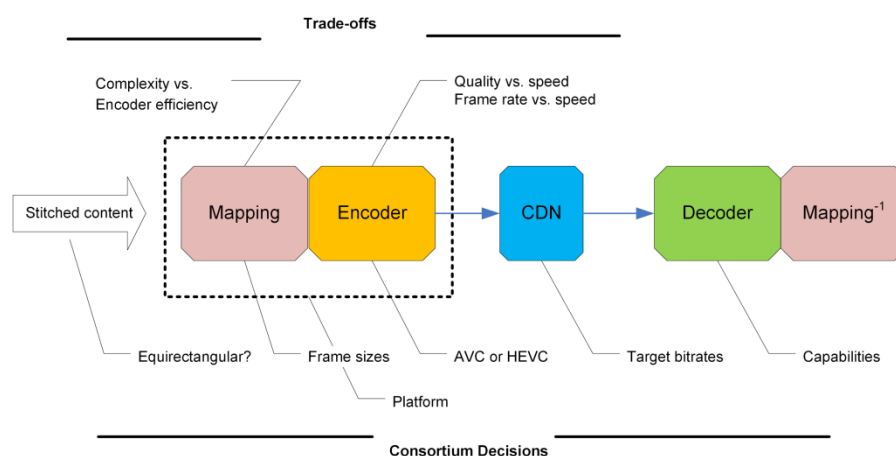


Figure 28: Real-time video codec within the ImmersiaTV work flow

There are different process steps moving from left to right in the above figure:

Both the Videostitch Vahana VR and EDM/AZiPix Studio. One realtime stitching system provides stitched content at up to 4K resolution and 30 fps over an HDMI/SDI output in uncompressed format (i.e. on a frame-by-frame basis). This sets the benchmark for maximum encoder performance in a real time setting.

Both Vahana VR and Studio. One stitch input frames onto an equirectangular projection. This projection is sub-optimal for encoding purposes as it contains a high degree of redundant pixel data (see section 5.4.5 below). Therefore, the output from Vahana VR can be re-mapped onto a projection which enables the encoder to perform in a more optimal fashion. This is of particular importance in a real time setting where resource wastage (in terms of frame sizes) should be avoided.

For Pilot 2 the re-mapping process will be integrated into the stitching system (i.e. inside the machine hosting the Vahana VR software or the Studio. One server).

The particular re-mapping applied to the video frames prior to encoding needs to be considered at the rendering stage of the playback process. Different devices require different mappings. As such, a re-mapping to projections supported by display devices may be required. As detailed in Section 5.4.5 this “re-re-mapping” will lead to a loss of quality and will incur additional computational load at the rendering stage. As such it may be advantageous to perform the pre-encoding re-mapping to the projection which is natively supported by the display device while making compromises in terms of best encoder efficiency and quality.

The stitched and appropriately re-mapped video frames are passed to the real time encoder which is to be realized on a dedicated computing platform. This platform needs to support the output generated by Vahana VR and therefore be equipped with an HDMI/SDI input card with suitable drivers and an API supporting programmatic control, real time management and pipelining of the input video frames.

Downstream play-out in ImmersiaTV restricts the options of possible encoders to either H.264/AVC or H.265/HEVC. The trade-offs resulting from an encoder choice in terms of quality, bandwidth, input frame formats and rates, chroma subsampling, bit depth, computational complexity and host platform requirements are summarized in Section 5.4.5 below.

In terms of a forward looking platform design, H.265/HEVC is the natural encoder choice. Yet this choice needs to be matched to the capabilities of the decoder on the terminal which plays out the received video bitstream. At this point in time these capabilities are not always guaranteed and there are open questions in terms of supported codecs, target frame size, rate, bit depth, color coding and projections which can be rendered.

In view of a conservative solution for Pilot 2, support for H.264/AVC will be extended beyond the Pilot 1 realization to support real time encoding. Given that the Vahana VR system supports native output of H.264/AVC over RTMP, the development work can be limited to configuring an H.264/AVC transcoder to operate in real time and output the frame sizes and rates which can be successfully decoded on the client end. This approach ignores the potential computational and bandwidth reductions that can be realized through re-mapping of the video frames generated by Vahana VR. Yet, introducing this re-mapping into the workflow would require an initial decoding step (of the H.264/AVC content generated by Vahana VR), followed by a re-mapping followed by a re-encoding to H.264/AVC. In addition (see above) the received and decoded H.264/AVC bitstream may require an additional re-mapping to conform to the requirements of the display device.

