Analysis of adaptive streaming technologies for segmented multimedia transport of high-resolution videos for 360° playback on multiple devices

Master-Thesis

Benedikt Meyer-Schwickerath
[redacted]

Institut für Rundfunktechnik München (IRT)
University of Applied Sciences Hamburg (HAW)
Faculty of Design, Media and Information
Department Media Technology - Master Sound/Vision

[redacted]

First examiner: Prof. Dr. Robert Mores - HAW
Second examiner: Sebastian Siepe - IRT

Hamburg, June 2019

Abstract

Streaming of conventional planar video content is widespread in modern everyday life. Recently, dispersion of 360° video and virtual reality content, has increased with the upcoming development of transmission technologies and playback devices in the past decade and is trending to increase enormously in the next years. This evokes the need for even more efficient streaming and bitrate reduction technologies, considering today's available technical resources. This work gives an overview on current 360° video coding techniques and its peculiarities for video streaming. The principles of tiling at encoding-level, segmented transmission, and other concepts for streaming of 360° video, targeting bitrate reduction, will be discussed. The advantages of the new MPEG Omnidirectional Media Format (OMAF) standardization [51] will be outlined, with regard to other non-standardized streaming approaches. On the basis of these theoretical findings and according to the technical infrastructure and demands of the *Institut für Rundfunktechnik*[1] (IRT) and the public broadcasters, a use case scenario is conceived. Considering this use case, different 360° streaming systems, some of which OMAF-based, will be implemented and tested. Further, future trends and chances of 360° video streaming will be examined.

[1] En: Institute of Broadcasting Technologies IRT, Munich

Contents

Acknowledgments

First of all, I would like to thank my tutors and advisers Martin Schmalohr and Sebastian Siepe of the IRT, and my mentor and professor Dr. Robert Mores for their support, constructive feedback and confidence in my work.

Moreover, my thanks go to the IRT for providing the resources and the working environment for realization of this work.

Further, I would like to thank the developers of GPAC Telecom ParisTech, Nokia Technologies and Fraunhofer HHI, for providing their codes and Tiledmedia for their cooperation.

Acronyms

ADC	Asset Delivery Characteristics
ADST	Asymmetric Discrete Sine Transform
AOM	Alliance Of Open Media
AV1	AOMedia Video Codec 1
AVC	Advanced Video Coding, MPEG-standard
AVDE	AVC viewport-dependent baseline profile
BD-rate	Bjontegaard Delta Bitrate Savings
BLP	Baseline Profile
BL	Base Layer
BM-SC	Broadcast Multicast Service Center
CABAC	Context-adaptive Binary Arithmetic Coding
CAVLC	Context-adaptive Variable Length Coding
CB	Coding Block
CDF	Cumulative-distribution Functions
CDN	Content Delivery Network
CI	Composition Information
CMAF	Common Media Application Format
CNN	Convolutional Neural Network
CQ	Constant Quality
CTB	Coding Tree Blocks
CTU	Coding Tree Units
CU	Coding Unit

DASH	Dynamic Adaptive Streaming Over HTTP, MPEG-standard
DCT	Discrete Cosine Transformation
EL	Enhancement Layer
ERP	Equirectangular Projection
FEC	Forward Error Correction
FOV	Field Of View
GFD	Generic File Delivery
GOP	Group Of Picture
HDS	HTTP Dynamic Streaming
HEVC	High Efficiency Video Coding, MPEG-standard
HEVD	HEVC viewport-dependent baseline profile
HEVI	HEVC viewport-independent baseline profile
HHI	Heinrich-Hertz-Institut
HiLS	High Level Syntax
HLS	HTTP Live-streaming
HMD	Head-mounted Display
HQ	High Quality
HVS	Human Visual System
IDTX	Identity Transform
ILP	Inter-layer Prediction
IRT	Institut Für Rundfunktechnik
ISM	Image Saliency Maps
ISO	International Organization for Standardization
ISOBMFF	ISO Base Media File Format, MPEG-standard
ITU	International Telecommunication Union
JEM 7	Joint Exploration Test Model 7
JVET	Joint Video Exploration Team
MANEs	Media-Aware Network Elements

MBMS	Multimedia Broadcast Multicast Service
MB	Macroblock
MCTS	Motion-constrained Tile Set
MC	Motion Compensation
MFU	Media Fragment Unit
MMT	MPEG Media Transport, MPEG-standard
MOS	Mean Opinion Score
MPD	Media Presentation Description
MPEG	Moving Picture Experts Group
MPM	Most Probable Mode
MPU	Media Processing Unit
MTU	Maximum Transmission Unit
MVR	Multicast Virtual Reality
MV	Motion Vector
NAL	Network Abstraction Layer
NBMP	Network-Based Media Processing
OBMC	Overlapped Block Motion Compensation
OMAF	Omnidirectional MediA Format, MPEG-standard
PPS	Picture Parameter Set
PSNR	Peak To-Signal-Noite Ratio
PU	Prediction Unit
QoS	Quality Of Service
QP	Quantization Parameter
QTBT	Quadtree Plus Binary Tree
RAP	Random-access Points
RA	Random Access
RD	Rhombic Dodecahedron Projection
RL	Reference Layer

RNN	Recurrent Neural Network
ROI	Region Of Interest
RSP	Rotated Sphere Projection
RTSP	Real-Time Streaming Protocol
RWQR	Region-wise Quality Ranking
SAP	Stream Access Points
SB	Super-macroblocks
SDT	Signal Dependent Transform
SEI	Supplemental Enhancement Information
SPS	Sequence Parameter Set
SRD	Spatial Relationship Description
SVC	Scalable Video Coding
TS	Transport Stream
TU	Transform Unit
UHD	Ultra High Definition
VCEG	Video Coding Experts Group
VCL	Video Coding Layer
VOD	Video On Demand
VPS	Video Parameter Set
VUI	Video Usability Information
WST	Watershed-Transformation

List of Figures

Listings

Chapter 1

Introduction

360° video is one of the most emerging technologies of the audio-visual media industry of the last years, as will be stated below. Especially due to improvements of available soft and hardware, the generation and playback of high quality 360° video applications has become more and more common. One should keep in mind, that 360° video applications are still a spade area, except for the gaming industry. The exact potential for all industrial sectors is still not completely conceivable. The term omnidirectional-, spherical- or 360°- video, defines video generation and playback, where a user can perceive an entirely surrounding scenery. Every direction and angle is covered by the video and the user can look around. This switching of viewport can be done either by turning the head, when a head-mounted display (HMD) is used or by a computer mouse or touchscreen, i.e. a pointing-device. This concept is not new, but the advance of today's camera and lens systems, storage devices, computer-chips and display technologies paved the way for new kinds of applications and broad availability for many users.

In this sense, the amount of HMDs will grow globally nearly five-fold, from 18 million in 2016 to around 100 million in 2021, whereas 50% will be connected to smartphones and the other 50% will be connected to PC, console or as standalone, as prognosticated in [1]. Considering the imminent launch of 5G in Europe, consumption of video over the internet may arise even more. In 2021, over 78% of the world's data traffic is expected to be covered by videos [1]. In particular, world-wide traffic of virtual and augmented content will grow 12-fold from 22 petabytes per month in 2017 to 254 petabytes per month in 2022 [2]. Not only the application potential, but also the commercial interest in 360° video is immense. Since a lot of expenditures are spend in this scope, the sales induced by virtual reality are forecast to double, from around 9 billion US-Dollar in 2019 to around 19 billion USD in 2021, as visualized in Figure 1.1 of [15].

Most of this conventional and spherical video files are encoded and compressed by common video codecs like MPEG-4/H.264 AVC Part-10, its successor MPEG-H/H.265 HEVC or the open source solutions like VP9 or AV1, as described in Section 3.1. Due to the inevitable higher resolution and the more content that needs to be covered, 360° videos naturally require much higher bandwidth. Hence, a major challenge is still how to efficiently encode and transmit the 360° video streams. This implies new demands for popular streaming technologies, such as HTTP live streaming, Dynamic Adaptive Streaming over HTTP (MPEG-DASH), etc. and for the video codecs. Against this background, the central questions that motivate this thesis are:

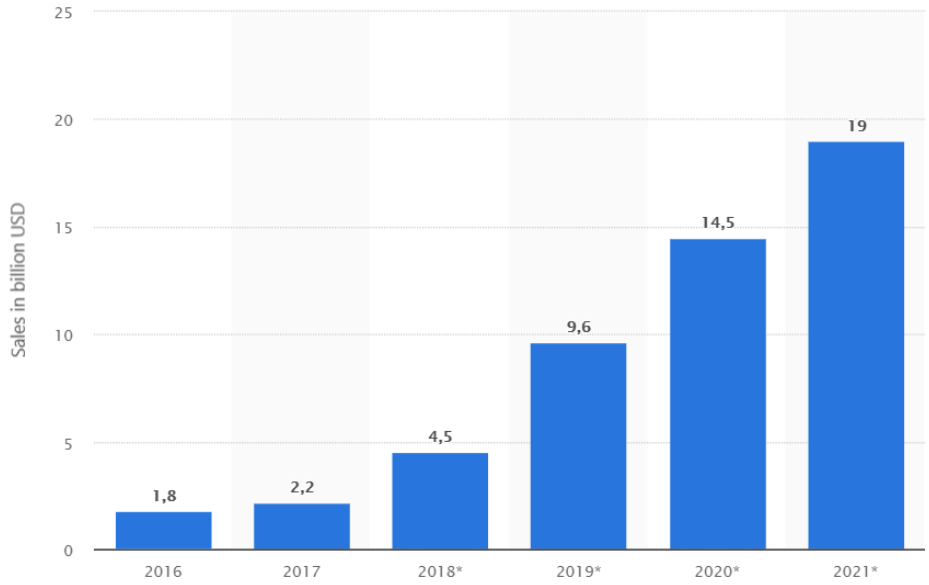


Figure 1.1: Virtual reality sales forecast worldwide from 2016 to 2021, in Billion USD. Source: [15].

- **How can we transmit 360° video content at high quality without a bursting increase of required bandwidth, compared to that of conventional video?**
- **How can we enhance the viewing experience, i.e. the immersion of the spherical video and ensure a playback with less technical impairments, such as buffering pauses, compression artifacts and low resolution?**

By high quality is meant, that the technically measurable video quality, by resolution, frame rate and data rate, significantly exceeds the values for conventional video. Exemplary, at this moment the public broadcasters stream their video content normally at 1280x720 with 25fps at around 4 Mbps. That is, for spherical video content, the resolution, framerate and hereby bitrate should be much higher, e.g. 4K at 60fps or more, as will be stated later in this work. Since all data traffic produces financial costs for the content providers, the goal is to keep the data-rate low.

Some research attacking this problem has been published recently, but no solution has become a industry standard yet. Especially the concept of using slices and tiles, proposed by H.264 and H.265 respectively, has been established and proven as an effective approach. Tiles are regions of images that can be decoded independently from other regions, hereby the spherical video can be split in different parts, covering the viewport, side-regions and the background. This concept can further be used at transmission level, when video content is transmitted in segments, at best corresponding to the tiling scheme of the encoder. Hereby the idea of viewport-dependent streaming can be realized. That is, streaming only that region at high quality, the viewer is currently watching at and the rest at lower quality.

In this thesis popular available technologies for efficient video encoding and streaming will be examined first and recent approaches for efficient viewport-dependent 360° video

streaming will be evaluated in the aftermath. By this, it is tended to construct a more complete understanding of the concepts and peculiarities of omnidirectional video transmission and playback, using up-to-date technologies. As some companies and developers put high effort into development of efficient omnidirectional streaming systems, this work can be understood as overview of emerging technologies of this topic, with an evaluation of some of these, according to the demands of the IRT and the corresponding use case. In this sense, first 360° video is broadly delineated in Chapter 2. An overview of related technologies, like popular video codecs and streaming standards is given in Chapter 3 and the respective Sections 3.1 and 3.2. In Section 3.2.6 the concept of tiled viewport-dependent streaming is introduced, which is one of the key-approaches to achieve acceptable bitrates. This concept is used by all of the tested streaming systems. The first standardized technology is MPEG-I Part 2 Omnidirectional MediA Format (OMAF), [51], described in Sections 4.1.4 and 4.1.5. Other recently proposed efficient approaches of viewport-dependent 360° video streaming are presented in Section 4.3. Some of these approaches propose entire architectures, while others examine parts of it extensively. The main focus will be on the recently introduced OMAF standard, which aims to generalize and standardize the various particular workflows and technologies.

The featured 360° streaming systems are compared and their technical functioning is evaluated theoretically. The practical part and objective of this work is the implementation of some of the presented technologies and approaches for the *Institut für Rundfunktechnik*[1] (IRT) in Chapter 5. It will be investigated, which approaches and technologies are practical and appropriate and which provide what kind of features. The recent OMAF standard will be the main part of this practical evaluation, so OMAF implementations from different developers will be compared and tested on the server system of the IRT. Furthermore, the findings could be used to realize or optimize possible future 360° applications.

The practical part is structured as follows: In Section 5.1, the demands will be carved out and a use case scenario will be drafted. Thereafter, Section 5.2 comprises the technical infrastructure of the IRT, with available software and hardware. For evaluation of the streaming system, test sequences are required, which are self-generated recordings of the Insta360 Pro 2.0 in Hamburg, due to licensing issues, when publishing a spherical video stream online. The creation process of content generation and the technical description of the test sequences is also part of Section 5.2. The single implementations used for evaluation are the HEVC Tile-based GPAC Kvazaar system [6] and the OMAF implementations of Nokia Technologies [108] and Fraunhofer Heinrich-Hertz-Institut [90]. The implementation and evaluation process, i.e. encoding of the test footage and creation of a MPEG-DASH stream, is documented in Section 5.3. These descriptions are based on the guides, tutorials and *Git* documentations of the OMAF Nokia Technologies implementation [108], the GPAC Télécom ParisTech implementations [75] and the OMAF Fraunhofer Heinrich-Hertz-Institut implementation [90]. The created media files are then integrated into the IRT server system. The evaluation of the used implementation approaches is stated in Section 5.4, with the focus on the two OMAF implementations and the used creation modules and playback environments. A conclusion and outlook is given in the aftermath, recapitulating the theoretical parts and analysis of technologies, i.e. Chapters 2 to 4 and the practical implementation, including the evaluation of the tested approaches,

[1] En: Institute for Broadcasting, Munich

with regard to the demands of the IRT.

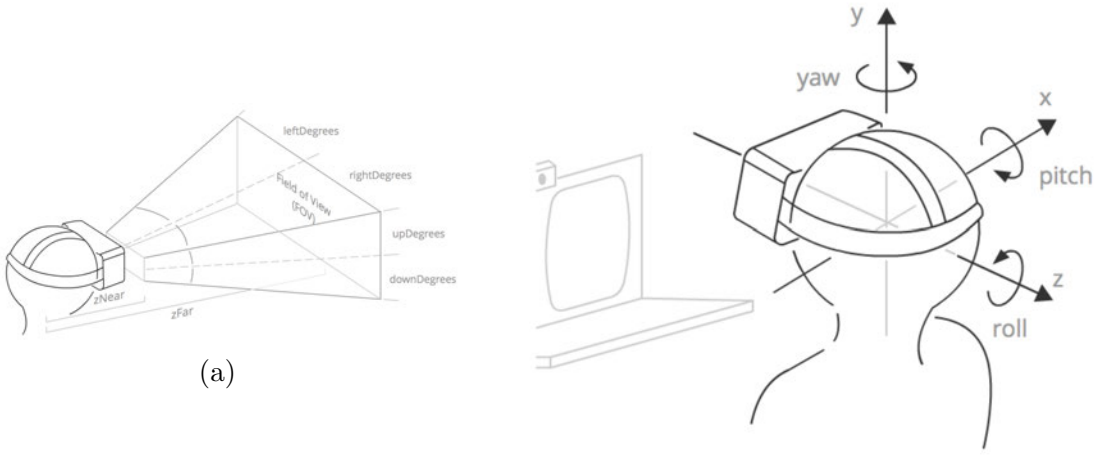
Chapter 2

360° video

Recently, the use of 360° video and the interest in Virtual Reality (VR) and Augmented Reality (AR) applications is rising, primarily due to technical improvements in this field. That is not just the possibilities for streaming of high resolution 360° content, but also the availability of recording and playback devices are increasing. The term 360° video generally does not define the playback device, but is a generic term for all kinds of applications, where a user can watch video content in a 360° surrounding representation, i.e. the content is presented like the inside of a sphere. Common playback devices are head-mounted displays, with integrated displays or via smartphone, and 2D representations for tablets or computers, where the field of view (FOV) can be selected by a pointing device, as a substitute for the head-movement. The FOV is that area of the video, currently being watched by the viewer and played by the device, as depicted in Figure 2.1 a). As stated in [91], the FOV, also referred to as *viewport*, can be defined as the head rotation angles on the X, Y, and Z axes, presented by the Euler angles pitch, yaw and roll respectively, as shown in Figure 2.1 b). The playback device displays only the selected corresponding area of the whole video but not the whole 360° content at all time, which is one of the main ideas when compressing 360° videos and will be stated later more detailed. Although, a free choice of the viewport does not involve, that the viewer has to interact with the environment within the video. Interactivity in 360° video is possible, but not automatically implied. In this work, the main attention is spent on the conventional 360° video, without interaction with the environment, except the free choice of the viewport, which indicates a communication between viewer or client and the provider of the video content.

The FOV or viewport have to be differentiated from the humans field of vision, which is biologically predetermined and describes that area in which objects are visible, when staring in one direction. According to [102], it is separated for each eye in the monocular visual field, which is defined by the inner 30 degrees of vision and represents the central vision and the peripheral visual field, which is about 100 degrees laterally, and vertically 60 degrees upward and 75 degrees downward. The binocular visual field consists of the overlap of the two visual fields, and is about about 114° horizontally, as stated in [60]. Since the remaining 40 degrees of each eye on the left and right sides are only covered by one eye respectively, they are outside the binocular vision. When a 360° video is watched via a HMD, nearly the whole visual field is covered by the display, in contrast to 2D representation on a 2D screen.

The region of interest (ROI) is the area of the video, where the action focus is placed



(a) (b)

Figure 2.1: a) Field of view representation. b) Three-dimensional and Euler axes. Source of both: [16].

and thus it is the region, that attracts the viewer's interest most. Hence, it is the part of the video, that the viewer is moving his viewport to most likely. Further, the content of 360° video is not predetermined. That is, content can be virtually, synthetically animated or recorded by 360° capable camera-systems. The distinction between Augmented Reality and Virtual Reality is only defined by the content and the playback system. AR places virtual objects in our real environment. Hence, the real environment is not replaced entirely, but only expanded by digital content [40]. It is important to note, that in this way, parts of the real environment are masked and hereby the user's perception is already manipulated. VR represents an entire virtual environment [40], without parts of the surrounding real environment. The term 360° video is the overall term and its technical representation is stated more precisely in Section 2.2. In this work, only playback systems with a free choice of the user's viewport are considered as 360° video. Since the term immersion is a key to describe the user's 360° experience, a definition is important. As Grau stated in [57, p. 13]:

> "Immersion can be an intellectually stimulating process; however, in the present as in the past, in most cases immersion is mentally absorbing and a process, a change, a passage from one mental state to another. It is characterized by diminishing critical distance to what is shown and increasing emotional involvement in what is happening."

Hence, an immersive 360° experience, is a situation that makes the user cutting out his consciousness and the virtual environment appears to be real [40]. In relation to 360° video, Grau [57, p.13] differentiates the term immersion, as follows:

> "The intention is to install an artificial world that renders the image space a totality or at least fills the observer's entire field of vision [...]. Unlike, for example, a cycle of frescoes that depicts a temporal sequence of successive images, these images integrate the observer in a 360° space of illusion, or immersion, with unity of time and place."

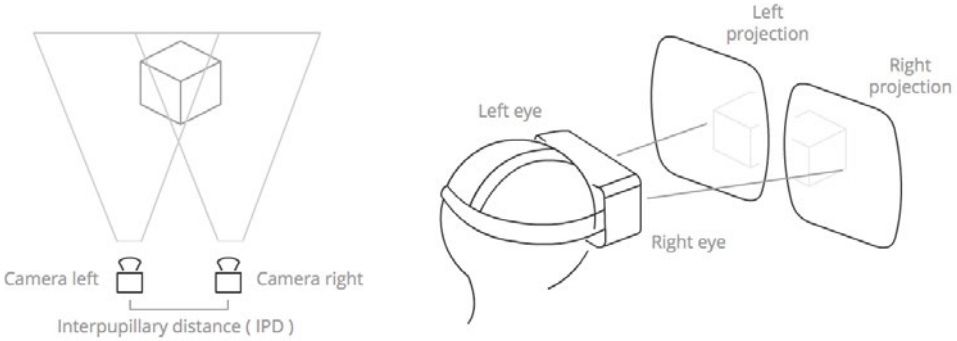


Figure 2.2: Creation of a stereoscopic image and representation via HMD. Source: [16].

2.1 Content generation

Understanding how humans spatial or three-dimensional stereoscopic viewing works, is important to understand how 360° content can be generated and displayed on a VR-capable playback device. This section will focus on playback via a HMD, where the user sees and hears solely the virtual 360° audio-visual content.