In view of assuring substantially better visual quality, H.265/HEVC should be adopted as the encoding standard. Here the computational requirements at both the encoder and decoder are substantially higher. But on the encoder side, cost effective solutions are now available.

Furthermore, and in view of Pilot 3 where encoder control and optimization is foreseen through QoE parameters supplied in real time by the play-out system, H.265/HEVC offers a greater scope for dynamic encoder control and adaptation.

5.4.4. Comparison of H.264 vs. H.265

For a preset target visual quality, H.265/HEVC offers compression which is between 35% and 50% better than H.264/AVC. The improved compression offered by H.265/HEVC is offset by a substantially higher complexity of the encoding process. Likewise, decoding of H.265/HEVC encoded content is more complex than that for H.264/AVC. This poses challenges, particularly for battery powered devices where it only makes sense to offer H.265/HEVC decoding with embedded hardware support.

The following table lists the real time HEVC encoder solutions currently on the market:

	Encoder	Real-time	Implementation	Open Source	License	Notes
1	Intel MSS HEVC Encoder	4k p60 8-bit	SW	No	Commercial	Intel® Xeon® and 6th generation Intel® Core™
2	x265	4k p60 10-bit	SW	Yes	GPL 2 and commercial license	Dual Intel Xeon E5 v3 server
3	Fraunhofer HHI HEVC SW Encoder	4k p60 10-bit	SW	No	Commercial	Available as SDK (multiple platforms)
4	Vanguard Video V.265 Encoder	4k p60 8-bit	SW	No	Commercial	Runs on 36 cores
5	Nvidia NVENC	4K 8-bit	HW (+ SDK)	No	Commercial	Maxwell (GM206), Pascal;

Table 13: List of HEVC implementations on the market

In view of Pilot 2, solution number based on NVidia NVENC (number 5 in Table 13) offers the most efficient route to real-time encoding but offers limited flexibility with regards to encoder optimization (as this is implemented in hardware within the NVENC GPU). In view of Pilot 3 where real-time optimization on the basis of QoE measurements will be implemented, x265 solution (number 2 in Table 13) is the only avenue. Here it should be noted that the x265 code base (as available under GPL 2) does not achieve real-time performance. This can only be reached after substantial re-factoring of the code base for high computational efficiency.

5.4.5. Re-mapping

Omnidirectional images and video must be represented in a panoramic mapping in the form of a rectangular picture/frame in order to be processed by an encoder (H.264/AVC, H.265/HEVC, etc.) Equirectangular projection today is the most widely used representation supported by all capture devices and stitching tools. However, this representation contains redundant pixels,

especially in the polar regions. To optimize the geometrical representation and before processing a frame by an encoder, re-mapping should be performed.

There exists a range of projections which utilize different geometric transformations in order to reduce the number of pixels per frame, notably variations of cubic projection (cubic 3x4, cubic 3x2, cubic 2x3 rotated), truncated square pyramid projection, and more complex icosahedron and dodecahedron projections. Some of these projections preserve the visual quality of the rendered viewports (e.g. cubic) while others decrease the quality of select viewports via downscaling of the corresponding regions in the spherical image (e.g. truncated square pyramid). The following Figure 29, schematically outlines the re-mapping for an input YUV image and indicates the respective reductions in the resulting number of pixels:

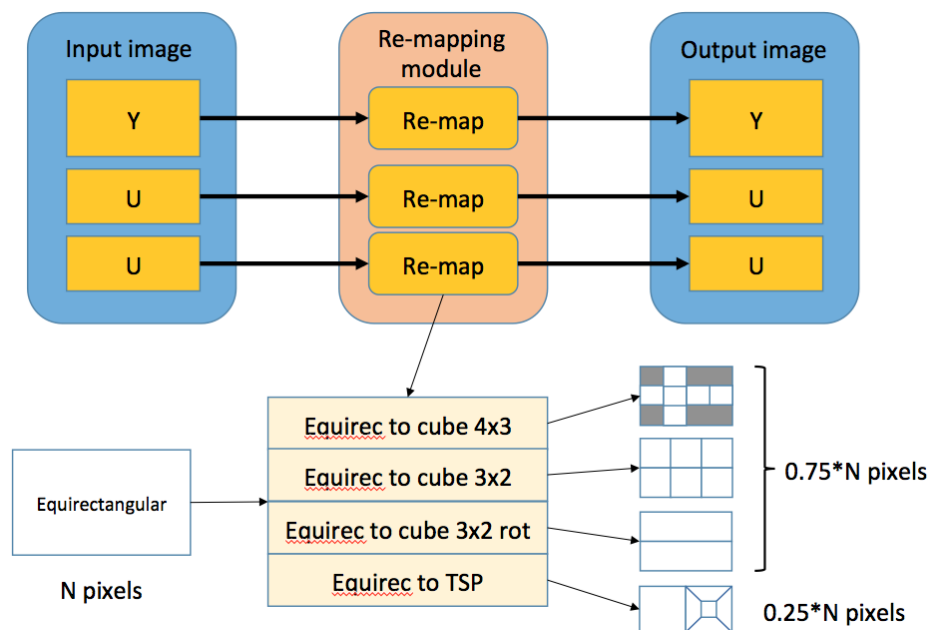


Figure 29: The re-mapping process for an input YUV image

During transformation from one projection to another, not every pixel of the target image has a correspondent pixel in the source image. Figure 30 illustrates pixel loss during re-mapping from equirectangular to cubic projections. Pixels which do not exist in the source equirectangular picture are marked with red color:

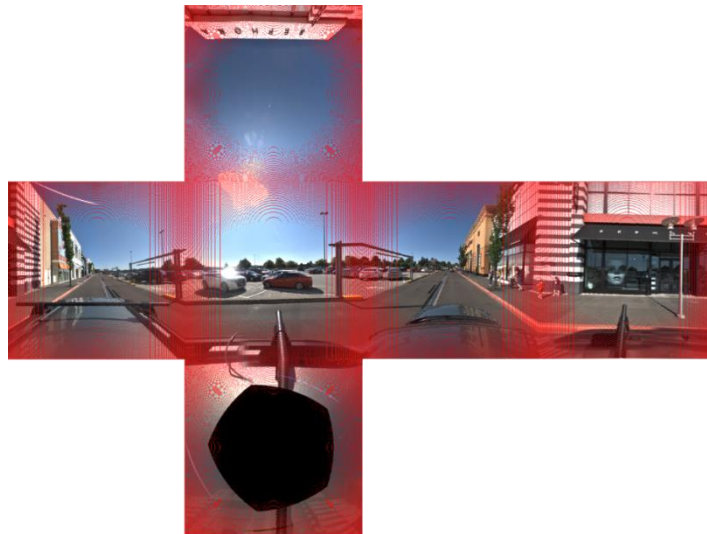


Figure 30: Pixel loss during remapping from equirectangular to cubic projection

In order to fill these gaps, an interpolation or a re-sampling of the source image must be performed. Another purpose of re-sampling is to prevent aliasing. It appears in details where a Nyquist frequency is higher than the sampling frequency of the signal. The most common re-sampling algorithms are nearest neighbor, bi-linear, bi-cubic, and Lanczos interpolations. Re-sampling represents the computationally most expensive part of the remapping process and a bi-linear strategy will be used for Pilots 2 and 3.

In the context of remapping for encoder optimization we must choose an optimal projection. The following main criteria should be considered: representation efficiency (pixel redundancy), visual quality of rendered image, and re-mapping complexity. Cubic projection 3x2 with rotated faces is the recommended choice as it preserves virtually the same quality as equirectangular picture, reduces pixel redundancy and the computational complexity is low. Moreover, rotating particular cube faces reduces spatial complexity of the picture which leads to higher compression efficiencies.

5.5. Distribution

Architecture and workflow of Distribution component follows the approach from Pilot 1 with the exception of the WAN content publication. Cinegy Transport and Cinegy Live will publish content to the AWS instance running Universal Streaming Platform (USP) software. MPEG-DASH versions of the streams and remote viewers connections are to be handled by USP. Such approach ensures the optimal bandwidth requirements for publishing content from the Production origin, while distribution can use much wider bandwidth available for AWS instances and properly scale to the required number of clients.

It is expected that the Distribution architecture will be revised in several iterations during the Pilot 2 preparation and test runs on dedicated events.