Normally, the distance between our two eyes is about 6-7 cm, thus we see objects from two slightly different positions, the closer the focused object, the more our eyes turn into the middle. A focused object, far away from the viewer, results in parallel eye position [94]. Hence, very long distant objects can only be discriminated by our viewing-experiences. This spatial perception gets lost when watching a picture on a paper or screen [94]. Therefore 3D stereoscopic spatial seeing can be produced by presenting two different pictures of two slightly different positions, one for each eye separately [94], as shown in Figure 2.2. The accuracy of the content generation also decides, whether the brain can easily create a spatial impression out of the two separately presented images, to prevent motion sickness.

Today, special camera-systems can record 360° videos, composed by multiple single recording modules, i.e. lens-systems. To record all content around a certain point, i.e. the center of the camera system, at minimum two single optical lens-systems need to record simultaneously. Distortions resulting from wide-angle optics have to be compensated, to generate an omnidirectional image. These pictures need to be merged to a full 360° representation. There are two kinds of omnidirectional images, monoscopic and stereoscopic. Monoscopic systems provide identical images for each eye, disregarding the natural characteristics of the humans visual system. For stereoscopic omnidirectional video, separation of the two images for each eye needs to be done by recording the same content out of slightly different positions, matching the perspective for the left and the right eye. Alternatively, the difference between the two slightly different pictures, that are presented to each eye, can be calculated. Besides, virtual 360° content can be generated via 3D modelling on a computer. However, for both kinds of omnidirectional representation, errors resulting from incorrect merging, the so-called stitching of the several single recorded pictures can occur, which are called stitching errors. These errors can be reduced by recording with more than two lens-systems, so more data at the edges of the single images can be used for calculation. This section illustrates the recording of such content,

Figure 2.3: Stitched equirectangular sample frame from a H.265 320 Mbps 8K 2D recording with the Insta360 Pro 2.0

at the example of the Insta360 Pro 2.0 camera, with reference to [9]. The Insta360 Pro 2.0 is equipped with 6 single F2.4 fish-eye lenses with 10.57 mm focal length. Each of which covering 1/3 of the whole 360° sphere, hence stereoscopic recording is possible. Real-time Stitching is only activated using H.264 encoding and only up to a resolution of 3840 x 3840 pixels with 30 fps. Higher resolutions and H.265 encoding are only activated with stitching in post-processing. Data-rates up to 120 Mbps are disposable per lens. The 2D content representations can be of different formats and shape. One example for a stitched, post-processed recording of the Insta360 Pro 2.0 is illustrated in Figure 2.3, to introduce a possible 2D transformation of a 360° camera recording. In the following section, the 360° content representations and sphere to planar mapping methods are examined in detail.

2.2 Content representation

The idea of 360° video is not new. As aforementioned, 360° video content can be displayed by different playback devices, even though, the most common one is the HMD. The first HMDs have been developed in the 1960, when Morton Heilig patented his concept of a HMD, where users could watch short video content in a 3D stereoscopic way, although with stereo audio playback [40, p. 25]. Today's HMDs work with the same principle by displaying to images, separately for each eye, in case of stereoscopic representation out of slightly different perspectives. They are extra equipped with head- and/or eye-tracking sensors, detecting the viewer's head movement, and hereby his viewport, to change the displayed content accordingly. Further, the resolution of the video content and the installed displays are decisive for an immersive 360° application. Considering the short distance between eye and display, the viewer might be able to distinguish between single pixels, when presenting videos in low-resolution. High resolution and high framerates are recommended, to enhance the depiction of the 360° environment.

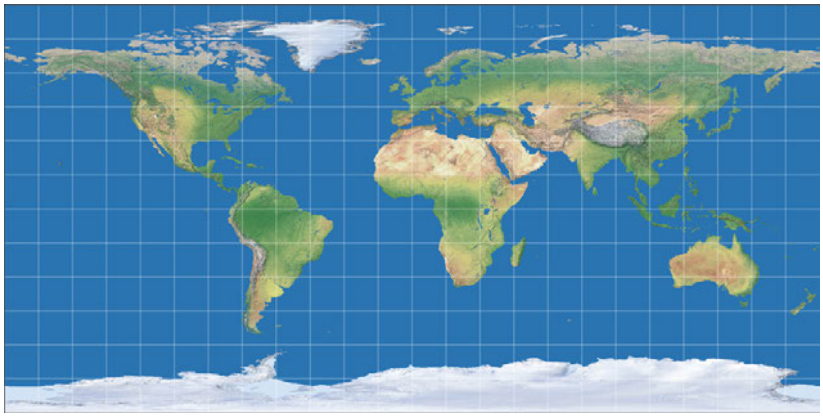


Figure 2.4: Equirectangular (0°) world map projection. Source: [117].

Transformation of the recorded or generated spherical video content is necessary for efficient en- and decoding, transmission and displaying, in order to generate a single planar image for each eye. Especially, when using reliable popular video codecs. This transformation of spherical content into a planar representation is called projection or mapping. Several sphere-to-planar mapping methods have been established, some of which have been proposed recently, optimizing prior approaches by minimizing shape distortion. Those mapping methods can be categorized into two groups, depending on whether the 360° content is projected with constant quality over all parts of the sphere or with quality grades, i.e. variable quality, preserving higher quality for parts of the viewport and lower quality for side regions [38]. First, the two main mapping methods, equirectangular projection and cubemap projection will be explained and afterwards other approaches like i.a. Rhombic Dodecahedron Mapping, Pyramid Mapping and tile-segmented Mapping will be introduced.

1. **Equirectangular projection**
   Because of its simplicity and compatibility equirectangular projection (ERP) is very common and widley used [39]. The inner surface of a sphere is unfold on a two-dimensional rectangular map, as depicted in Figure 2.4, interpolating or duplicating the missing pixels towards the poles. The width and height, according to the circle properties are $2\pi r$ and πr respectively, where r is the radius of the sphere [38][39]. No special playback device is necesseray, since this content can be displayed by conventional screens. Contrarily, redundant information at the poles, resulting from stretching, i.e. interpolating pixels, lead to an increase in bandwidth [39]. Bitrate or bandwidth-consumption should be treated very economical, as will be stated later more detailed.

2. **Cubemap projection**
   Another common projection method is the cubemap projection, which is supported by various graphic libraries. First the sphere-surface is projected onto a cube's surface and afterwards a 2D map from the cube is generated, without creating as many redundant pixels towards the poles of the sphere [39]. The most common cubemap projection uses a six-sided cube, each of which covering a part of the spheres surface. Each perspective with a 90° angle and each looking down an axis

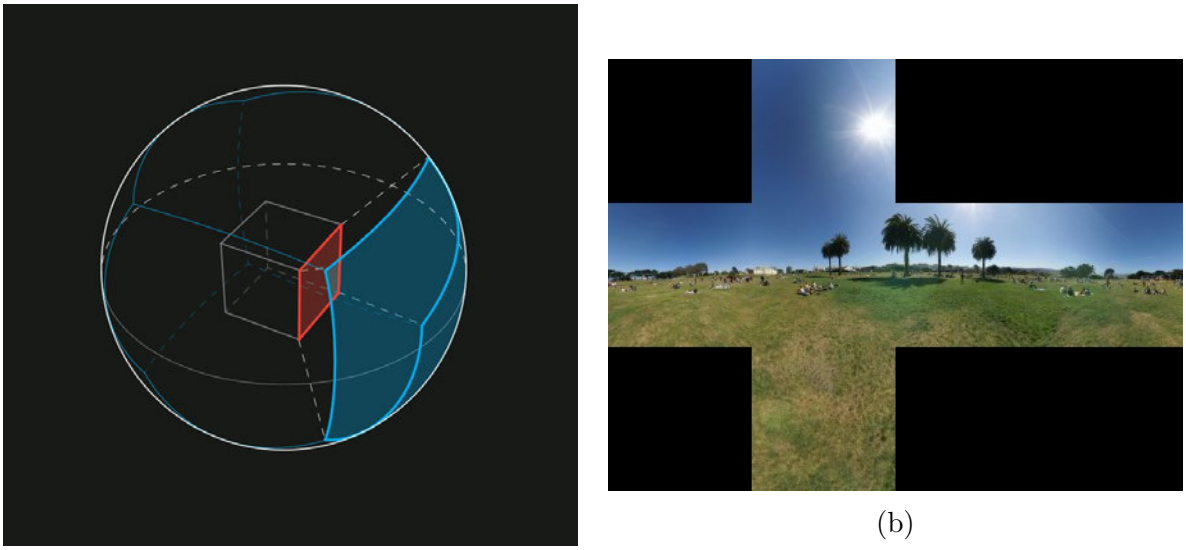
of a coordinate system, with it is origin at the viewport, i.e. the camera, as stated in [58]. The single steps of sphere-to-cube projection and cube-to-planar projection are illustrated in Figures 2.5a and 2.5b, according to [71]. Multi-resolution cubemap and equirectangle projection approaches are introduced in [103], where the viewport is streamed at higher resolution, than the side and background regions, in a 50/50 ratio of the overall resolution, as depicted in Figure 2.5c.

3. **Rhombic Dodecahedron projection**
   As stated in [53], Rhombic Dodecahedron projection (RD-projection) subdivides the sphere in 12 normally equal-sized parts, i.e. rhombi or pentagons. The dodechaedron is built by inscribing a cube into an octahedron, where the cube's edges touch the edge midpoints of the octahedron. Combining the eight cube vertices and six octahedron vertices results in an equal 12-sided dodecahedron, whose rhombi can be projected onto a 4π sphere surface via Gnomic projection [53] as shown in Figure 2.6a. To get a planar 2D picture, compatible to conventional video en- and decoders, a re-segmentation has to be done, which is shown in Figure 2.6b and 2.6c.

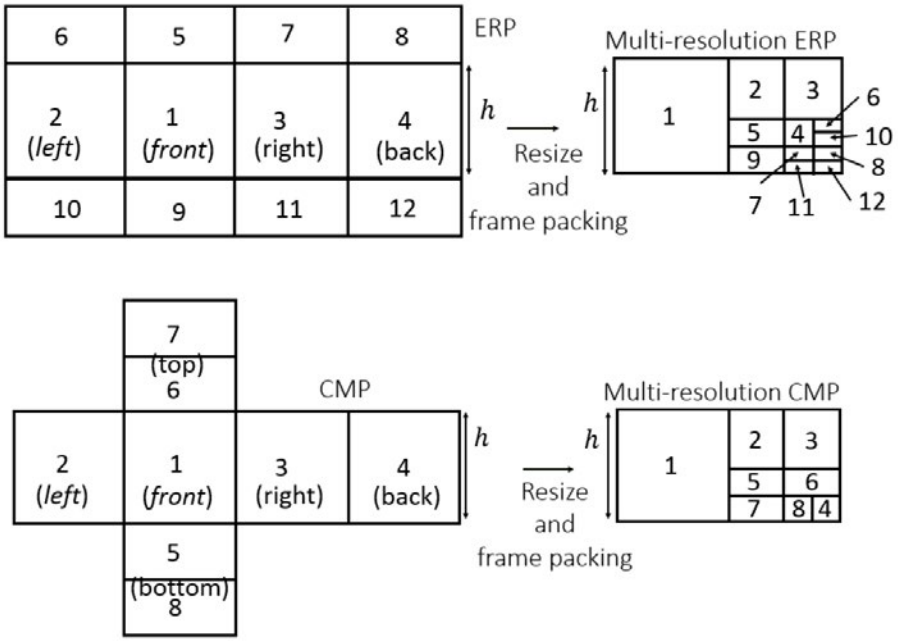
4. **Pyramidal projection**
   One approach of pyramidal projection is proposed by developers of Facebook [72], where the spherical 360° content is projected onto a six-sided regular pyramid, as depicted in Figure 2.7a. The proposed approach of [72] uses different video qualities for different parts of the mapping area, i.e. the base of the pyramid containing the FOV is maintained high, while the resolution of the pyramidal sides, representing the side area and the area behind the viewer, is reduced. As illustrated in Figure 2.7b, the unwarped pyramidal representation can be processed as a rectangular image, by stretching the sides to fit the full 360° area. The main idea is to switch the pyramid, which the user is looking at, when shifting the perspective. The viewer steps into a new pyramid, when changing the perspective of the viewport. Therefore, the sphere is covered by 30 different viewports in total, each covering an area of around 30°. Quality adaptation is introduced by offering five different resolutions for each viewport, resulting in 150 different viewports. Hence, first transformation and transcoding are processed and afterwards each pyramidal viewport representation is stored on the server, instead of computing each viewport in real-time, for each client-request.

   Two modifications of the pyramidal mapping are treated by Kammachi-Sreedhar et al. in [103]. They change the geometrical arrangement in two different ways, to address coding inefficiency caused by sharp diagonal edges between front and side faces of the pyramid. These alternative arrangements are illustrated in Figures 2.7c and 2.7d, howbeit the proposed frame-packing is only one of many possible solutions [103].

5. **Tile-segmented projection**
   In 2015 Yu et al. [118] proposed a tile-segmented sphere-to-planar mapping approach. As shown in Figure 2.8a, a equirectangular representation of a sphere is divided into several striped latitude pieces, i.e. tiles. Where the density of tiling and sampling can be adjusted by changing the size of the tiles. This tile-segmented projection is quality-adaptive by changing the number of tiles according to the view-



(a) (b)



(c)

Figure 2.5: a) Sphere-to-cubemap projection. Source: [71]; b) Example of an unfolded cubemap, by Emil Peerson. Source: [89], Creative Commons Attribution 3.0 Unported License; c) Multi-resolution equirectangular and cubemap projection arrangements. Source: [103].

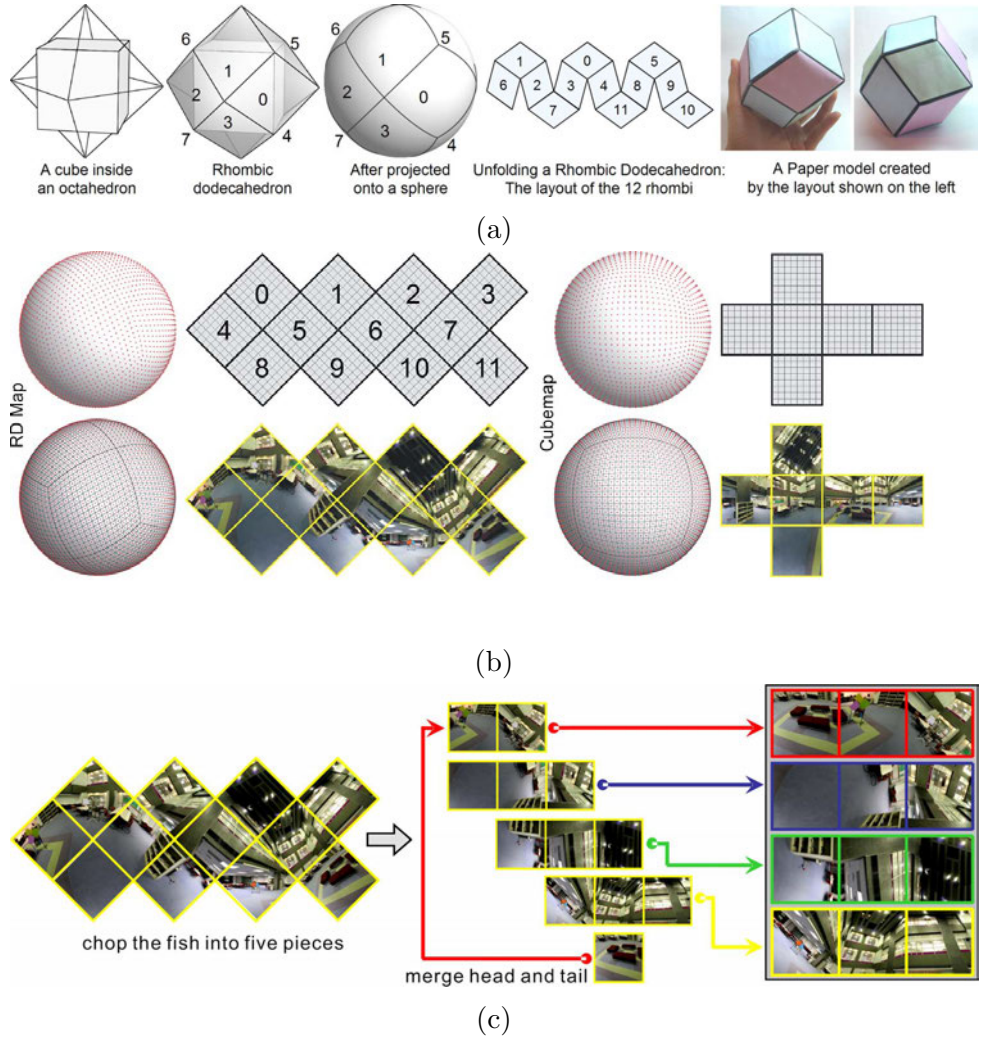


Figure 2.6: a) Geometric structure of a rhombic Dodecahedron, b) Distinction between cubemap and Dodecahedron mapping, c) Re-segmentation of the unfolded Dodecahedron, for encoding. Source: [53].

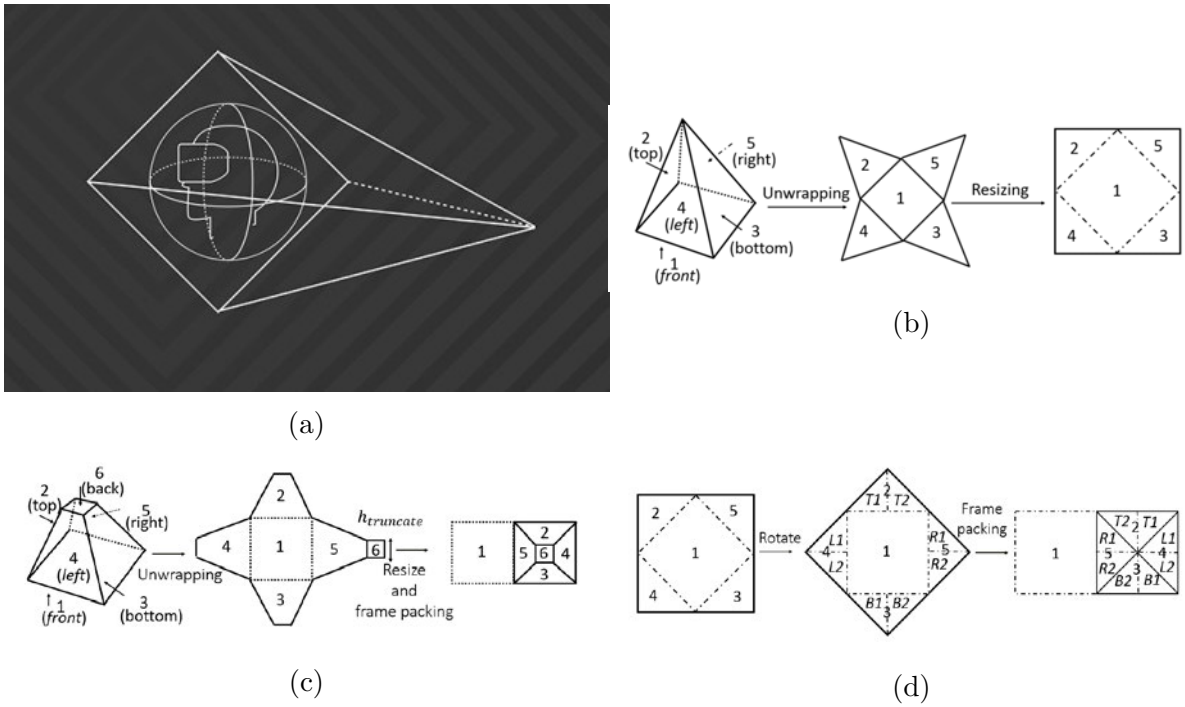


Figure 2.7: a) Viewport orientation and pyramidal projection of [72]; b) Geometric structure of rhombic pyramidal mapping, c) Truncated pyramidal mapping, d) Square pyramidal mapping. Source: [103].

ers behaviour. Content in the ROI can be processed prioritized at a higher quality and less important or detailed regions can be coded at lower quality. To overcome artifacts at the edges of tiles, they use overlapping tiles. Li et al. optimized this approach by changing the tiling structure and re-segmentation of the tiled pieces for encoding [79], as depicted in Figures 2.8b, 2.8c and 2.8d. Instead of tiling after equirectangular mapping, the sphere is cut into several tiles first and the single tiles are cut open, unfold and reshaped into rectangular stripes afterwards. The cambers at the poles are treated differently, as two circles and for re-segmentation filled black, due to reshape them into rectangular representation.

Some of these sphere-to-planar mapping methods have been modified or extended, to gain coding efficiency and quality enhancement. An offset-cubemap projection scheme is proposed by Facebook [73] and reported in [38]. The Offset-cubemap projection is a variable quality mapping, where, unlike to the aforementioned cubemap projection, the viewer's position is pushed backwards from the centre of the cube, by a predefined offset, as illustrated in Figure 2.9a [38][39]. The quality of the extended viewport in front of the viewer is set higher. The viewports to the sides and backwards of the viewer have gradually reduced quality, as shown in Figures 2.9b, 2.9c, 2.9d. The coding- and compatibility advantages are still kept up and pixel interpolation at the side edges is used, to overcome artifacts, caused by missing pixel information at the edges of each cube side, referring to [73] and [38].

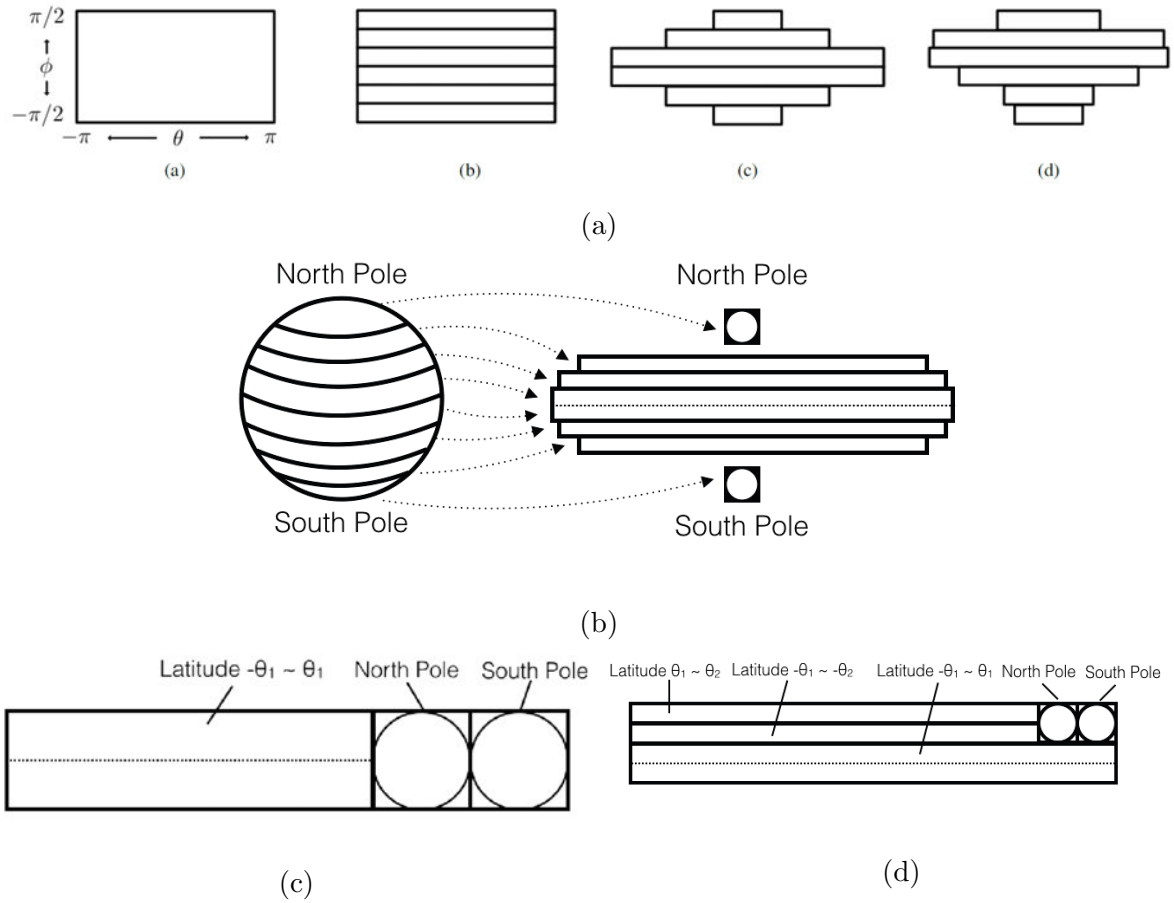


Figure 2.8: a) Tile-segmented projection by Yu et al. [118]. b) c) d) Sphere-to-planar tile-segmented projection scheme and segmentation and frame-packing for encoding of Li et al. Source: [79].

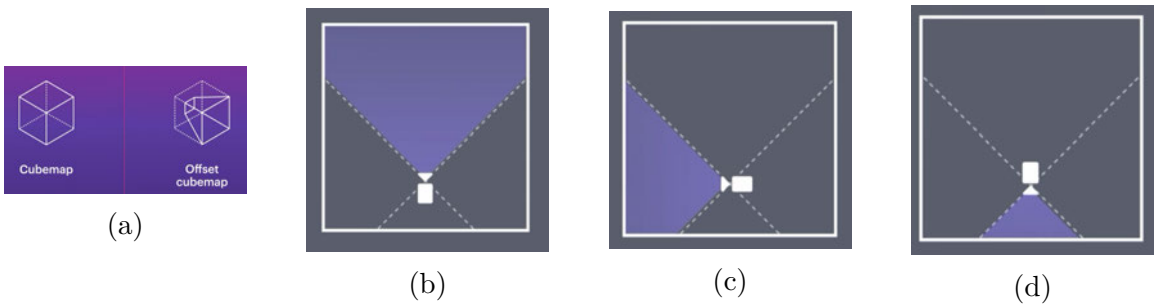


Figure 2.9: a) Conventional cubemap (left) and offset-cubemap (right). Source: [73]; b), c), d) Quality reduction according to viewport changes. Source: [73].

2.3 Content distribution

The distribution of 360° video consists of different processing modules and entities. With reference to the VR-industry forum, a working group, specifying recommendations for the use of omnidirectional video in [52], the following work-flow for content distribution is expounded. The capturing is done by a content provider, applying the stitched content to the service provider separately for audio and video. This module pre-processes and encodes the applied audio and video content, in one of the respective 2D-representations discussed before, producing elementary streams at the outlet. Moreover, the separated media parts are then encapsulated, e.g. in segmented tracks and delivered over a Content Delivery Network (CDN) to a VR service module. This module consists of a computing-platform and the application module or playback device. This service module communicates with the CDN to manage viewport changes adequately, measured by sensors of the HMD or by input of a pointing device. The service platform decapsulates the segmented stream, e.g. the single tracks, and decodes the elementary streams for audio and video. It is worth to be note, that the playback device and the service platform can be merged in one single module, or separated into two devices. The service platform, e.g. a computer, smartphone, etc. is responsible for correct decoding of the incoming bitstream, whereas the playback device playbacks the rendered audio and video stream and tracks the head- and eye-movements. Detailed concepts of content distribution are the key aspects of this work and will be explained more detailed in the following chapters.

Chapter 3

Overview of related technologies and clarifying terms

3.1 Video coding technologies

Recorded or artificially generated digital video content has to be encoded in some form and packed into a bitstream to provide subsequent access to the digital data. For this purpose, various coding methods have been developed, to pack the data either lossy or lossless in a particular codec. Lossy video compression uses various techniques to efficiently reduce bitrates for transmission, while at best preventing visible image artifacts. The output data does not exactly match the input data in this case. Most compression techniques use similar base technologies, like transformations of the data, dependencies of the image content of neighbouring frames, dependencies and similarities of image parts within a frame, compressing strings of a data set, prediction of possible movements of objects within the image, and segmentation structures such as blocks or tree structures to put pixels into groups, to reduce the overall amount of data. Thus, the individual frames and pixels of a sequence are not stored entirely, but redundant and irrelevant data is identified and omitted. The approach of reducing irrelevant data often depends on the characteristics of the human visual system (HVS). The most common and widely used lossy video codecs are the various MPEG and H.26X versions developed by the Video Coding Experts Group (VCEG), a working group of the International Telecommunication Union (ITU) and Moving Picture Experts Group (MPEG) of ISO and IEC and the associated mechanisms and algorithms designed to make the transmission, storage and access of video data more efficient. MPEG[1][2] is a working group of the ISO/IEC developing standardizations for coded representation of digital audiovisual data. The goal is to improve the digital media experience of the user and to unitize technological improvements, making them better available for the audiovisual industry.

To understand the demands and peculiarities of efficient 360° video streaming, it is necessary to understand the mechanisms of encoding and decoding videos and to know how basic image compression works. The functioning of inter- and intra-frame coding, structure of macroblocks in MPEG, motion compensation, transform coding, discrete cosine transformation (DCT), quantization and entropy coding, are stated precisely by

[1] Official working group name: ISO/IEC JTC1/SC29/WG11

[2] https://mpeg.chiariglione.org/

Watkinson in [116], Milde in [84], Richardson in [93] and Sze, Budagavi and Sullivan in [107]. The following section starts with the functioning of MPEG-video coding, at the example of MPEG-4/H.264 as a still widely used codec and High Efficiency Video Coding (HEVC)/H.265, as its powerful successor, followed by other state-of-the-art video codecs. The principle of JPEG still-image compression is used for MPEG video coding. It is based on the DCT, to convert image-values (YCbCr, YUV etc.) into DC and AC components and a subsequent quantization to get rid of high-frequency values, less visible for the HVS. This concept will not further be discussed in this work, as it is a wide topic and less relevant for the understanding of a 360° streaming-architecture. This principle of image compression is stated precisely by Watkinson in [116] and by Richardson in [93]. Recently, AV1 and JEM, as two powerful and high-efficient codecs have been released, competitive to the performance of H.265. The base-functioning of AV1 and JEM will be explained later in this section and at the end, those four codecs will be shortly evaluated, based on related works.

As aforementioned, bitrate should not be wasted by the use of inefficient coding tools. To reduce bitrate and to provide decent image quality, different video compressing standardizations can be used. In the following, the overall principle of compressing a video sequence is explained, to maintain acceptable quality, but reduce data. According to [116, p. 13], a video signal is generally presented as a temporal sequence of images of a certain size x, y. Even 360° video, is presented to the eye in this two-dimensional way. The whole video sequence can be described in three dimensions, the direction of the time axis and each dimension of the direction of the spatial axes. A further dimension results from the properties of each pixel. The distinction of the way a frame is coded and what dependencies to other images may arise, depend on whether it was intra- or inter-coded.

Frequently, neighboring individual images of digital video sequences have similarities and differ only slightly to the preceding or following frame. In many lossy video codecs, this property is used and not a complete frame is stored, but only differences from the preceding or following frame. A frame or block that can only be decoded with a so-called residual information together with another frame is called inter-frame coded, inter-coded or inter-predicted. The functioning of this inter-coding is not only based on similarities between neighbouring pixels, but also uses prediction of the motion of objects of neighboured frames. This is called motion prediction or motion compensation (MC). This inter-coding, with motion prediction can reduce the amount of data enormously and is one of the base ideas of video compression [116, p. 15].

A frame or block that is encoded completely independent to other frames and is compressed only within itself, can be called inter-coded and referred to as key or I-frame. Here, the time axis does not affect the compression of the frame to be encoded. In intra-coding the compression is only achieved by compressing in the spatial dimensions and can therefore also be called spatial coding, intra-coding or intra-prediction. Often, intra-coding uses compression methods developed for still images. An example of this is the compression using methods like JPEG, which is applied in MPEG-2 [116, p. 14-15].

A video sequence can be presented in form of different frequencies. Intra-coding exploits the fact that the HVS is less sensitive to high-frequency parts of the image and can only perceive them to a certain extent. Individual images often have blocks in which identical or similar pixels are present. This property increases, the higher the spatial frequency of the single image is. To simply omit the high frequency fractions of an image

would result in softening and visibly blur the image. The amplitude of the spatial components coincides with the spatial frequency, which is utilized for the compression. Thus, when the local frequency spectrum is divided into frequency bands, the high-frequency components can be represented by fewer bits, not only because their amplitudes are lower but also because the eye perceives the hereby increased noise less strongly. In order to describe the frequency ranges of two-dimensional images, transformations such as the DCT are applied.

In intra-coding, theoretically the complete data of each frame can be transferred, depending on the quantizing parameter. The resulting data quantity is significantly higher, but the processing effort while decoding is reduced, compared to inter-coding. Even with a high quantization parameter, the bitrate reduction between intra-only modes and inter-coding is not comparable. On the other hand, coding methods that use inter-coding can compress more strongly, by evoking higher decoding expenditure, since the resulting frame must first be calculated from its predecessor [116, p. 13-14].

Changes in transfer-rates and data-loss can influence the decoding of a streamed video sequence. Artifacts can occur if the necessary data of a dependent frame does not exist at the decoder. H.264 and H.265 usually use intra- and inter-coding and have different mechanisms to prevent package-loss and artifacts, H.265 better than H.264. Besides, there exist scalable extensions for streaming, such as SVC for H.264 and SHVC for H.265. These extensions optimize the coding architecture especially for video streaming applications.

3.1.1 H.264

H.264/MPEG-4 AVC[3][46] is based on the basic mechanisms of MPEG-2, but has been optimized and new functionalities. It was developed by the VCEG and the MPEG. In the following, only the differences and advantages compared to MPEG-2 are stated, referring to Richardson [93] and Marpe et al. [82]. For base information on MPEG video coding, the reader is referred to Watkinson [116]. The basic functions of the MC and the idea of storing only differential information and no complete picture data, by using a frame structure of intra-frames (I-frames), prediction-frames (P-frames), and bidirectional prediction-frames (B-frames), have been forwarded from MPEG-2. Depending on the application field and the transmission environment, different profiles can be set up in H.264, each of which is suitable for different applications. A distinction is made between baseline profile (BLP), main profile and extended profile, which offer different and extended functionalities. One of the introduced key features is the concept of slices. A slice is a set of macroblocks, i.e. one part of an image, that is independently decodable from other slices. It is a useful feature to classify dependencies for different regions of a frame. In the BLP intra- and inter-coding through I- and P-slices, and entropy coding with context-adaptive variable length coding (CAVLC) is used. The main profile also supports B-slices as well as weighted prediction for inter-coding, entropy coding with context-adaptive binary arithmetic coding (CABAC) and interlaced processing. The extended profile does not support these extensions of the main profile, but introduces other features that prevent package loss and enable a video stream to be changed. The most common profile is the main profile, which provides efficient video coding with appropriate complexity [93, p. 162].

Important for improvement of the picture quality is the optimized macroblock (MB)

[3]ITU-T H.264; ISO/IEC 14496-10

structure and the use of slices in H.264. Further, the MC and inter-coding as well as the sample prediction and thus the intra-coding were improved in the standard. Furthermore, H.264 also uses the transformation of prediction residuals, which is called transform-coding. These transforms are performed in different block-sizes with different integer-transformations based on the DCT. In block-based video coding, blocking artifacts can occur, when the target bitrate is too low. To overcome these block artifacts, in H.264 a so-called Content Adaptive in-loop Deblocking Filter is applied [82].

3.1.2 High Efficiency Video Coding (HEVC)

MPEG-H/H.265 HEVC[4][48] is a video coding standard released in 2012 by the ITU-T VCEG and the ISO/IEC MPEG, specifically designed to obtain a significant reduction in bitrates, about 50% higher compared to its predecessor MPEG-4/H.264 AVC, meanwhile preserving consistent video quality [106]. The following descriptions are based on the descriptions of Sullivan et al. in [106] and Sze et al. in [107]. The uprising development of high-resolution and high-framerate video systems, the accompanying development of spherical video and plenty of other applications, call for more efficient coding standards to save data while, reducing visual artifacts. HEVC thus gradually usurps the place of the widely used H.264 codec, addressing video content requiring high image quality, at the expense of higher encoding and decoding times. HEVC is better adapted to parallel processing architectures, which counteracts the increased computational power. Besides defining the coding-structure, the developers propose a reference implementation of the standard.

As in previous standards, different profiles and levels are defined, addressing different applications and use cases. The profiles define the coding mechanisms used and the levels define the coding parameters such as maximum bitrate, etc. In addition, tiers were introduced that define two different bitrates within the levels. Initially, three profiles were predefined: Main Profile, Main Still Picture Profile, and Main 10 Profile. The Main Profile with 8-Bit per sample, 4:2:0 chroma subsampling, is the standard profile for video sequences in the consumer area. The Main Still Picture Profile can be used for the compression of single images, similar to JPEG. The Main 10 profile provides 10-bits per sample and overall higher quality of color representation. The Main 10 profile offers chroma subsampling up to 4:4:4 and bit depths up to 16 bit. In newer versions, the Multiview Main Profile, the Scalable Main Profile and the Scalable Main 10 profiles have been added [106], also referred to as SHVC.

The coding efficiency is achieved by extending and optimizing the basic operating principle of H.264, mainly by improvements of the block structure, the intra-coding, the inter-coding, the entropy coding and the High Level Syntax. As shown in Figure 3.1, demonstrating the coding-scheme of HEVC, the basic process of motion estimation and compensation, the resulting inter- and intra-coding, subsequent transformation and quantization and finally the entropy coding are comparable to H.264/AVC. As HEVC's in-loop filtering has been changed, new processing steps have been added.

[4]ITU-T H.265; ISO/IEC 23008-2

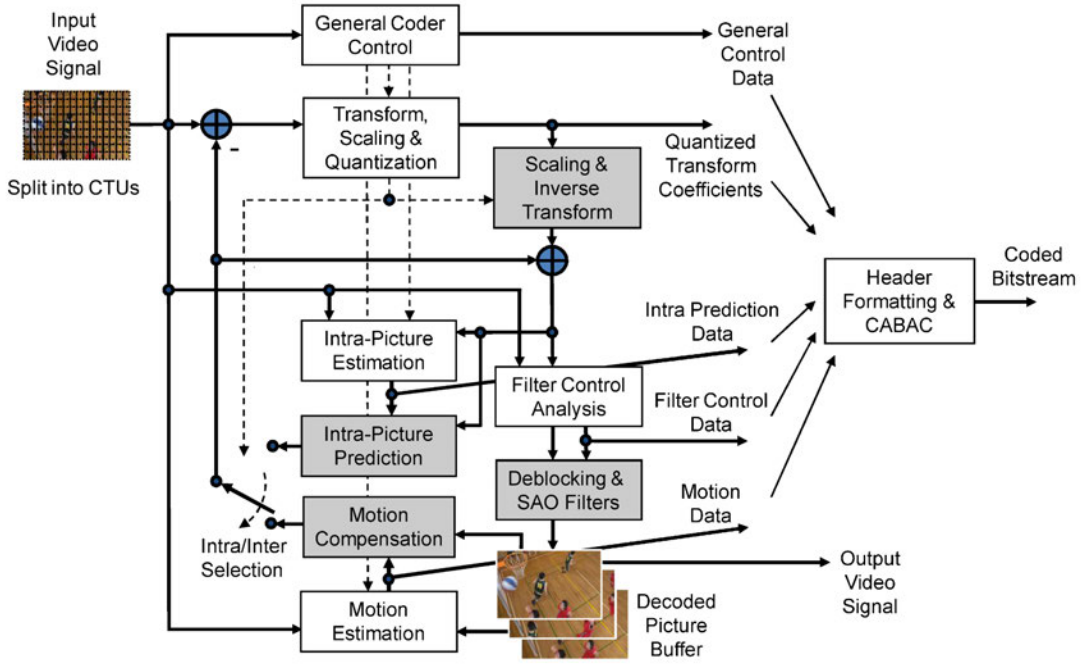


Figure 3.1: Typical HEVC coding scheme, by Sullivan et al. Source: [106].

3.1.2.1 Segmentation of Processing Units

One of the key aspects for achieving higher compression in HEVC, is the segmentation of the luma and chroma processing units. In H.264, pixels were grouped into 16x16 macroblocks. In HEVC, the corresponding structure is a tree-like division into Coding Tree Units (CTU), which are divided into luma and chroma by Coding Tree Blocks (CTB) [106]. These blocks have sizes of NxN of N=8, 16, 32, or 64 pixels and can be subdivided into smaller blocks. In HEVC a CTB can consist out of a single coding unit (CU), but can also be divided into several coding units. These coding units consist of a luma coding block (CB) and usually two chroma CBs with the associated syntax information. These CUs are in turn divided into prediction units (PU) and transform units (TU) [106]. Figure 3.2 shows the tree-structure of HEVC and the division and subdivsion of these single processing units, according to Sullivan et al. [106]. The exact processing of these units and the single steps of inter-/intra-coding and transformation are described in [107].

3.1.2.2 High Level Syntax

As stated by Sullivan et al. in [106], the High Level Syntax (HiLS) defines the structure of a bitstream, providing information of slices or full images, e.g. the spatial resolution or the encoding settings to be used. In HEVC, the HiLS is defined by a Network Abstraction Layer (NAL). Thus, a bitstream in HEVC consists of a sequence of NAL units, which is transmitted as an even number of bits. The first two bytes define the NAL unit header, the remaining define the payload. These NAL units transmit either slices, with slice header and data, or parameter sets that provide additional information. Depending on whether a NAL unit contains encoded images, or other necessary data, it is called Video Coding Layer (VCL) NAL unit or is classified as a non-VCL NAL unit, respectively. The

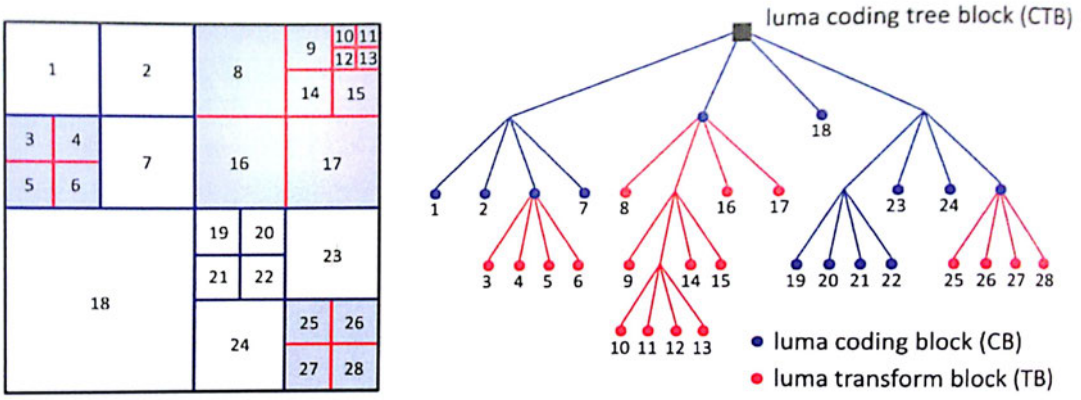


Figure 3.2: Dividing a luma-CTB in CBs (blue) and TBs (red). The numbers represent the processing order of encoding the TB.