5.6. Reception, Interaction and Display

Architecture and workflow of Reception, Interaction and Display component remains the same as in Section 4.6 (Pilot 1).

5.7. Quality of Experience

5.7.1. Description

Quality of Experience (QoE) represents the degree of satisfaction or annoyance of the immersive visualization at the end-user's side. The QoE module delivers quality scores for omnidirectional video contents which is well-correlated to human user opinion. According to the obtained quality score, the codec parameters can be adjusted to improve quality.

5.7.2. Software Architecture

The QoE module is located at the server side and is responsible for quality assessment of the videos. Figure 31 shows the overall diagram of the QoE module implementation. The QoE metric is a model to estimate the quality based on the distortion, content features, bitrate, etc.

As shown in the Figure, the QoE module receives information from client and server sides for quality assessment. The *QoE metric* module has three inputs: (a) content features, (b) distortion features and (c) parameters sent by client (delay, etc.) through a side channel.

The **content features** can be directly extracted from reference videos. The content features describe the spatial complexity of the input video.

The **distortion features** are the parameters that represent the quality degradation. These parameters include quality-sensitive features extracted from the decoded (distorted) frames. The distortion features can be obtained either from a decoder simulator located in the *External server*, or directly from the client side.

The **client information** (such as delay and Movement acceleration of HMD, Display device, etc) are provided via a side channel.

The gathered information are fed into the QoE metric module (in a xml file). Based on this information, the QoE metric delivers a stream of quality scores.

It is important to consider the effect of delay on the user experience. The delay problem has not been observed in Phase 1 but it may become an issue in Phase 2 due to online streaming. In case of observed delay (freezing/stalling), the raw information of the delay (start and duration of delay) will be considered in the model. The raw information will be sent to the server through a side channel.

The scope of Pilot 2 is to develop an objective QoE module to evaluate the quality of videos. In pilot 3, a QoE module with a feedback loop will be provided for the ImmersiaTV platform that will steer the parameters of the video encoding in real time.

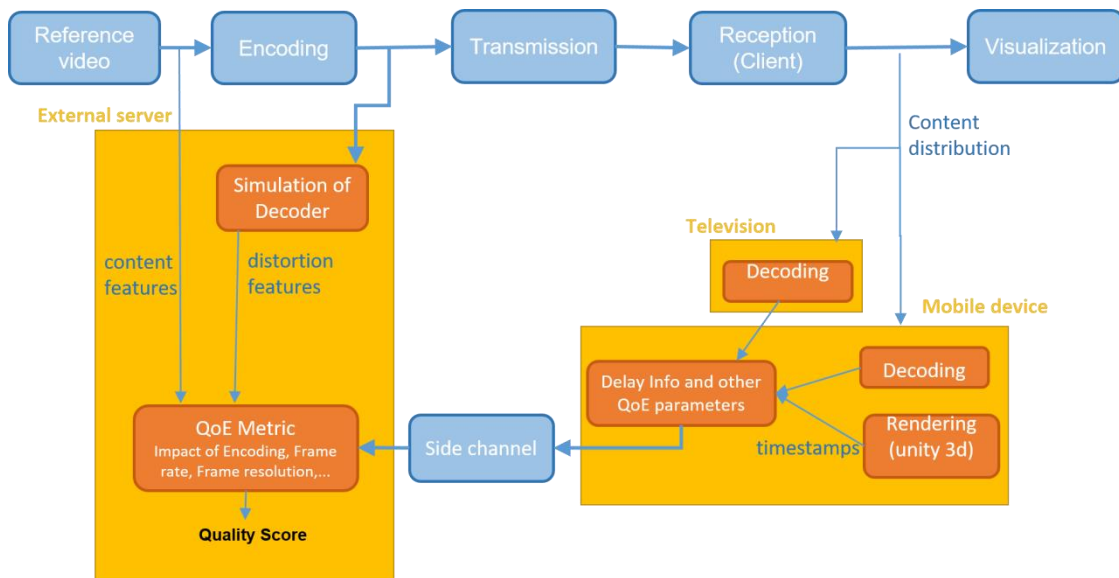


Figure 31: Architecture of QoE component

5.7.2.1. List of QoE parameters

The QoE module will require a number of information as input to assess quality. Table contains a list of required information for QoE module.