NAL provides the data uniformly for various transport layers, such as RTP/IP, ISO MP4 and H.222.0/MPEG-2 Systems. A parameter set contains fixed information, that may not further be changed and enables decoding of a large amount of NAL units. For HEVC there are three kinds of parameter sets:

1. Video Parameter Set (VPS) - A set providing unified information for all layers of a bitstream.
2. Sequence Parameter Set (SPS) - A set providing information for several sequences.
3. Picture Parameter Set (PPS) - A set providing information for several frames.

NAL units may contain Supplemental Enhancement Information (SEI) and Video Usability Information (VUI) metadata. These provide additional information about the temporal use of video frames, i.e. timing, the correct interpretation of the color space used in the video signal, 3D-stereoscopic additional information, 360°-specific information and other additional information. The parameter set structure ensures a robust access structure.

3.1.2.3 Slices

This section relates to the descriptions of Sze et al. in [107] and Sullivan et al. in [106]. As aforementioned, slices represent parts of an image or an entire image that can be decoded independently from other slices of the same image. As macroblocks in HEVC are replaced by CTUs, they are a single or a sequence of CTUs and are processed in raster-scan order. Slices are coded by entropy coding, signal prediction and reconstruction, to compensate data loss during packetized transmission. In order to minimize the overhead of a packet and to keep the size of a slice within the prescribed limits, the maximum size of the payload of a slice is limited. Meanwhile, varying the number of CTUs in the slice.

The division into slices, shown in Figure 3.3a, is particularly useful for three aspects of coding: The reduction of errors caused by transmission, the maximum size of the portable units, and parallel processing. In the case of data losses, transmission-errors

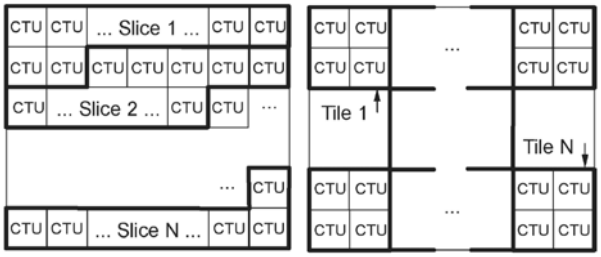


Figure 3.3: Segmentation into slices a) and tiles b). Source: [106].

can be compensated more easily, since independently decodeable image parts are located in different transmission units. The second aspect concerns the maximum size of the transmission units, also Maximum Transmission Unit (MTU). The so-called MTU-Size-Matching prescribes a maximum size of the payload, i.e. the maximum number of bits in the payload of a slice. The number of CTUs within the different slices can vary, to comply with this maximum bit count. The third aspect, why a division into slices is useful, is the parallel processing of data. Slices and the applied processing can be carried out independently of one another, i.e. in parallel.

A slice always consists of a header and the corresponding data, whose minimum size is determined by the CTUs. The independence of one slice to another requires certain conditions. For this purpose, the processing of the coding must be carried out in raster-scan and possible dependencies between CTUs of different slices must be broken up. Breaking down the dependencies of individual CTUs can also have a negative impact on coding efficiency. Thus, the more slices per image, the lower the data saving and thus the coding efficiency.

To overcome these drawbacks, HEVC optimizes the slice structure by dividing a slice into two levels. At the first level, a slice is divided into multiple slice segments at the boundaries of each CTU, as depicted in Figure 3.3a. The first slice segment is completely transferred and contains the complete header and data. The following slice segments have a very reduced header and are dependent on this first slice-segment. By defining these dependent slice-segments at the borders of the CTUs the coding efficiency is not affected that strongly. The second level are the so-called slice segment subsets or substreams, consisting of all encoded bits of the contained CTUs or their subdivisions.

3.1.2.4 Tiles

The ability to split images into rectangular tiles, as shown in Figure 3.3b, has been introduced in HEVC, to facilitate parallel processing and minimize data loss, and allows spatial random access to specific regions of an image. This concept is precisely stated by Sullivan et al. in [106] and Sze et al. in [107], and outlined based on these, in the following. Certain regions of an image can be coded independently of each other but with one header. This so-called shared header is transmitted within slices that combine several tiles. If different tiles are in the same slice, they can be transferred with the same header information. Normally, an image is partitioned into a certain number of rectangular tiles with the same number of CTUs in each tile. Tiles enable parallel processing at an

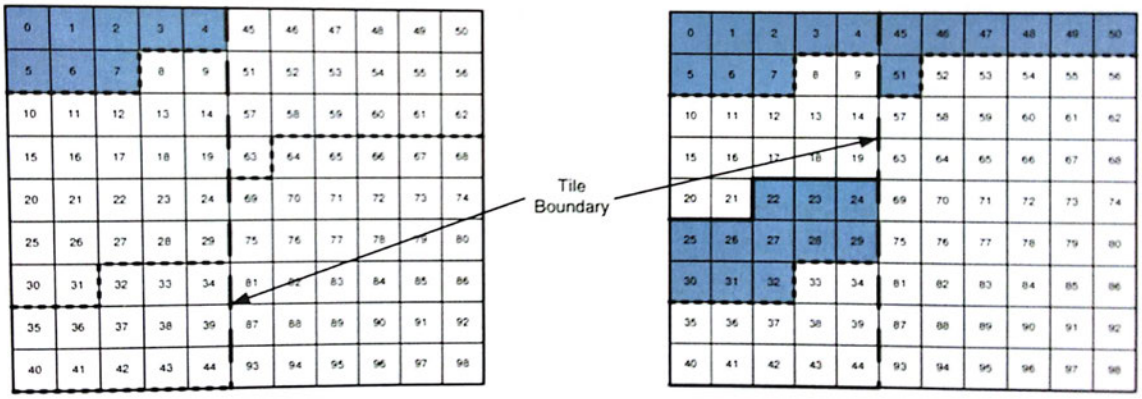


Figure 3.4: Two examples of dividing a picture into tiles, slices and slice segments. On the left, a slice (solid bold lines) with four dependent slice segments (dotted lines) and two tiles (dashed bold lines). On the right, three slices, each of which with a dependent slice segment. Source: [107].

even lower level, than slices and no complex synchronization of the individual threads is necessary, as is the case with block level parallelization in H.264. In addition, the dependencies between individual CTUs are omitted by tiles [106][107].

The size and number of tiles can be defined per picture or for a complete sequence of pictures. The information is transmitted to the decoder in the PPS, a NAL unit, by breaking up prediction dependencies, since independent individual packets can be transmitted, addressing error-proneness.

In-loop filtering, can be applied to the boundaries of nearby tiles, thereby impeding blocking artifacts at the boundaries of the tiles [107]. As mentioned earlier, tiles, slices, and slice segments can be used together to break up a picture into different parts. Figure 3.4 shows a possible segmentation of an image into one (left) or three slices (right), with four or three dependent slice segments respectively. Both image representations are further divided into two tiles, at the boundaries of the slice segments. Independence of tiles and slices is only given, when all CTUs in a slice belong to the same tile, or vice versa. At slice segment level, all CTUs must belong to the same tile, or all CTUs in a tile must belong to the same slice segment. This means that either a slice can be divided into several tiles, or that a tile can be split into several slices [107].

Overall, tiles can lower the overhead of the slice-header, especially if multiple slices per tile are used. By distributing tiles according to the spatial dependencies between different regions of an image, coding efficiency can be maintained. Nevertheless, the more tiles are used, the more spatial dependencies have to be disbanded and hence, the more the coding efficiency is affected negatively. A balance between optimization of parallel processing and overhead reduction on one hand and disbanding spatial dependencies of the CTUs on the other hand has to be established [107].

3.1.3 Scalability features of H.264 and HEVC

3.1.3.1 SI- and SP-slices

The concept of SI- and SP-slices is described in the following section, with reference to Richardson et al. in [93]. Changing from one video stream to a lower bitrate video stream can be one solution to provide constant video playback in unstable network environments. In H.264, for inter-coding this change of a video stream cannot be performed by simply using the reference-frame of the other stream for prediction. This would evoke artifacts, because images of a video sequence with lower bitrate are technically different to those of a higher bitrate. For the change of video streams, for random access (RA) or streaming scenarios, SI- and SP-slices are introduced in the Extended Profile of H.264. They provide RA to compensate changes in bandwidths. For this purpose, a video sequence can be encoded with different bitrates, to choose the optimal stream with a higher or lower bitrate depending on the current available bandwidth.

SI and SP are used to exchange identical video sequences A and B with different bitrates. There are several SP-slices and a special switching SP-slice on so-called switching points. These switching SPs can be used for the prediction of the following SP-slices of the stream A as well as the following SP-slices of the stream B. Figure 3.5 shows exactly this change of a stream A at a switching point to a stream B. The subsequent subtraction and formation of the residual is performed differently from the usual P-slices after block transformation. SI-slices can also be used as switching slices. For SI-slices the intra-frame coding is performed in 4x4 blocks and they, like ordinary I-frames, require a higher amount of data, [93, pp. 216-218]. Inter-coding and prediction of inter-coded frames using SI- and SP-slices need to be adjusted, since the reference frames, used for prediction have to be available at the right time. The encoding and decoding of a switching-frame AB, has dependencies to stream A and stream B, thus the en- and decoding process has to be adjusted. The exact processing of encoding for SI- and SP-slices and the adjusted MC and prediction is explained by Richardson et al. in [93].

3.1.3.2 Scalable Video Coding (SVC)

The Scalable Video Coding (SVC) is outlined in the following, based on Schwarz et al. in [97] and Boyce et al. in [28]. Two or more non-scalable streams transmitted as simulcast usually can evoke the same functionalities as the scalable profiles of the MPEG codecs. A scalable bitstream can be generated by removing parts of its data, while still creating a valid bitstream, that can be decoded and interpreted correctly. The reconstructed bitstream consists of multiple layers that form a single scalable bitstream. These layers are of different quality or resolution, ergo of different Peak to-Signal-Noite Ratio (PSNR). It is a measure for the relationship between the maximum possible power of a signal and the power of the noise. The base layer (BL) is the one with the lowest bitrate and quality, when the bandwidth is reduced it guarantees flowing playback. The enhancement layer (EL) contains additional data, for higher quality.

The changes in quality are achieved by changing the spatial and temporal resolutions and compressing the bitstream more strongly by e.g. higher quantization. Offering a stream with different resolution and quality as single layers would lead to en- and decoding of a video entirely for each parameter. Transcoding a video more than once increases the

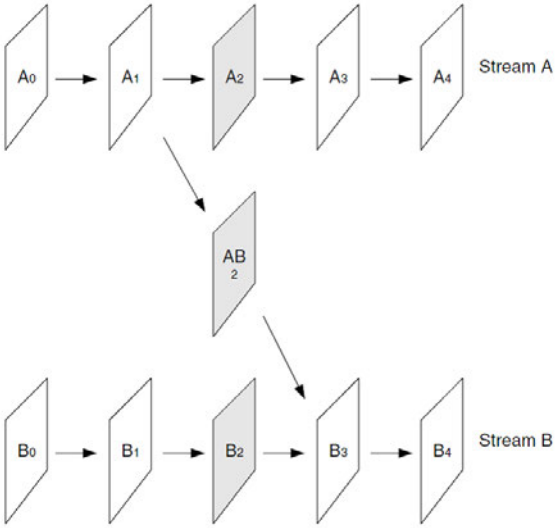


Figure 3.5: Switching a H.264 video stream by using SI- and SP-Slices. Source: [93, p. 217].

en- and decoding time and usually influences the computational power. Encoding the stream as a scalable bitstream with the highest desired quality, provides decoding in many different qualities with different video specifications. This variability and advantages cause an increase in bitrate compared to single layer non-scalable video coding. But especially for transmission channels with high packet-loss rates and unstable network connection the SVC Extension can be very useful. By using so-called Media-Aware Network Elements (MANEs), information about the transmission channel can be received and used to protect the most important data of a bitstream, by skipping the ELs and using the lower layers. For systems with less computational power optionally only the lower layers can be used [97] [28].

An exemplary SVC Encoder creates a scalable bitstream containing a H.264/AVC encoder to maintain compatibility, as shown in Figure 3.6. The MBs of the EL can be predicted from the collocated MB of the BL or by using intra- or inter-prediction with ELs. Using inter-prediction includes the use of motion vectors (MVs) which can also be predicted from the corresponding BL. In a conventional H.264 decoder, the decoding of a MB is looped to prevent errors. As described in [97], SVC decoding uses a single-loop, to reduce complexity and memory requirement. Decoding a bitstream of multiple layers also may provide reduced complexity, since partial decoding of the reference layers in motion compensation may be sufficient. The Reference Layer (RL) MBs that are used for prediction of other layers, can only be intra-predicted by spatially neighbouring MBs. The residuals of the ELs can be predicted by using the inter-prediction residuals of the corresponding RLs [97].

In SVC many extensions to the common H.264 main profile were made. These extensions affect the prediction structures for temporal and spatial scalability, the inter-layer prediction and MC, the usage of key frames and the compatibility to common H.264 bitstreams. SVC can create bitstreams with bitrates less than 10% higher than those of common single layer H.264 bitstreams, but containing a scalable bitstream for individual transmission channels, as explained in [97]. The specific functionalities of SVC are further

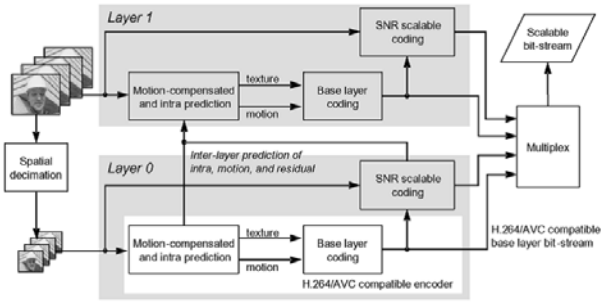


Figure 3.6: An example for a SVC encoder structure. Source: [97].

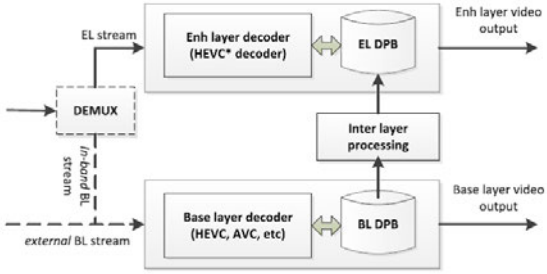


Figure 3.7: SHVC decoder principle with two layers. Source: [28].

explained in [97].

3.1.3.3 Scalable extensions of High Efficiency Video Coding

According to Boyce et al. [28], the scalable extension for HEVC (SHVC) uses the HiLS-only, to create scalable bitstreams. The block structure and coding logic remains the same as in single-layer H.265 coding. In SHVC also a simulcasting coding principle is defined and realized in one bitstream, with so-called independent non-BLs [28]. The inter-layer prediction (ILP) is realized in two different modes. The first one is similar to H.264's SVC ILP, by predicting the current macroblock either by using the corresponding reference layer (RL) or by conventional temporal or spatial prediction in the same layer. A flag at the prediction unit and coding unit level indicates the way of coding the current block [28]. In HEVC a reference list contains a number of possible reference pictures for inter-prediction.

In the second ILP mode, only an index is signalled which indicates the number in a corresponding reference picture list, together with other temporal reference pictures. At the PU level, the indices of the reference pictures are used for reconstruction of the layer to decode. This second reference index approach causes similar coding efficiency, by leading to some new advantages in comparison to the ILP of SVC, whereby it is the main model for SHVC [28].

The decoding principle shown in Figure 3.7 by Boyce et al. [28] is based on the reference index approach with two layers. It outputs a BL and EL and provides backward-compatibility of the BL, to older MPEG codecs. The input bitstream, that has to be demultiplexed first, can contain a BL and an EL, or the BL can be sent as an external BL stream, for example in form of a HEVC, H.264 or MPEG-2 stream. Subsequently

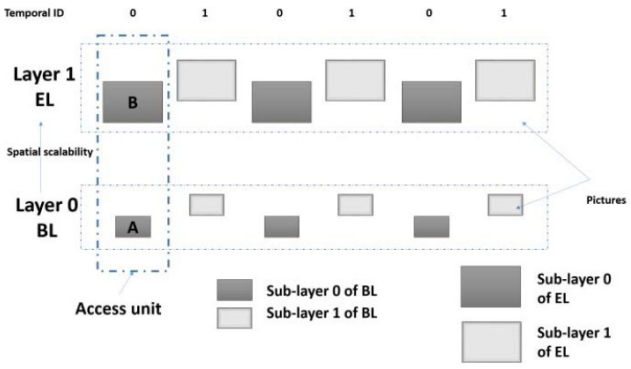


Figure 3.8: Two SHVC spatial-scalable layers with temporal sub-layers. Source: [28].

the EL and BL are allocated to the decoders, respectively. The decoder for the EL has nearly the same architecture as a conventional HEVC decoder, excluding the HiLS-only structure. The interlayer prediction is applied to the reconstructed BLs from the picture buffer and afterwards used for prediction of the ELs, via another picture buffer for the ELs. The discussed decoder in Figure 3.7 [28] contains two layers, but decoding architectures with more layers are also possible, where the base layer (layer 0) can be sent externally and all other layers have to be included in the single SHVC bitstream. Different to SVC, not only single-loop decoding but multi-loop is provided, which allows the use of RLs of other layers, for interlayer processing. This multi-loop design increases the complexity for temporal and spatial scalability. As reported in [28], the advantages of SHVC concern the HEVC single-layer coding architecture, which can be used without adjustments in the SHVC coding structure. This concerns the MB tree structure, the context-adaptive in-loop filtering and the coding logic of HEVC. The coding information, i.e. layer dependencies and interlayer processing types, are made only at the HiLS level. The implementation of SHVC stays simple due to the possibility of adapting parts of HEVC.

Another advantage is the reconstruction of the ELs, which can be performed by using the decoded BLs of the picture buffer and the corresponding motion data [28]. A spatial scalable-layer architecture is shown in Figure 3.8 [28], with temporal sublayers 0 and 1. The exact proceeding of SHVC, with the high level syntax and the processing of the reference pictures is further explained in [28].

3.1.4 AV1 video coding

To push the development of efficient video coding tools on one hand and to be independent of HEVC and other royalty-requiring video codecs on the other hand, the Alliance of Open Media (AOM)[5] launched the open project AOMedia Video Codec 1 (AV1). The AOM is a video technology consortium with 40 members, mainly big hi-tech companies, like Amazon, Apple, ARM, Cisco, Facebook, Google, IBM, Intel Corporation, Microsoft, Mozilla, Netflix and Nvidia, just to name the founders. AOMedia Video Codec 1 is fundamentally based on Googles VP9 codec, which was first launched in 2013, but could not overcome entirely the newest demands of compression-efficiency of the last years.

[5]https://aomedia.org/

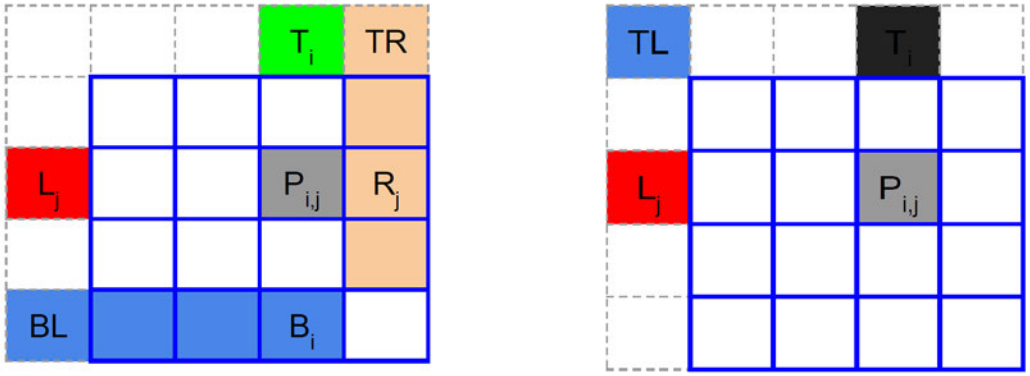
The first version of AV1 was launched in 2018 and achieves bitrate reductions of around 30%, compared to VP9 [34]. AV1 is also based on fundamental established compression techniques, similar to the ones of MPEG image-compression like H.264 or H.265. Thus, block-partitioning, inter- and intra-frame coding with different coding modes for spatial and temporal prediction, transform coding, entropy coding and in-loop filtering are also used in AV1. Furthermore, these classic compression technologies are extended and optimized, in a different way than HEVC does, achieving different PSNR values, i.e. coding efficiency, depending on the referred test and evaluation. With reference to Chen et al. [34], Grange et al. [56], Daede [37] and Trudeau et al. [113], the basic functioning of block-partitioning, intra-coding, inter-coding and other adjustments in AV1 are delineated in the following. The efficiency of AV1 compared to HEVC will be discussed at the end of this section.

3.1.4.1 Block-partitioning

The tree-structured block-partitioning used in VP9 has some similarities to that of HEVC. Referring to Grange et al. [56] MBs of a size of 64x64, so-called super-macroblocks (SB), can be subdivided into smaller sub blocks. At each level of a squared block the decision of subdividing is made using four different options. Three of these options define the block, either as a single square block, or two rectangular blocks, divided horizontally or vertically. The fourth option divides the block into four squared smaller blocks of half of the current block-size. This method is processed recursively, until the end of block-partitioning is reached, whereas the smallest possible division is 4x4. As stated in [37], in AV1 this block-structure is expanded by finer sub-partitioning, allowing partitioning into quarter rectangular blocks and by increasing the SB-size up to 128x128. Albeit quarter rectangular blocks can not further be sub-divided. Moreover, block-partitioning is mode-dependent in the sense, that the minimum or maximum block-size, of 2x2 in some cases of chroma-inter-prediction or 128x128 are not inevitably available at each intra- and inter-coding mode.

3.1.4.2 Intra-Coding

Pursuant to Grange et al. [56] and Mukherjee et al. [86], in VP9 intra-coding 10 different prediction modes with block-sizes up to 32x32 are supported: The `DC-mode`, a so-called true motion prediction mode and the modes depending on the prediction direction of the collocated reference MB, like horizontal-, vertical- or directional modes. The `directional modes` predict the MB, by using adjacent MBs, i.e. their luma/chroma values. In VP9 the subsequent transformation of the MBs is performed by a 4x4 transform and scanned in raster-scan order. The intra-coding in AV1 has been refined and optimized, due to an increase of available directional modes, improvement of the non-directional predictors, exploiting the common properties of luma and chroma signals and sequestrated caring about artificially produced image-content. The generic directional prediction has eight different modes, caused by eight main directions, with angles between 45 to 207 degrees, of VP9. These can be shifted by an offset or delta of a three degree step-size, leading to 56 modes in total. Directional prediction requires estimation of pixel values from neighbouring blocks. The refinement of angles for directional prediction leads to necessary sub-pixel interpolation for the reference pixels, which is done by 2-tap bilinear



(a) (b)

Figure 3.9: (a) Smooth intra-prediction mode in AV1. P as a weighted combination of TR, BL, T and L. (b) Paeth intra-prediction mode in AV1. Source: [83].

filtering [34] [37]. The `smooth` prediction mode is introduced in AV1, using horizontal, vertical or averaged quadratic interpolation of the block to be coded. Starting by edge pixels, i.e. the rightest pixel of the top row and the bottom pixel of the left row, as depicted in Figure 3.9a of [83].

Corresponding to Figure 3.9a, with TR = R, and BL = B, the current pixel P is calculated by a weighted combination of TR, BL, L and T, as described in [83]. In `paeth` intra mode, shown in Figure 3.9b, the values of the top (T), left (L) and top-left (TL) edge reference pixels are regarded and the most matching value is copied [34]. To detect the best matching value, each pixel value is compared to L + T - TL and the values with the minimum gradient are copied [83]. In AV1 a so-called chroma from luma prediction is introduced, where chroma values are defined by incident luma values. The reader is referred to [34] and [113] for a detailed description of this prediction mode. The `color palette` mode is primarily for artificially produced and animated video content with consistent colors and few distortion, for block-sizes down to 8x8. A color palette of two to eight base colors can be used to define pixel values and only information about the palette and indices need to be signalled. Entropy coding is used to reduce the data for the indices and palette-size values. In `Intra_BC mode`, whole blocks can be copied from already predicted areas, which is useful for coding of areas with repeating patterns or regular background areas, using in-loop filtering to reduce blocking-artifacts at the edges of the blocks.

3.1.4.3 Inter-Coding

Compared to VP9, the inter-coding in AV1 has extended reference frames and consists of single prediction modes or paired and averaged prediction of two modes. The prediction modes depend on the distance of the reference frame and its spatio-temporal similarities. Inter-prediction generally can be bi-directional or uni-directional like in the latest MPEG codecs. The exact functioning of inter-coding in AV1 is stated more precisely in [34]. The MV reference selection is based on spatial and temporal similarities, by creating a list with best matching candidates. The spatial neighbourhood for MV referencing is wider than in VP9. In AV1, a so-called overlapped block motion compensation (OBMC) is

used, to reduce prediction errors near edges of coding blocks, by combining predictions from adjacent motion vectors. The warped motion compensation divides the MC into global and local. Local MC is the conventional detection of motion and 2D displacements within the 2D scene, signalled by MVs. Global MC refers to camera movements and movement of the entire scene. According to [34], the compound prediction is a weighted inter-prediction mode, with two references. The weighting factor is determined based on a table with regard to different use cases. Different calculation of the weighting and different combinations of the the predictors lead to multiple compound prediction modes.

3.1.4.4 Transform Coding, Entropy Coding and additional features

Four different transform-types can be used: The DCT and the asymmetric discrete sine transform (ADST) were also used in VP9. Additionally, a so-called identity transform (IDTX) and a reverse-order ADST (flipADST) are used in AV1. In IDTX, transform coding is skipped in a specific direction [34]. Block-sizes between 4x4 and 64x64 are possible and not only squared, but 1:2 and 1:4 rectangular block-sizes are allowed [45]. Albeit, not all combinations of transforms are allowed for all inter- and intra-modes. A multi-symbol arithmetic entropy coding is used for efficient adaptive compression of the symbol-string. Alphabet-sizes up to 16, for binary and multi-symbols are used and the probabilities are stored as 15-bit cumulative-distribution functions (CDF) [34]. Many more features like recursive-filtering for intra-coding, in-loop deblocking filters, grain reconstruction and super-resolution are established. For more information about the functioning of AV1, the reader is referred to Chen et al. [34].

3.1.5 Joint Exploration Test Model 7

ITU-T VCEG (Q6/16) and ISO/IEC MPEG (JTC 1/SC 29/WG 11) are working together in the so-called Joint Video Exploration Team (JVET), to study and develop video coding technologies, exceeding the performance of current video codecs like AV1 and HEVC [59]. The resulting video coding model is called Joint Exploration Test Model 7 (JEM 7) [24]. Studying video coding algortihms is a very complex and broad subject, so that only a short outline of the main changes of JEM, compared to HEVC is given in the following section, based on the descriptions of [24]. In the performance analysis at the end of Section 3.1, all three codecs, AV1, H.265 and JEM are considered. For a detailled describtion of the single coding modules of JEM, the reader is referred to [24]. As JEM is based on HEVCs core coding scheme, enhancements of single technologies in block-partitioning, inter- and intra-coding, transform coding, loop filtering and entropy coding, lead to the performance improvement of JEM. The block-partitioning scheme of HEVC is changed extensively and simplified, by discarding the separation concept of coding units, prediction units and transform units and changing the partitioning scheme from the CTB and CTU structure to a quadtree plus binary tree (QTBT) block strucutre. Hence, the coding unit, i.e. the processing unit of the leaf-node is the only segmentation unit for both, prediction and residue-transformation. The main division is the quadtree structure and finer partitioning of quadtree leaf nodes is done in binary tree order. The shapes of the binary tree blocks can be either squared or rectangular, as they can be vertically or horizontally subdivided as depicted in Figure 3.10. Here, solid lines and broken-lines symbolize quadtree or binary tree division respectively.

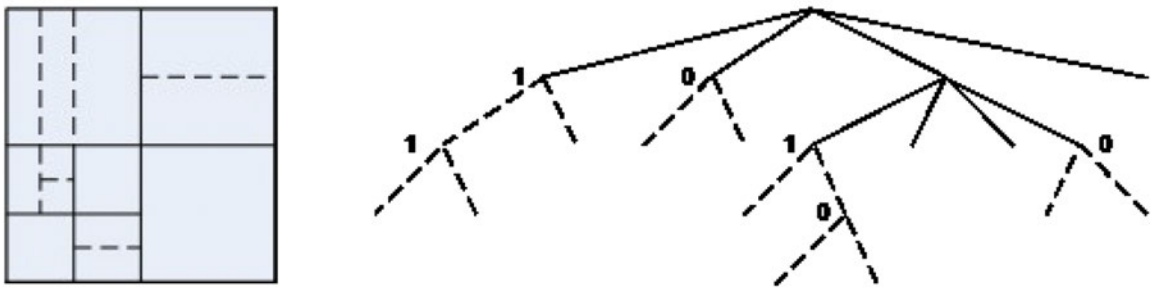


Figure 3.10: QTBT partitioning scheme of JEM, with vertical or horizontal sub-division into CUs. Source: [24].

Moreover, the sizes for the CTUs, as the root unit, can be larger than in HEVC with maximum sizes of 256x256. Furthermore, the partitioning restrictions of HEVC inter-coding into 4x8 and 8x4 blocks for bi-prediction are discarded in JEM. The concept of coding blocks for separated chroma and luma processing is maintained. Intra-frame coding in JEM is extended, compared to HEVC, due to addition of directional intra-prediction modes, from 32 to 65. Defining the corresponding intra-prediction direction is done, as in HEVC by selecting a most probable mode (MPM), with a list of several candidates, that can be used as a reference. Defining the MPM is done by an enhanced derivation. Additionally, entropy coding is used for data-reduction of the MPMs. For interpolation in intra-prediction 4-tap filters are used and enhanced boundary filtering, compared to HEVC is introduced in JEM. As the accuracy of inter-coding mainly depends on precision of the MVs, in JEM the MV-prediction and MV-resolution are amended. Further, an overlapped block motion compensation is used, similar to that of AV1. The used transformations in JEM are adaptive multiple core transform, mode dependent non-separable transform and signal dependent transform (SDT). The exact functioning of these modules and the in-loop filters and CABAC are further described in [24].

3.1.6 Peculiarities on 360° video coding and streaming

As already mentioned, 360° video has some special characteristics differing from those of common planar video, for the single processing steps. The content representation differs from conventional 2D representation. It is a projected two-dimensional representation of spherical content, which results in some special treatment of some coding steps, especially for motion compensation, intra-prediction and inter-prediction. Further, 360° videos should be displayed in very high resolution, principally when using HMDs for playback, to prevent the eye from distinguishing between single pixels. To overcome this extremely high resolution issue, not all parts of the spherical video should be entirely streamed at all time, or at least at weighted qualities. Thus, interaction between the client and the server is required, to establish viewport-dependent streaming.

Conventional 2D planar videos contain objects with predictable regular movements, from one frame to subsequent frames. Motion compensation uses motion-vectors to define the movement of objects inside a scene. Considering that spherical content needs to be mapped and frame-packed into a rectangular representation, like cubemaps etc., the

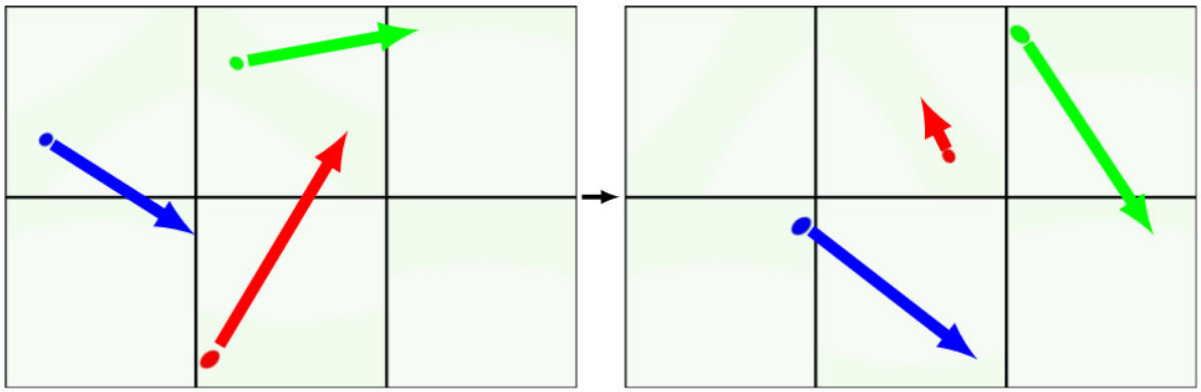


Figure 3.11: Motion discontinuity at the cube side borders. Source: [105].

resulting frame is partitioned into e.g. six pieces, each containing one cube side. In this kind of two-dimensional image representation, objects do not move the classical way. If, for example, in spherical representation an object moves constantly from one point to another, it may move from one cube side to another one. In the resulting frame-packed image, by changing and rotating the cube side, an object would move to a completely different part of an image, abruptly from one frame to another. Figure 3.11 of [105] illustrates this problem. Hence, for frame-packing MC needs to be considered, as unrefined classic motion compensation is less effective in 360° video.

This frame-packing problem also affects intra-prediction. In this case similarities of neighbouring samples within one frame are used for coding. The conventional sample-derivation from left and upward co-located blocks, works less efficiently, especially at the borders of the cubes faces. As a consequence, sample derivation from reference samples geometrically neighboring in the sphere would be more effective, by frame-packing the cube sides according to their similarities and regularities, like the approach of Su et al. in [105]. They introduce a sphere rotation prediction and train a convolutional neural network (CNN) to pack single cube sides into frames, according to their similarities, achieving data reductions of about 8% in average, for H.264 and HEVC [105].

Generally, viewport-dependent coding and ROI prediction lead to higher coding efficiency. By defining a content-dependent viewport and ROI, individually from scene to scene, these parts can be coded in advance at a higher quality, than less important parts of the image. These characteristics also effect the streaming settings, not just because of the huge amount of data, that needs to be transmitted, but also due to the peculiarities of playback and the user's viewing manners. In conventional broadcasting systems, every user receiving an conventional live stream can be served with one single multicast session. In 360° video each user can decide which part part of the video he/she wants to watch, and accordingly he/she can change the viewport during the streaming session [23].

These peculiarities are the thread of this work and take parts in all sections. They might not be resolved all in one solution, but should be considered, especially when it comes to evaluation of a streaming-architecture according to the infrastructure of the IRT.

3.1.7 Performance comparison of AV1, HEVC and JEM

Objective performance analysis of current high-efficient video codecs like AV1, HEVC and JEM is a complex and elaborate scope. Some performance evaluations are not that objective and their results differ enormously, depending on the used codec-implementation and version, the used test-sequences and the used evaluation method like PSNR, SSIM etc., just to name the key aspects. Often, developers of a video codec not only publish the codec-algorithm and/or implementation but also evaluate the proposed codec, compared to others, in the aftermath. Self-interest of the companies can not always be suppressed. Amongst others, the aspects mentioned above explain different results of different evaluations. These circumstances should be considered for the following introduced evaluations. Only results suitable for the 360° use case are considered in detail in this section, so mainly high-resolution low-delay content is considered, and lossless or all-intra configurations are omitted.

Grois, Nguyen and Marpe compare the performance of AV1, HEVC and JEM in [59]. For evaluation the JCT-VC HEVC Test Model HM 16.15 implementation, the AV1 version of August 2017 and JEM 6.0 were compared to each other. Similar coding parameters for all codecs and the HEVC and JVET CTC test sets of [32] were used, including Full-HD, 2K, 4K and 360° video content in equirectangular representation. Here, only classes A, A1-A3 (UHD), B (1080p) and 360 will be considered, to fit the aforementioned use case. The 360° content of 8K-resolution is downsampled to 4k to fit the PSNR encoding comparison. In this case, for fixed quantize-parameter settings, JEM outperforms AV1 with bitrate-savings between 47.7% to 53.9% for the same PSNR values, depending on the test-sequences. For 360° content, a bitrate-reduction of 51.6% can be achieved. The overall averaged bitrate-reduction between JEM and AV1 of these classes is around 50%. In this test-environment HEVC HM 16.15 outperforms AV1 for fixed quantization parameter (QP) settings for classes A, A1-A3 in average by around 22,25% *Bjontegaard Delta* bitrate savings (BD-rate). The same codec comparison for 360° content leads to around 34.8% BD-rate savings. For high-resolution content the bitrate-savings vary between 12.4% to 31.4% for classes A, A1-A3. VP9 is also considered in [59] and outperformed by all other-codecs. The evaluations for the multi-pass rate-control encoding configurations match all in all the listed results for fixed QP-settings. This bitrate reduction usually comes at the expense of significantly higher encoding times. So in average JEM needs 10.51x more time for encoding than the selected HEVC configuration. In contrast, the JEM encoder and the HM encoder are around three and 25-times faster than the AV1 encoder, respectively. It should be noted, that the AV1 implementation used in this test, is of August 2017 and hence still an early state of implementation of AV1.

In [98] Dias et al. evaluate the coding efficiency of AV1, HEVC and JEM using a subjective method - the mean opinion score (MOS), according to ITU-R BT.500-13(8)[114] and PSNR values. The test-set consists of five UHD and five full-HD broadcasting test-sequences of mainly 50 and 60 fps, in 4:2:0, 8-Bit. Except one in 10-Bit and one with 25 fps. The implementations used are HEVC HM 16.10, of summer 2016, AV1 of January 2018 and JEM 7.1 of October 2017. Noticing that the used HM implementation is obviously older than the other two. Nevertheless, with the proposed configurations, HM and AV1 achieve similar coding efficiency, measured by MOS. Generally, a quality gain for higher-bitrates is less visible than for lower bitrates. So a sequence encoded at 12 Mbps might look similar to a sequence of 15 Mbps, whereas the difference between

a sequence of 0.5 Mbps compared to 3.5 Mbps is much more apparent. For MOS and PSNR evaluation in this test, JEM achieves significantly higher coding efficiency than the other two, especially for lower bitrates. At higher bitrates the quality improvement is less clear and in some cases HM and AV1 achieve comparable qualities and in some cases one of both has higher coding efficiency than the other. The bitrates for the same PSNR values of HM and AV1 are quite similar, calculating the overall $PSNR_{yuv}$. In the test only the single PSNR values for Y,U,V are listed, which differ among themselves. For similar PSNR and MOS results HM 16.10 outperforms AV1 with respect to 106x faster encoding times and four times faster decoding than AV1. As JEM outperforms both significantly in terms of bitrate-savings measured by PSNR, the encoding and decoding times are 5x higher compared to HM. The coding times depend on the used configuration, resulting in faster encoding of AV1 when other faster presets are used. This is accompanied by a loss of coding efficiency.

In terms of bitrate-reduction of JEM over AV1 and HEVC Laude et al. come to similar results in [74]. HEVC HM 16.16, JEM 7.0 and an early AV1 implementation[6] were evaluated for intra-only, low-delay B, and random-access configurations, measured by PSNR. For coding of high resolution content with inter-coding configurations, that matches the streaming-scenario, HM achieves considerable bitrate-reductions of around 38.3% for RA-configurations and 34.3% for low-delay configurations over all test-sequences. For UHD content the bitrate overhead of AV1 is less drastic, leading to bitrate savings for HM of 17% for RA-configurations and 14.9% for low-delay-B configurations, in average for classes A1 and A2. The compression efficiency of JEM is crucial, compared to the others, with average bitrate-reductions of 38.7% and 31.1% for classes A1 and A2 in RA-configurations and 25.1% and 30.4% in low-delay-B configurations, respectively over HM and AV1. Due to these results, one could expect higher complexity, leading to higher encoding-times for JEM. In this case, the AV1-encoder needs 32.6x and 21.9x of the HM-encoding-time, for RA and low-delay-B configurations respectively. The decoding-time of both is comparable, and slightly faster for the AV1 encoder. The JEM decoder in this case is the slowest.

Another test, of Chen et al. [34], comparing AV1, VP9 and x265 for mid-res source files, comes to different results. Compared to the x265 implementation of HEVC, bitrate-savings of AV1 of around 22,23% for full-HD content, evaluated by $PSNR_y$ are measured. For chroma these measured savings are even higher. Nevertheless, this test does not fit exactly the proposed use case, as only mid-res sequences were tested and no UHD content or higher has been evaluated.