Parameter	Type and description	Units	Frequency	Where is measured
Bit-rate	Dynamic Information	(int) kbps	Per second	Client
Frame-rate (fps)	Dynamic Information - content frame-rate	(int) fps	Per second	Client
Frame resolution	Dynamic Information - content resolution - Width and Height	(int) Width pixels int Height pixels	Per frame	Dash Server or Client
Field of view	Static Parameter of the device	(int) degrees in horizontal direction	Per stream	Client
Display device (Tablet, HMD, etc)	Dynamic Information	-	Per change-occurrence	Client
Movement acceleration of HMD	Dynamic Information - Position and time	(float) Horizontal and Vertical degrees of the center of FOV	per frame	Client

Video Frames (reference and distorted ones)	Dynamic Information - The distorted frame can be obtained in the server side using a simulation of decoder (Decoder emulator)	-	per frame	Server
Delay Information	Dynamic Information - start time and end time of stalling/delay	(float) seconds since start of the session	per occurrence of each buffer underrun start and end	Client

Table 14: List of statistics required for QoE assessment

5.7.2.2. QoE Library

The QoE library will require both static and dynamic logging information as input. The static information will be sent once when setting up the QoE module. The dynamic information needs to be continuously recorded at specific time intervals.

The logging information will be provided in XML format. The format should be backwards compatible if new parameters are added, also it should allow recording only a subset of full parameters' list. The library will be written in C/C++ in the server.

The QoE library uses the parameters of logging to estimate quality. The output is a stream of quality scores whose values are between 0-100. The higher value shows the higher quality.

5.7.3. Workflow

For pilot 2, the QoE module will be devised and tuned for H.264 codec. In pilot 3, the QoE can be extended and/or redesigned if new quality degradation sources are introduced (e.g. quality degradation in audio, capturing, new encoding type - HEVC, etc).

The QoE module can provide quality scores (from 0 to 100 for instance) to represent the quality degree. It could be a stream of scores. In order to consider the effect of motion characteristics, we report the quality score after analysing several frames instead of providing a score for each frame. The QoE module extracts the required parameters from the XML file and feed the parameters into the QoE model to estimate the quality.

As an initial prototype, we will only focus on the main distortion factors (Bitrate or delay for instance) to estimate the quality. The main goal at this stage is to ensure that the QoE can properly receive information and perform an objective quality assessment. Afterwards, the QoE module can be further improved by using more sophisticated models.

In order to validate the QoE module, the performance of the QoE will be examined on test videos with different distortion levels (videos generated with different codec parameters).

5.7.3.1. Feedback to the encoder

In Phase 3, the QoE module design will include a feedback loop to communicate with the Encoder. A number of advice codes will be provided for Encoder (based on the quality score) to steer the parameter settings for better quality.

6. CONCLUSIONS

This document presents the architecture design of the ImmersiaTV system being a base for all components and tools. The document was created iteratively, what means that the system architecture and components in section 4 which were focused on Pilot 1 implementation were extended and described more precisely in section 5 defining Pilot 2.

Regarding Pilot 1, although the market of the omnidirectional cameras is very dynamic and new models are presented by various vendors, most of the development still focuses on low-resolution home devices for amateurs. The only way we can foresee to achieve the goals of ImmersiaTV (high resolution high frame rate live camera) is to design and construct own cameras that meet all the requirements. This was presented in Section 5, which reflects these concerns and describes Orah 4i compact live camera and EDM omnidirectional camera rig.

Regarding codec, the AVC was selected for both Pilot 1 and Pilot 2. Adoption of HEVC was delayed due to limited availability of end user devices, such as smartphones or tablets. We believe, the use of HEVC encoding and decoding will be possible for Pilot 3.

For Pilot 1 and 2, the most important thing is to define the list of statistics will be gathered from all components and how to analyse them by QoE tools. Logger module provided at the end of Pilot 1 will enable to collect all necessary data, while advanced statistics gathering for real-time QoE assessment will be delivered for Pilot 3. What is also important, there is a difference in logging model described in previous version of this document (focused on Pilot 1). The reason for that was simplification the logging process in C# development of the client and using standard mechanisms of the C# and Unity 3D environment.

Distribution server and reception client are mostly the same across Pilots 1 and 2, due to the fact, the Immersia TV partners kept in mind live scenarios at the beginning of the project.