[6] AV1 Version 0.1.0-5913
[1] Classes A (2K), A1-A3 (UHD)
[2] Fixed-QP settings
[3] 360° content
[4] Rate-Control settings
[5] MOS
[6] BD-rate $PSNR_Y$
[7] RA-configuration
[8] Low-Delay B
[9] Classes A1, A2 (UHD)
[10] Calculated based on the given BD-rate / encoding time ratios of JEM vs HM and HEVC vs AV1
[11] Equirectangular projection
[12] Rotated Sphere projection

Parameter	Grois et al. [59]	Dias et al. [98]	Laude et al. [74]	Chen et al. [34]	Topiwala et al. [112]
Implementation	HM 16.15 AV1 (Aug. 17) JEM 6.0	HM 16.10 AV1 (Jan. 18) JEM 7.1	HM 16.16 AV1 0.1.0 (2016) JEM 7.0	x265 AV1 (Apr. 18)	HM 16.15 x265v.2.4 AV1 0.1.0 (2016) JEM 6.0
Coder Configuration	Fixed-QP, Multipass Rate-Control, RA	Fixed QP, RA	Fixed-QP, RA, Low-Delay B	Fixed QP - Constant Quality (CQ), Multipass Rate-control	CQ 1-pass, target bitrate 2-pass
Test-Set	CTC JVET (2016) CTC HEVC (2013)	5x UHD 5x Full-HD Broadcasting Content	CTC JVET (2017)	AWCY Full-HD and lower-res. content	8k JVET test set (year n.a.)
Year	2017	2018	2018	2018	2017
Evaluation Method	BD-Rate/ PSNR	MOS (ITU-R BT.500-13) and BD-Rate/ PSNR	BD-Rate/ PSNR	BD-Rate/ PSNR	BD-Rate/ spherical PSNR evaluations
Bitrate-savings [%]					
JEM vs HEVC	$-32.15\%^{1,2}$ $-26.0\%^{3,2}$	$-30\%^{5}$ $-32\%^{6}$	$-38.7\%^{7,9}$ $-25.1\%^{8,9}$	n.a.	$-26\%^{6,11}{}_{\text{HM}}$ $-21\%^{6,12}{}_{\text{HM}}$
HEVC vs AV1	$-22.25\%^{1,2}$ $-34.8\%^{2,3}$	$4.0\%^{5}$ $-5.0\%^{6}$	$-17.0\%^{7,9}$ $-14.9\%^{8,9}$	$22.23\%^{6}$	$-15\%^{6,11}{}_{\text{x265}}$ $-12\%^{6,12}{}_{\text{x265}}$
JEM vs AV1	$-50.0\%^{1,2}$ $-51.6\%^{2,3}$	$-35.4\%^{10,6}$	$-31.1\%^{7,9}$ $-30.4\%^{8,9}$	n.a.	$-51\%^{6,11}$ $-51\%^{6,12}$
Encoding-times					
JEM vs HEVC	$10.51x^{2}$	5x	$10.35x^{7,9}$ $8.9x^{8,9}$	n.a.	n.a.
HEVC vs AV1	$0.04x^{2}$ $0.05x^{4}$	0.0094x AV1 106x over HM	$0.024x^{7,9}$ $0.033x^{8,9}$	n.a.	n.a.
JEM vs AV1	$0.37x^{2}$ $0.52x^{4}$	$0.047x^{10}$	$0.25x^{7,9,10}$ $0.29x^{8,9,10}$	n.a.	n.a.

Table 3.1: Overview of particular evaluation results by [59], [98], [74], [34] and [112] for HEVC, JEM and AV1 codecs.

Considering the sought 360° streaming use case, the test of Topiwala et al.[112] is also appropriate. The used test set consist of five 8K 30fps 4:2:0 YUV 360° sequences of 8 to 10 bit-depth, from a JVET test set, not further described. Equirectangular projection and Rotated Sphere projection (RSP)[22] are used for sphere-to-planar mapping. For evaluation of the different codec-implementations, specific spherical PSNR measurements are used, which are stated in [112]. The implementations are HM 16.15 for HEVC, JEM 6.0 and AV1 version 0.1.0 from 2016 for constant quality, 1-pass vbr encoding and x265 2.4 replacing HM for target bitrate 2-pass encoding. Before encoding the test-sequences into AV1, HEVC and JEM, the 8K source clips are downsampled to 4K resolution. Afterwards they are encoded and subsequently up-sampled to 8K, presumably to overcome computational-power issues and run-time problems. The results conform to those of the above mentioned tests, with JEM outperforming AV1 and HEVC. An average bitrate reduction of around 26% of JEM to HM for ERP and 21% for RSP is documented, for all spherical PSNR measurings, of the luma component. The contrast between JEM and AV1 is even higher, as with -51% BD-rate for ERP and RSP for PSNR Y, JEM needs around half of the bitrate than AV1 at a comparable quality-level. Even the x265 implementation at target bitrate encoding outperforms AV1 with around 15% bitrate savings for ERP and around 12% for RSP, for PSNR Y. For exact values of the single PSNR Y,U,V components for each sequence, the reader is referred to the paper of Topiwala et al. [112]. The gap between the presented results of all tests, might also be caused by the different versions, implementations and encoding configurations used. An excerpt of certain results of the referred tests of Grois et al. [59], Seixas Dias et al. [98], Laude et al. [74], Chen et al. [34] and Topiwala et al. [112] is given in table 3.1.

3.2 Video streaming technologies

Constant quality and constant playback are a key for video streaming of an immersive audiovisual experience. As a consequence, the main demands on streaming scenarios and the prior coding, are handling the varying network coverage and transmission quality, saving computing power and hereby energy consumption and simultaneously maintain adequate subjective and objective video quality [78]. Regarding streaming scenarios most important for a constant fluent playback is to control the transfer rate to let the data arrive before it has to be displayed. This section first treats basic HTTP streaming roughly and outlines streaming technologies, especially attractive to 360° video and scalability. Thus, with respect to 360° video, the approach of adaptive streaming using tiles is introduced and approaches of predicting the ROI and FOV are presented. Afterwards MPEG Dynamic Adaptive Streaming over HTTP (DASH) as a common and effective streaming standardization over HTTP is explained in Section 4.1.2. It has been developed by MPEG and published in 2012.

Unlike as in traditional broadcast scenario, for streaming per internet a direct connection between each client and server, hosting the video content is established. For the data transfer several transport protocols are used, to standardize the communication, i.e. requests of the client and responses of the server. It is useful to distinguish between on-demand and live streaming, to select the respective streaming enviroment, i.e. protocols, specifications etc. Popular protocols for on-demand access are HTTP and FTP, whereas protocols for live streaming are RTP, RTCP, RTSP, SIP [81]. Figure 3.12 illus-

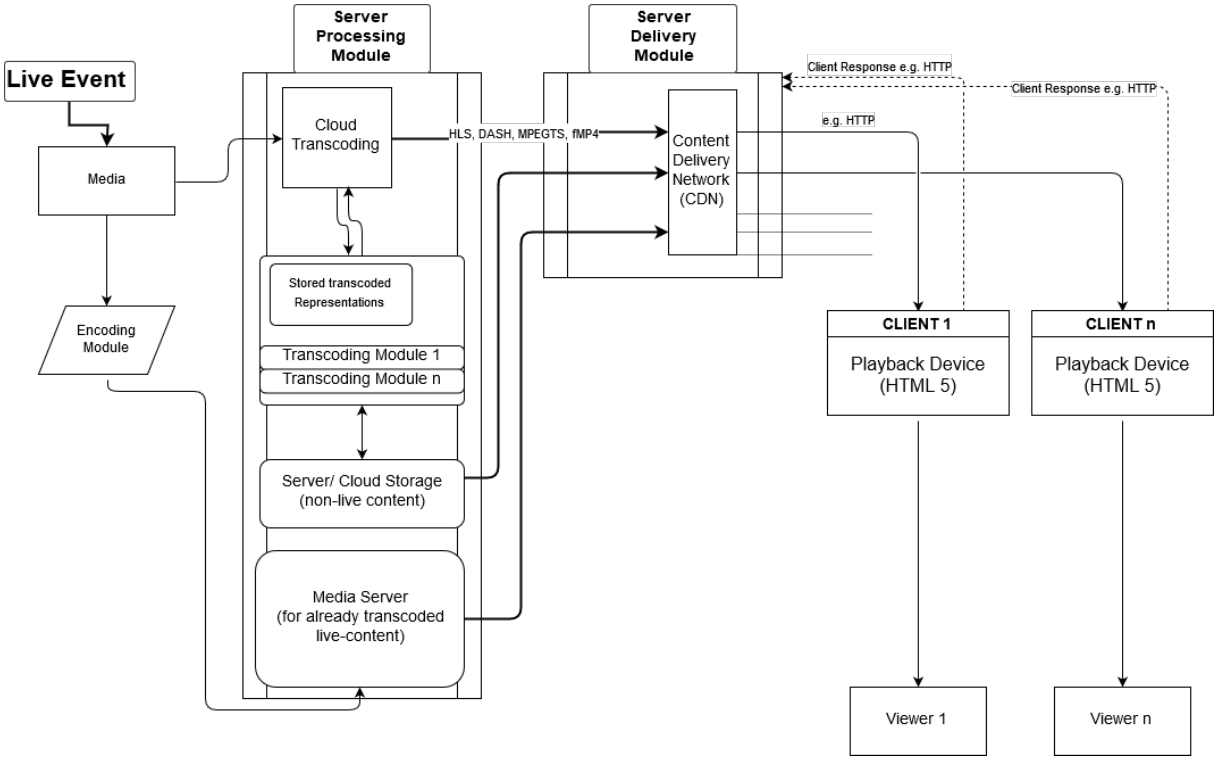


Figure 3.12: Streaming workaround and architecture to categorize the used technologies.

trates the overall principle of a streaming architecture and the relations inbetween the used technologies and protocols.

3.2.1 HTTP and RTSP

Hypertext Transport Protocol

The Hypertext Transport Protocol (HTTP) is the most common protocol, used since 1990 for communication between clients and servers. After a TCP connection is established, the process consists of a request of the client and a corresponding response of the server, containing hypertext links to other files [104]. As described in [44], the nowadays supported HTTP request lines are GET, HEAD, POST, PUT, DELETE, CONNECT, OPTIONS, TRACE, even though only GET, HEAD and POST where established in the HTTP 1.0 version. These are the most important request start-lines to understand how HTTP works and are delineated in this paragraph. According to [43], a basis HTTP request (Version 1.1.) appears as follows:

```
1 ----------------------------
2 Request-line | Start-line
3 Message-header
4
5 [message body]
6 ----------------------------
```

Using the command GET, requests a transfer of a target resource and is the primary mechanism of information retrieval [44]. The command GET is followed by a URL, addressing the server and the corresponding HTTP version. The request to an example

server with HTTP 1.1 using GET could appear like this:

```
------------------------
GET http://www.example.com/public/object.html HTTP/1.1
Host: www.example.com
User Agent: Mozilla/5.0 ...
Accept: text/html...
Accept-Encoding: gzip,deflate,sdch...
Accept-Language: en-US...
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
Cookie: ...
------------------------
```

To request the file directly from a certain origin server the host can be directly addressed. A response message consists of a status line, headers and the body. The status line again is composed by the HTTP version, a status code and the status phrase. An example is depicted below:

```
------------------------
Status-Line
General-Header | Response-Header | Entity-Header

[message body]
------------------------
```

For the status code, various responses like `"200 OK"`, `"100 CONTINUE"` or `"202 ACCEPTED"` are possible, just to name a few. They are divided into five groups with according code-numbers in the brackets: informational (1xx), successful (2xx), redirection (3xx), client error (4xx), server error (5xx). For the above named GET request a typical server response could be:

```
------------------------
HTTP/1.1 200 OK
Server: nginx/2.0.1
Connection: keep-alive
Content-Type: text/html; charset=utf-8
Via: HTTP/1.1 GWA
Date: Wed, 31 Oct 2018 12:16:22 GMT
Expires: Wed, 31 Oct 2018 12:16:22 GMT
Cache-Control: max-age=0, no-cache
Transfer-Encoding: chunked


100
<!doctype html>
(content)
...
------------------------
```

In this case, first the request of the resource is confirmed by `"200 OK"` and header information is delivered beneath. Ensuing, the requested HTML content is transmitted. The request line HEAD is quite similar to GET, but instead of requesting the whole resource, only the header is requested. This can be used for the delivery of metadata of the resource, without transmission of the whole payload. The command line POST can be used for delivery of packages to the server, i.e. block-data, as an input of a HTML

form, messages or mails etc. or as extension of requested resources. The response in this case could be `"201 created"`.

HTTP Pseudo Streaming

This paragraph is based on the descriptions of Longolius in [81]. Accordingly, many providers transmit audiovisual content via HTTP progressive download or HTTP Pseudo streaming. In this case, the data-stream is requested from a server by HTTP, but the content is accessed before finishing the download. The simplicity of the HTTP protocol makes it very easy to establish. To access the data before it has been received entirely, some adjustments of the audio-visual content have to be done. Technically, the header of a video stream is located at the end of the bitstream, with important information for correct decoding. So the header of a video stream has to be put from the end to the beginning of the bitstream, preventing access errors. HTTP pseudo streaming is based on this idea, extending the basic principle by some extra functionalities. It can be considered as a client-based solution, since the server only hosts the video file and no extra protocols or implementations are needed. The client has the software and computational power to decode and display the incoming bitstream correctly, by first downloading the video header and following media parts subsequently. By limiting the bandwith or transferring the file in split parts the provider can reduce waste of traffic resources and client-sided buffers can prevent from unfluent playback, caused by unstable network conditions. To provide random access at every position of the video stream some special treatment is necessary. In this case, the client requests the bitstream at a certain point Y, represented by a certain Byte Y. So first a new modified header, with a new starting point Y of the bitstream is transmitted and subsequently the following parts, so it appears to be a new video stream. Since most video files have a group of picture (GOP) structure, with reference- or key-frames and predicted-frames and dependencies from inter-coding as a consequence, point Y has to be modified, corresponding to the next reference frame.

As described in [81], apart from that, there exist the TCP and UDP protocols i.a. They are designed, to reduce overload and traffic waste, to enable live streaming and to provide quality of service (QoS) (TCP only) and interaction between client and server. In this case, only the desired and requested parts of the video stream are transmitted and adaptation to the available bandwidth is done by changing the video stream. In TCP QoS is included by transmitting packet-recipes, informing about possible package loss, by measuring the time needed to transmit a certain package. A threshold is set by a so-called Retransmission Timer, defining the time to restart the transmission of a certain data-package. UDP has less included functionalities and QoS is not automatically included, but hereby has more possibilities for extensions. As TCP, UDP and other protocols do not play an important role for the implementation of the 360° stream at the end of this work, they will not further be discussed.

Real-Time Streaming Protocol (RTSP)

This paragraph refers to the RFC 7826 documentation of Schulzrinne et al. [96]. Correspondingly, RTSP and its successor RTSP 2.0 have been designed for delivery of data in real-time scenarios, to establish a stateful communication between client and server. Noticing, that the syntax of RTSP 2.0 has slightly changed to its predecessor. Moreover, it is worth to be note, that RTSP 2.0 uses TCP and TLS over TCP obligatorily for all

RTSP messages, indicating, that UDP is no longer supported for RTSP (2.0) message transport. In contrast to HTTP a stateful session is established by a RTSP server after a successful `"SETUP"` request of the client. In `"SETUP"` the requested resource, the streaming type (unicast/ multicast), RTP/AVP and the corresponding port are specified. This initiated session is maintained until a certain time-out by the server or explicit removal via a `"TEARDOWN"` request of the client. Different session-states can be triggered by certain client requests and are maintained by a so called session state machine of the RTSP server. One single or multiple media streams can be associated to one single RTSP session and controlled by the RTSP server. RTSP is bidirectional, providing requests and responses for both sides. RTSP consists of three main steps: The establishment of a session, the media delivery control and the extensibility model [96]. The media delivery is based on RTP, using UDP/ TCP or RTSP connection. The transmission of the stream is controlled by RTCP. Detailed information about RTP and RTCP are stated in [95] and [61] and will not be further expounded. Different to conventional HTTP, during a media transmission positioning and searching within the media content is possible and the client can send requests to start, stop or pause a stream. By extra configurations this is also possible in HTTP, but it is not automatically included. In RTSP, the message lines are either requests or responses. A request consists of a method line, header information and a message body. The method line, identifies the used method, the protocol version and the resource information. The host is delivered as URI, where IP4 and IP6 are included. The message header defines the body length, i.e. the length of the submitting data. A RTSP request is composed as follows:

```
-----------------------------
Request-line
*((general-header / request-header / message-body-header)CRLF)
CRLF
[message body data]
-----------------------------
Request-Line = Method SP Request-URI SP RTSP-Version CRLF
-----------------------------
```

Possible request-lines in RTSP 2.0 are:

```
DESCRIBE, GET_PARAMETER, OPTIONS, PAUSE, PLAY,
PLAY_NOTIFY, REDIRECT, SETUP, SET_PARAMETER, TEARDOWN
```

The reader is referred to [96] for detailed functioning of these requests. An exemplary `"PLAY"` request, after successful establishment of a RTSP session, between client C and media server M is depicted below:

```
-----------------------------
C->M:    PLAY rtsp://media.example.com/media.3gp RTSP/2.0
         CSeq: 835
         Session: yd21lyv
         Range: npt=0
         Seek-Style: RAP
         User-Agnet: PhonyClient/1.2
-----------------------------
```

The response starts correspondingly with a response line, the used protocol and version, the status code and the reason phrase:

```
-------------------------
Status-line
*((general-header / response-header / message-body-header)CRLF)
CRLF
[message body data]
-------------------------
Status-line = RTSP-Version SP Status-Code SP Reason-Phrase CRLF
-------------------------
```

The Status-codes in the RTSP server response resemble to HTTP and are: 1xx informational, 2xx success, 3rr redirection, 4xx client error, 5xx server error. A possible response, containing the necessary information of the RTP packet with the media content, corresponding to the preceding request is depicted below:

```
-------------------------
M->C: RTSP/2.0 200 OK
      CSeq: 4
      Server: PhonyServer/1.0
      Date: 06 Nov 2018 12:32:23 GMT
      Session: yd21lyv
      Range: npt=0-634.10
      Seek-Style: RAP
      RTP-Info: url="rtsp://media.example.com/media.3gp/trackID=4"
          ssrc=0D12F123:seq=12345;rtptime=3450012,
         url="rtsp://media.example.com/media.3gp/trackID=1"
          ssrc=4F312DD8:seq=54321;rtptime=2876889
-------------------------
```

Conventional HTTP pseudo streaming and RTSP differ in terms of use cases and transmission configuration and result in different advantages for different scenarios. Some of them are noted hereafter based on the aforementioned properties. First, the interoperability, compatibility and network configurations differ noticeably. HTTP is a common and extensively used protocol, supported by nearly every common network device, server and client. Less configuration steps and few requests and responses are needed to establish a media transmission. RTSP needs to setup a session with more communication steps for delivery. In case of varying network conditions and transmission errors, HTTP and RTP, used in RTSP for transmission, have different key solutions. In case of package loss, RTSP clients skip expired RTP packages, to maintain signal-reliability and low-delay transmission. On the other hand, that is, parts of the mediafile will not be received and can not be played at all. Users accustomed to stops and re-buffering might prefer watching the video with pauses, instead of omitting whole parts just to maintain low-delay. Nevertheless, this depends on the user's watching behaviour, the use case and the importance of the (live-) event to be displayed. Special streaming architectures based on HTTP are discussed in Section 3.2.2 and 4.1.2, addressing aforementioned problems of HTTP streaming.

3.2.2 HTTP Live Streaming

As already stated, live streaming needs some special treatments, due to particular main demands, differing from those for conventional video on demand (VOD) streaming:

- High distribution-rate, due to provide simultaneous delivery to many clients.

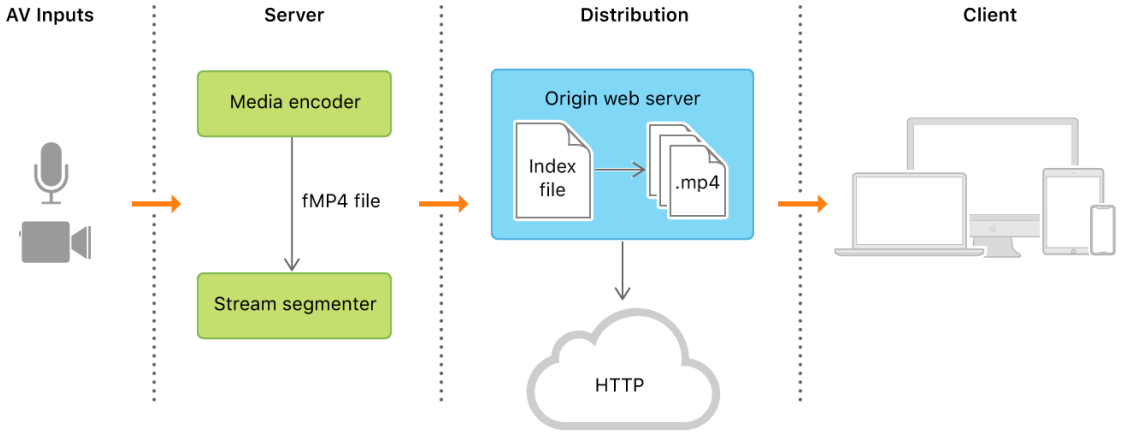


Figure 3.13: HLS streaming architecture. Source: [65].

- Low-delay delivery in terms of recording, live encoding, uploading, server speed and bandwidth.
- Constant-quality: Fluent playback, without drastic changes in perceived quality.

These main demands are not covered by HTTP pseudo streaming imminently. In 2009, the so called HTTP live streaming (HLS) [64] was launched by Apple Inc. defining a streaming architecture, based on the HTTP protocol, transmitting audiovisual content over conventional servers, for VOD and live streaming scenarios. Using the simple and popular HTTP protocol provides compatibility, simplicity of integration and configuration of the servers. According to the corresponding RFC document [92] and the descriptions of Apple in [65] the following specific transmission and processing scheme is used. First, the source material needs to be H.264 or H.265 encoded, hence no other codecs are supported. After encoding, the media representation is segmented into multiple parts, i.e. short media files. An index file list referred to as playlist is created, defining the file-order for re-segmentation. These single split media parts and the corresponding index list are stored on a web server separately. A client, requesting the resource, first retrieves and accesses the index file and subsequently requests the listed media parts in the desired order. During data-retrieval and playback, the segmented single media parts are merged to avoid breaks. This scheme is depicted in Figure 3.13 of [65].

In detail, the h.264/h.265 encoding or transcoding and segmentation of the audiovisual resource is made by the server module. For live-sessions, a live encoder is required, providing real-time low-delay encoding and outputting a MPEG-2 transport stream (TS) or a fragmented MPEG-4 file (fMP4). For segmentation, apple suggests the use of its own media stream segmenter, creating the single media parts of equal length as single MPEG-2 transport streams, i.e. `.ts` files or fragmented MPEG-4 files and the related index list, in `.M3U8` format. On client side, the segmented media parts need to be re-assembled and merged using the index file, where the ordering and location of the media parts, the decryption keys and any alternating available representations are listed. The `"EXT-X-ENDLIST"` tag in the index file defines the end of the media stream. That is, for a running live-stream without expiration, this tag will not appear. To reduce storage resources and bandwidth requirements, the index files can be compressed by zipping. After a client request they are automatically unzipped. Further, it is possible to use

stream alternates, to switch the media representation quality dynamically, for the change of bandwidth or different playback devices, as stated in [67]. In this case, a master index file is downloaded once, pointing to the alternating index files. For live streaming the alternate index files are updated periodically and the client chooses an adequate alternate quality representation, depending on its network conditions. In case of delivery errors, due to server crashes etc. these alternate streams can also be used as redundant streams. This is depicted in Figure 3.14 of [67]. In this case same media representations are located at different server locations and listed separately in the master index file. An exemplary representation of a conventional index file of Apple in [66], with three un-encrypted media parts of 10 seconds length is illustrated hereafter:

```
-----------------------------
#EXT-X-VERSION:3
#EXTM3U8
#EXT-X-TARGETDURATION:10
#EXT-X-MEDIA-SEQUENCE:1

# Old-style integer duration; avoid for newer clients.

#EXTINF:10,
http://media.example.com/segment0.ts

# New-style floating-point duration; use for modern clients.

#EXTINF:10.0,
http://media.example.com/segment1.ts

#EXTINF:9.5,
http://media.example.com/segment2.ts

#EXT-X-ENDLIST
-----------------------------
```

3.2.3 Caching - Edge and origin server principle

Usually, a video stream is addressed not only to a small group of users, but to many users at the same time. Depending on the specifications of a single streaming server, a limited number of users can be supplied. To overcome a server and bandwidth overload, multicast single servers access the requested media data from a certain single storage module. The servers storing the content can be called origin servers and the interposed caching servers can be called edge servers. In this way, when many users request the same content, the edge server only needs to access the content once and can divide it accordingly. Less demanded video streams can be deleted to maintain storage space and computational power for frequently requested streams [81]. This principle is depicted in Figure 3.15.

According to [81], in this case, first the (live-)content is transcoded and transmitted to an origin server, which is not accessible directly by a client. After the request of a client, a load balancer selects a suitable edge server, to balance the payload on each server homogeneously, optimally regarding the servers resources and the clients requirements. The more client requests of different content, the more requests and responses between origin and edge server and hence the more complex the distribution.

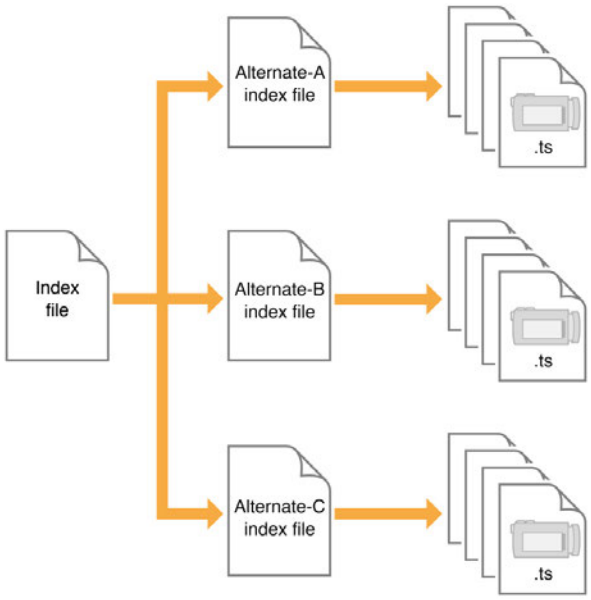


Figure 3.14: Structure of stream alternates, with a master index-file and index files of the corresponding media alternates. Source: [67].

3.2.4 Multicast and unicast streaming

A unicast transmission is a widely used and supported way to transfer media content from the server separately to each user. It is easy to establish and supported by all kinds of networks, using the TCP transport protocol [41]. Nevertheless, unicast streaming has its limits when many users request the same content simultaneously, due to missing scalability options.

Multicast transmissions aim to reduce a potential server overload, by sending a single packet to a group of users, from one or more distribution points and with one particular IP adress [25]. In this case, the administration of delivery is done by the network routers instead of the streaming servers [25]. Hereby, up-scaling of content distribution is more effective, at the cost of less adaptivity to the individual bandwidth requirements of the users. It is worth to be note, that mulitcast usually uses UDP protocols, due to more implementable options and extensions [41]. Compared to unicast, bandwidth savings are reduced, so multicast requires 1/N of bandwidth, with N: separate unicast clients [41].

3.2.5 Cloud transcoding

The term cloud transcoding is commonly used in relation with video streaming. Albeit, it only defines that transcoding of audiovisual content is not exclusively done before distribution by a hard- or software (live-)encoder, but at a transcoding server module. Most cloud transcoding services provide an all-in-one solution for processing and distribution of the media files. Consequently, a cloud transcoding service consists out of several modules for different processing steps, depending on the used service. These can be generally divided into video processing and distribution, i.e. transcoding server and streaming server. Video-processing includes the transcoding into multiple different video formats. The distribution is conventionally done by a CDN, suitable for different users and devices. Cloud

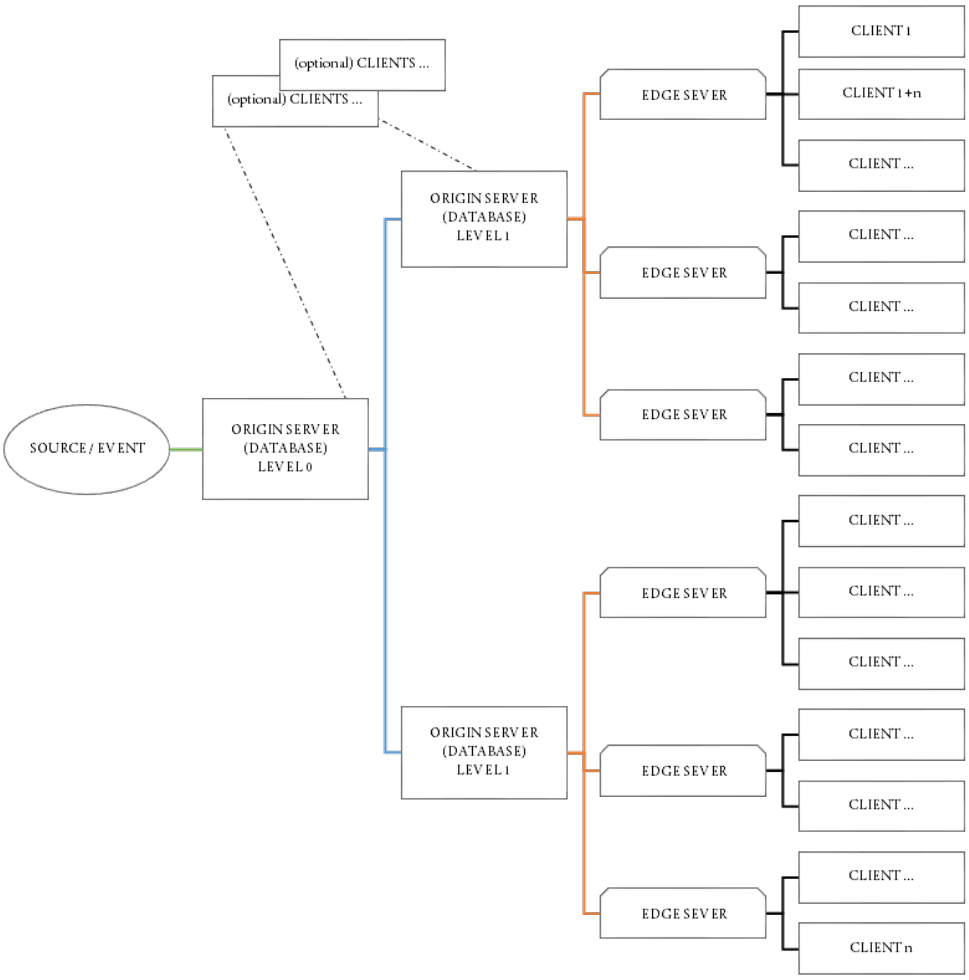


Figure 3.15: Principle of the origin-edge server architecture, scalable by different optional origin server Levels, based on [81].

transcoding is used especially by customers with varying capacity demands. Capacities in the cloud are easily scalable, so additional transcoding engines can be enclosed for simultaneous transcoding of several projects and media files. This procedure can save encoding time and costs and provide a wide range of available formats and distribution ways for the content provider. The cloud transcoding service provider, on the other hand, can manage the coding jobs for all of their customers according to all resulting capacity-demands. Examples for these cloud transcoding services are the Amazon Elastic Transcoder[7] or the solutions of Akamai[8] or Google[9] just to name a few.

3.2.6 Viewport-adaptivity and tiles

This section gives an overview of the principle of viewport-adaptive tiled streaming. The latter Section 4 gives a more detailed insight into the scope of possible combined viewport- and bandwidth-adaptive tiled streaming approaches. This section outlines the overall principle of viewport-adaptivity and describes the related modules. To overcome huge bandwidth-consumption and equally-weighted quality, the concept of viewport-adaptive systems is a powerful solution. The main idea is to transmit the entire 360° content to the client, but with differently weighted parts of quality. Completely omitting parts, less probably chosen by the viewer leads to black pauses and re-buffering, causing disagreeable impairments for an immersive 360° experience. Hence, streaming the entire content but with regions of different quality and adjusting the quality of the corresponding region, when the user shifts the FOV, is preferable. This process should not produce a high delay, i.e. higher than around 10ms [36], to not impair the viewing experience. For this purpose, the concept of tiles can be used, which was broached earlier in this work. In the context of content delivery, tiling means delivery of content in separated parts or tiles, each of which representing a small region of one single video frame. It should be noted, that a higher number of tiles can increase the storage resources of the server, but smaller tiles can map the viewport more accurately.

Combining those two concepts, enables the transmission of tiles covering the viewport at a higher quality, i.e. encoded at higher bitrate and/or higher resolution, whereas all other tiles are streamed at lower qualities. This requires a constant exchange of information about the change of viewport and a server system capable to switch the stream with low-delay. Due to this low-delay use case, each single tile of each quality level needs to be available at the server. The following example illustrates this concept of creating tiles in different quality representations.

For instance, regarding a video sequence of 7680x3840 resolution can be split into e.g. 8 squared tiles, each of which of 1920x1920 pixels. This equirectangular 360° video lasts 12 seconds and is HEVC encoded at 8K60fps at 50 Mbps. The resulting number of frames is 720. The selected number of tiles is 8, leading to 5760 tiles for one single quality level. The simplest viewport adaptivity would be to stream all viewport tiles at best quality and all others at one lower quality, i.e. two quality levels. The best quality in this case would be e.g. 8K tiles at 8 Mbps, the lower quality could be of lower resolution or bitrate, accordingly 4K at 1 Mbps. Consequently the resulting sum of tiles, for our

[7] https://aws.amazon.com/de/elastictranscoder/
[8] https://www.akamai.com/de/de/resources/video-transcoding.jsp
[9] https://cloud.google.com/solutions/media-entertainment/use-cases/live-streaming/

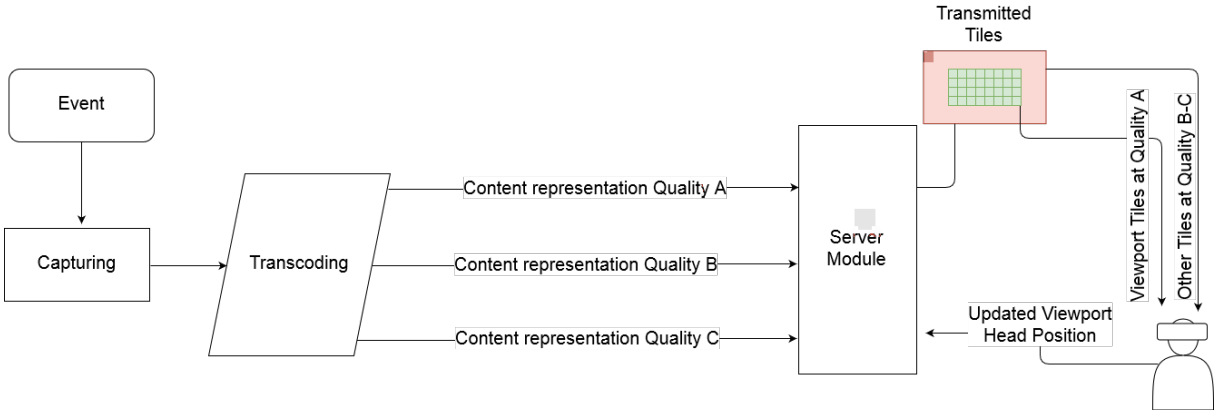


Figure 3.16: Viewport-adaptive tiled content distribution.

12s 360° video would be 11520. Moreover, these tiles need to be packed, e.g. as tile-tracks, to be available for the client. The exact principle of creating tiles, tile-tracks and a segmentation structure for distribution is explained in Section 4.1.5.

This example illustrates, that creating an adaptive architecture, with tiles at multiple quality levels is a complex task, especially for live streaming. When the viewer changes the viewport, the corresponding client needs to send a request for the updated tiles to the server. The server module changes the tile-representations accordingly, for a sequence of time s. This system as a much higher server storage consumption than a conventional streaming system, increasing with the number of quality levels. Simultaneously, this adaptivity should not only address viewport changes, but also network conditions. Combining it to the bandwidth adaptive DASH-architecture is possible, which is discussed in Section 4. Moreover, the concept of tiling depends on the 2D-mapping. For an equirectangular 2D-representation, tiling can be easily conducted, as all neighbouring tiles are neighbouring regions of content. For instance, a frame-packed cubemap might better be tiled along the cube side edges. Figure 3.16 depicts the scheme of viewport adaptive tiled streaming systems.

3.2.7 ROI and FOV prediction

Adaptive tiled streaming can be one solution to overcome huge data-rates of even high-efficiently encoded 360° videos. Thus, the common process is comprised by action of the viewer and reaction of the server, i.e. delivering module. In Figure 3.17 an interaction chain of common 360° streaming applications is illustrated. In most cases the whole 360° content is delivered, although with variable quality regions and viewport adaptive at best. The user changes his viewport to neighbouring parts, usually when another region attracts his/her interest. As a consequence, another part than the current viewport has to be displayed. Thus, initially only parts with lower quality can be displayed. After re-sending information of this viewport change from the client to the server, the FOV is updated. So after all, by changing the bitrate of the requested new tile, the quality is re-enhanced. This action of the viewer, resulting in a request to the server and subsequently transmitting new parts of higher quality to the client, lasts a few perceptible moments, depending on bandwidth and computational power of the server and CDN module. This change of stream is perceptible, so the viewer first watches a sequence of low quality

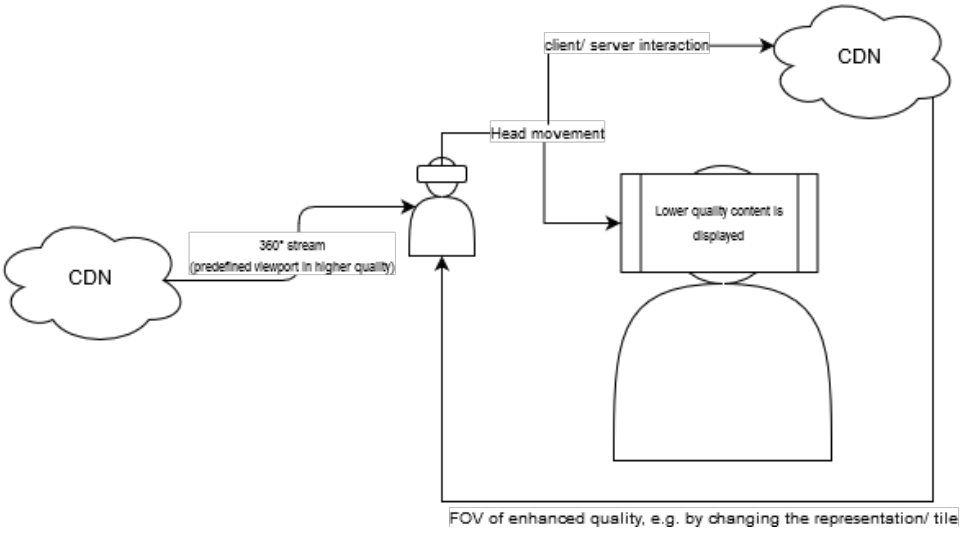


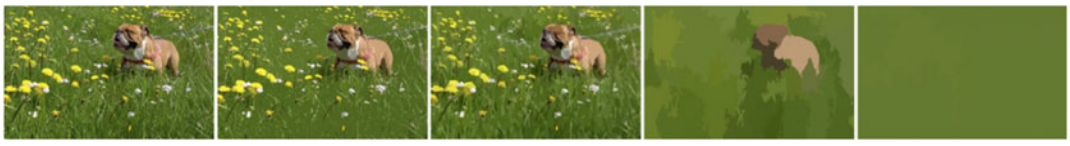
Figure 3.17: Interaction chain for FOV-updating, causing delay and bitrate-overhead.

which changes abruptly to high quality and disturbs the immersion. Moreover, an abrupt change of about 180° of the viewport is unusual. Consequently, transferring all content at all time at homogeneous quality, wastes bandwidth resources and lowers the overall quality.

All those approaches of adaptivity and scalability depend on the question of the viewport: When does the viewer move his/her head in what direction? And ergo: Which area of the 360° video should be transmitted at which time, to display it exactly when it is needed? Predicting the FOV, i.e. the area, where the user will turn to most probably and transferring only that parts, the user will fixate most likely, could avoid sudden changes in quality. By this more bandwidth for fever parts of the whole spherical video content can be provided, resulting in an increase of perceived quality.

3.2.7.1 Image saliency maps

One way to define that ROI is by creating so-called image saliency maps (ISM). There has been a lot of research concerning the classification of image regions, sensitive for the HVS. One initial model for saliency detection was proposed by Itti, Koch and Neibur in [68]. An overview of different methods for image saliency is given by Borji et al. in [27]. In the following section the approach of an advanced method by Shi, Yan, Xu and Jia [99] for creation of image saliency maps is introduced and roughly delineated, based on their corresponding paper. ISMs classify reagions of images according to their saliency or importance for a human viewer. Usually, defining the image saliency encompasses eye-fixation habits and object detection and estimation. As aforementioned, segmenting these regions for a single picture, according to their saliency, can be of considerable interest for creation, processing and transmission of video content, especially 360° video. Generally, object detection and image analysis can be done, by image transformations and pixel comparison, according to their luma- and chroma-values. Common problems in this scope occur, when areas with less interest for the viewer are classified as high salient or the opposite way, when important regions with homogeneous content are classified as background. In [99], for construction of the saliency map, three layers based on the original



(a)

(b)

Figure 3.18: a) Original image (left), Resulting over-segmented layer (second left) and three layers of regions with merged segments by different thresholds. b) Original image (left), three layers of saliency cue maps (right-sided) with resulting final saliency map (rightest). Source: Shi, Yan, Xu, Jia [99].

image are computed. These three layers are of different detail levels, but all of lower detail than the original. The base-layer is calculated based on an over-segmentation of the source image. This over-segmented layer, is generated by the watershed transformation (WST). Based on this, for each segmented region of the image, a scale-value is calculated and applied. Regions below a certain threshold are merged to the next neighbouring segment, neighbouring in the sense of CIELUV color distance. This first base layer is the one with the highest detail level, i.e. number of segments and lowest threshold. The mid and high layers are calculated the same way, only with higher thresholds, to reduce the number of segments. Figure 3.18a shows the WST-generated layer and the resulting three layers of different granularity. Additionally, the regions are analysed in terms of color, size and position. As the HVS is more sensitive to high-contrast edges and objects allocated to the center of an image, saliency cue maps are generated. The local contrast and objects positions and size are used as weighting factors. Based on those three resulting saliency cue maps a final saliency map is calculated, considering refined hierarchical inference, as depicted in Figure 3.18b.

3.2.7.2 Neural network based FOV Prediction

Creating image saliency maps is an important base for the prediction of the user's viewport. Since at this time FOV prediction is a very recent and emerging technology yet, only the approach of Fan et al. [42] is considered. It is picked as an example, to show possible enhancements by using neuronal networks for FOV prediction. With reference to [42], in the following, the functioning of this approach is briefly described.

They use two neural networks, combining content-related information of the image and sensor-related information of the user's behaviour. Content-related information include motion detection and image saliency maps, whereas sensor-related information include the viewers watching behaviour, like head orientation. As one feature, image saliency is ascertained by a CNN, designed for discovery of objects and content categorization. This network outputs an image saliency map for each video frame. Additionally, a motion

detector module, outputs motion maps for the single video frames. These information plus HMD sensor data, as the user's head orientation, are stored in a buffer module. This buffer module stores these features, like a sliding window by a particular number of previous frames. The key module is a recurrent neural network (RNN), responsible for the fixation prediction or FOV prediction, with a successive tiles rate selector. Based on the above mentioned feature data, the fixation prediction network outputs a single probability for all single tiles of a respective future spherical video frame. Predicated on this, the tiles rate selector defines the output data-rate of the single tiles for a frame. Moreover, two different prediction networks are introduced, one working as described above. The other is working similar, except using already watched tiles, in place of orientation-behaviour of the viewer, as input data. The RNN is trained and validated by self-generated user data, of 25 subjects watching different spherical videos, in 4K resolution, at 30fps.

As stated in the work of Fan et al. [42], the neural network using orientation-data comes to more accurate tile probability. Moreover, compared to other solutions of adaptive hybrid streaming systems, the proposed FOV prediction reduces the initial buffering time and the consumed bandwidth, at similar quality levels, measured by PSNR and SSIM, among others. The three other compared solutions use: a) the current orientation for viewport-adaption, b) the velocity of the orientation for viewport-prediction, c) tiles only at a certain average saliency-level. Compared to those, as a result, an average buffering time reduction up to 43% and bandwidth reductions between 22-36% can be achieved [42].

Chapter 4

Related 360° video streaming systems

4.1 MPEG Standardizations and 360° Video Streaming Technologies

In the scope of 360° video, many technological approaches have been proposed, offering solutions for the enhancement of omnidirectional video processing, transmission and playback. Especially the optimization of 360° video applications has been subject to immense research effort. As MPEG proposed standardizations for nearly every level of audiovisual data processing, e.g. video coding standards, such as MPEG-4 and MPEG-H, encapsulation standards, such as ISOBMFF and streaming system standards like DASH or MMT, the standardization of an omnidirectional media format is a logical step.

For this reason, the following section concentrates on common encapsulation and streaming technologies, standardized by MPEG, which build the fundament for the standardization of an Omnidirectional Media Format (OMAF). Accordingly, the file encapsulation via ISO Base Media File Format (ISOBMFF) is explained first. Then MPEG's HTTP streaming standards MPEG-DASH, unifying scalable HTTP streaming approaches like HLS etc. and MPEG Media Transport (MMT), replacing MPEG-TS are introduced.

4.1.1 ISO Base Media File Format

The ISO Base Media File Format, specified in ISO/IEC 14496-12 [50], is a standardized format for the access of media files, developed by MPEG. It contains timed media information, i.e. audio-visual content and extra information. It is based on the MP4 container format, which is predicated on the Quicktime MOV container format. The MP4 file format was generalized into the ISO Base Media File format and hence ISOBMFF is a superclass or basis for more specific container formats such as MP4 or 3GP, just to name a few. The MP4 container standardization was first published as ISO/IEC 14496-12:2004 in 2004 and has been updated continuously, latest in 2015. ISOBMFF is independent of particular network protocols and is hereby extendable and flexible for interoperability. The access can be per network or locally. According to [101], ISOBMFF is object oriented, specifying the content as a series of boxes, to enable segmented access of media

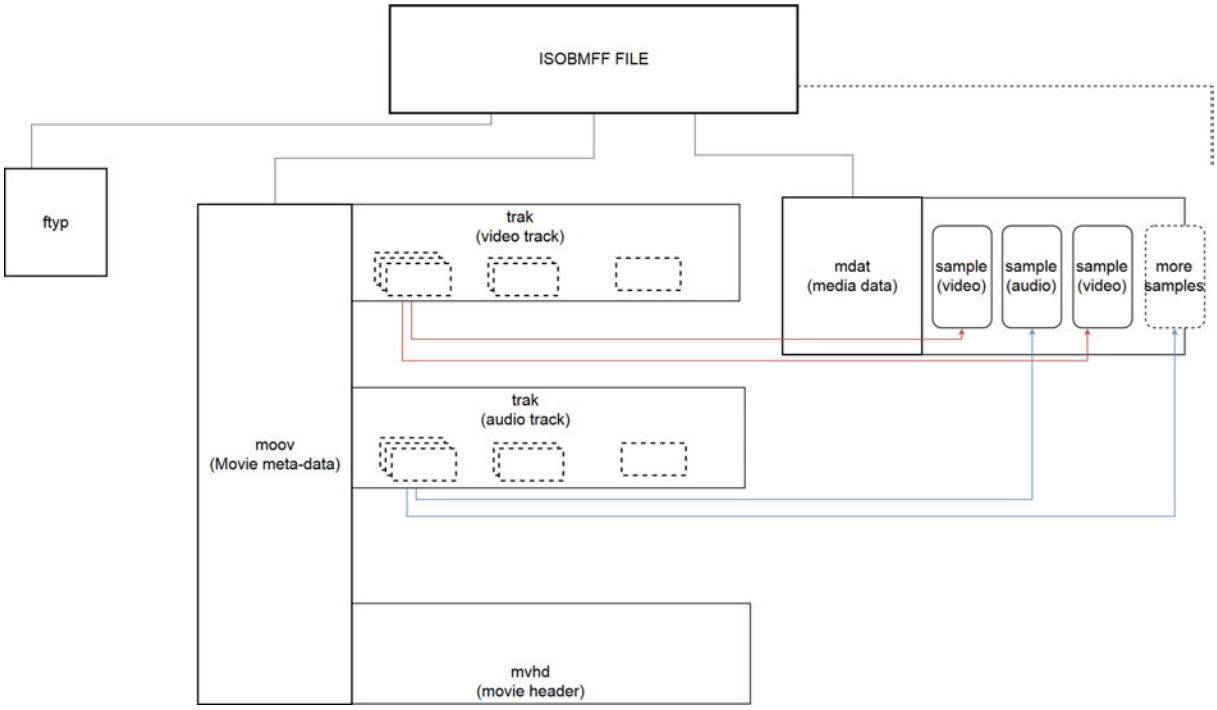


Figure 4.1: Logical and hierarchical physical structure of ISOBMFF.

data. Each box type is defined by 4 printable chairs and a certain length amongst others. The ISOBMFF file contains descriptive information in the boxes `'ftyp'` and `'moov'`.

The overall hierarchical structure of ISOBMFF boxes is depcited in Figure 4.1, where the dotted boxes indicate the sub-structure of boxes below the `'trak'` box level. Moreover, `'ftyp'` specifies the file type and version, whereas the `'moov'` box contains the metadata of the respective media. The media stream, i.e. the respective coded audio or video data stream can be accessed by the track, i.e. `'trak'` box, of an according type, like audio, video, text, etc. The media data itself is packed into the `'mdat'` or `'idat'` box and can be accessed sample-wise by reference of the respective track. The sample entry specifies the media stream, as it contains the name of the exact media type, to define how a decoder can access the media correctly. When fragmented fMP4 files are used, the `'moov'` box only contains basic information about the tracks, defining the quantity, type, codec and initialization etc. Information about the sample locations and sample size in the track are stored separately in the `'moof'` box. A `'mdat'` box is following after each `'moof'` box, containing the samples as defined in the preceding `'moof'` box. The length of a `moof-mdat`-pair is not restricted but typically may be between 2-10 seconds, depending on the use case. The signalling of ISOBMFF in MPEG-OMAF and the packaging-structure between MPEG-DASH and ISOBMFF will be explained in Section 4.1.5.

The Common Media Application Format (CMAF) is a MPEG standardization for encoding and packaging of segmented media. It can be used by many popular streaming technologies, such as MPEG-DASH or HLS. CMAF defines a logical structure for several media objects, such as CMAF tracks, derived from ISOBMFF, CMAF switching sets, CMAF presentation, CMAF Header, Segment, Chunk and Track File. For detailed information about CMAF, the reader is referred to [100] and [63].

4.1.2 MPEG-DASH

Streaming over HTTP is a very simple and rife way of audiovisual content distribution. As RTP and RTSP are not supported by all CDNs and an individual streaming session for each client has to be established and maintained by the server, increasing the distribution effort, optimized HTTP streaming can tackle these problems and comes along with some advantages, like interoperability, compatibility and preservation of server resources, just to name a few. HTTP streaming architectures like HLS[92], Adobe's HTTP Dynamic Streaming (HDS) or the approach of Microsoft's Smooth Streaming, all use similar and effective HTTP streaming, but un-unified and un-standardized. For this reason, MPEG introduced the Dynamic Adaptive Streaming over HTTP, which was released as ISO standard ISO/IEC 23009-1 [47] in 2012. Today, MPEG-DASH is widely supported by many manufacturers on client side, for instance by popular internet-browsers like Chrome, Safari, Firefox, Android's ExoPlayer etc., and devices like Chromecast, FireTV and many Smart-TVs. The functional schematic of MPEG-DASH, based on the revised standardization ISO/IEC 23009-1:2014(E) [49] is described in the following.

MPEG-DASH is a bandwidth-adaptive streaming solution, providing efficient quality management, robust for varying network conditions. DASH specifies XML and binary formats, meanwhile exploiting the advantages of the HTTP protocol. When the client desires a certain resource, it requests a so-called Media Presentation Description (MPD), via GET, containing metadata and additional information about the media. The media content is transferred in Segments, each of wich containing parts of coded media data, including the allocated metadata. The main idea of DASH is, to provide split Segments of media content, at different quality levels. At client side, the merged Segments create a continuous stream. Figure 4.2 illustrates the hierarchical composition and relations between the single elements: MPD, Period, Adaptation Set, Representation, Segment and Sub-Segment.

The superordinate instance is the MPD, in XML format, containing information about the resource, like types of Representations, timing information, information of the segmentation and HTTP-URLs, referring to the server location of the single Segments. That implies, in live-scenarios the MPD needs to be updated regularly. The MPD can be structured into content information, Period information and information about the adaptivity and data-location. The metadata on content- and Period level contain the description, time and duration respective to the content or Period. The adaptation information describes the Adaptation Set, considering the network-conditions and contains the attributes of the Representations. With the provided information, such as bit rate, resolution or frame-rate the client can request the appropriate alternative, i.e. Segment. The metadata can be split into certain parts, to reduce processing complexity and data flow, so the metadata of content only needs to be sent once and not periodically, divergent to Period level information [111].

In the second amendment of MPEG-DASH [87] the so-called Spatial Relationship Description (SRD) is introduced. The SRD feature extends the MPD and contains spatial relationship information of single parts of one video. Hereby, the DASH client is able to request only video streams at the respective relevant resolutions and spatial position. it is used in some of the approaches at the end of this chapter and in the system implementation at the end of this work, to describe the single positions of the individual tile-tracks in the resulting spherical video stream.

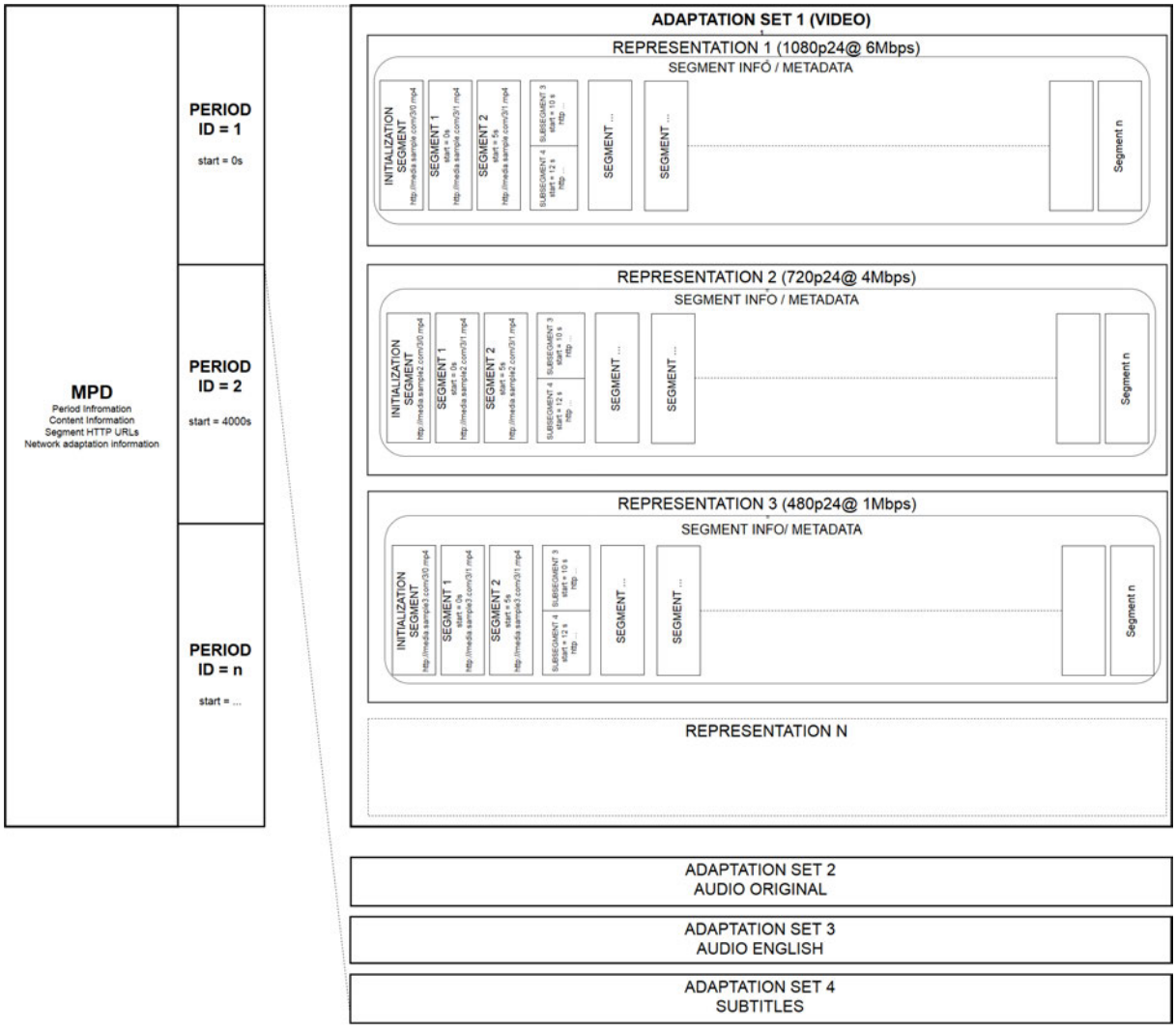


Figure 4.2: Structuring and segmentation scheme of MPEG-DASH

The Period element defines the temporal interval for the whole media-presentation. Additional information like languages, subtitles etc. can be uniformly defined for a whole Period. For each Period one or more Adaptation Sets are defined, containing multiple Representations of identical content at different quality levels. Identical content in terms of viewpoint, language, aspect-ratio, but at different output bitrates and resolutions. Splitting of audio and video can be done at Adaptation Set level, although multiplexed bit-streams are possible. The client accesses the content within one Adaptation Set, capable to change the corresponding media representation, according to network conditions and other aspects. This change of stream, i.e. request of Segments at different quality level, should not be visible in terms of pauses or bucking, thus, only an increase or decrease of quality might be noticeable.

The segmentation is done for each Representation, resulting in multiple Segments of multiple qualities. Each single HTTP request of the client refers to a Segment, or Sub-Segment, representing an even finer division of Segments. Hence, a Segment is the largest possible media file in MPEG-DASH. Since the number and size of Segments is not predetermined, a single Segment can also cover a whole Representation. However, the division of the Segments and Sub-Segments should match the coding properties, like the GOP structure in MPEG. Typically Segments have roughly the same size. Besides the content body itself, the Segments contain time and synchronization information for accurate merging onto a timeline. Additionally, timing information notifies about the accessibility and validity, which is especially important for live-streaming scenarios.

Switching of streams is generally possible at any point or time of a certain Period. Different solutions to accurately switch the quality level can be used. So-called stream access points (SAPs), define fixed positions in the representation to switch a Segment. Different types of SAPs are predefined and described in [49]. Coding dependencies at SAPs need to be broken up, so the GOP structure needs to be considered or rather be used, i.e. the beginning of a closed GOP might be an appropriate SAP. Moreover, overlapping Segments can be useful to switch a quality Representation, but simultaneously evoke a bitrate overhead, as content is redundantly present in several Segments .

To illustrate the specific functioning of MPEG-DASH, the sample DASH dataset of Bacher et al. [76] and Lederer et al. [77] is used and described in the following. In this case, the video source, a high-bitrate 1080p AVC file, is encoded into 20 quality representations, from 0.047 Mbps to 4.51 Mbps. The encoded sequences are then split into six Periods of different length, each of which containing a set of Segments for each quality representation. The MPD informs about the structure of the DASH stream and points to the respective Segment locations, via HTTP-URL, as shown in the exemplary MPD-XML, Figure 4.3. The `SegmentTemplate media` and `initialization` elements, define the server base location of each Segment and the initialization Segment respectively, whereas the single files can be accessed by replacing the `'$'` element according to the naming structure. After requesting the MPD file, the client accesses the initialization .mp4 file and subsequently the single Segments, e.g. as .m4s files (exampleMedia_2s1.m4s), according to the network conditions. In this case, only H.264/AVC encoded streams are available, defined by the *codec* element in the XML.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- MPD file Generated with GPAC version 0.5.1-DEV-rev5379  on 2014-09-09T13:02:25Z
-->
<MPD xmlns="urn:mpeg:dash:schema:mpd:2011" minBufferTime="PT1.500000S" type="static"
mediaPresentationDuration="PT0H10M53.79S" profiles="
urn:mpeg:dash:profile:isoff-live:2011">
  <ProgramInformation moreInformationURL="http://gpac.sourceforge.net">
    <Title>dashed/ElephantsDream_2s_simple_2014_05_09.mpd generated by GPAC</Title>
  </ProgramInformation>
  <Period duration="PT0H10M53.79S">
    <AdaptationSet segmentAlignment="true" group="1" maxWidth="480" maxHeight="360"
    maxFrameRate="24" par="4:3">
      <SegmentTemplate timescale="96" media="ed_$Bandwidth$bps/
      ElephantsDream_2s$Number$.m4s" startNumber="1" duration="192" initialization="
      ed_$Bandwidth$bps/ElephantsDream_2s_init.mp4" />
      <Representation id="320x240 46.0kbps" mimeType="video/mp4" codecs="avc1.42c00d"
       width="320" height="240" frameRate="24" sar="1:1" startWithSAP="1" bandwidth="
      46319" />
      <Representation id="320x240 91.0kbps" mimeType="video/mp4" codecs="avc1.42c00d"
       width="320" height="240" frameRate="24" sar="1:1" startWithSAP="1" bandwidth="
      90555" />
<!-- more Representations at increasing quality levels
-->
      <Representation id="1920x1080 3.8Mbps" mimeType="video/mp4" codecs="avc1.42c032
      " width="1920" height="1080" frameRate="24" sar="1:1" startWithSAP="1"
      bandwidth="3790686" />
      <Representation id="1920x1080 4.2Mbps" mimeType="video/mp4" codecs="avc1.42c032
      " width="1920" height="1080" frameRate="24" sar="1:1" startWithSAP="1"
      bandwidth="4228790" />
    </AdaptationSet>
  </Period>
</MPD>
```

Figure 4.3: Sample of MPD XML file, from MPEG-DASH dataset [76].

4.1.3 MPEG Media Transport

For support of segmented real-time video delivery, in 1996 MPEG standardized the digital container format MPEG-2 TS, as ISO/IEC 13818-1:1996, which is widely used in today's streaming systems. MPEG-2 TS is used for multiplexing audiovisual media data into one consistent delivery stream, structured into small packets with a fixed byte-size of 188 Byte [80]. For contemporary streaming scenarios, especially for streaming of high-resolution or omnidirectional video content, MPEG-2 TS might not be the best solution, as the small segmentation with fixed-length entities of 188 Byte might not be suitable for high resolution video delivery. Moreover, individual access of single parts of a media stream, e.g. a single audio-track of one particular language, or low-latency live-streaming are difficult to integrate into MPEG-2 TS [69][80]. To overcome these new requirements for an adequate multimedia delivery environment, MPEG MMT has been developed and published as international ISO standard ISO/IEC MPEG-H 23008-1 Part-1 [88]. It replaces MPEG-2 TS and defines a delivery architecture and container format for audiovisual media streams, especially for delivery over broadcasting environments or IP networks. More precisely, MMT defines four main Sections of media-delivery [88]: The composition of media-conent via HTML5; The encapsulation format and data-model, based on ISOBMFF packaging; The delivery through an own application layer transport protocol MMTP; The signalling, by defining a message format for properties of delivery and consumption of media packages. Moreover, to provide reliable delivery in IP networks, MMT introduces new QoS features by the so-called forward error correction (FEC).

According to [80], the main advantages of MPEG-2 TS are kept, whereas some simplifications and adjustments were introduced. The easy format conversion between simple storage and packetized streaming and the compatibility for future technologies are maintained. The MPEG-2 TS signalling message framework is mostly adopted. The simplifications mostly concern the fixed-packet size and the clock-reference, both of which are discarded.

The MMT architecture and data model is structured as follows: The smallest entity is the Media Fragment Unit (MFU), building a slice of media data and being independently processable. The next unit is the Media Processing Unit (MPU) containing one or more MFUs and being likewise independent from other MPUs. That is, the media content is packed into a series of MPUs. To ensure the independency of each MPU, besides the content itself, it contains metadata, i.e. additional configuration information for correct decoding and a fixed starting point. As a consequence, inter-coding dependencies and the GOP structure of the HEVC or AVC conform bitstream need to be considered. Further, one single or multiple MPUs are packetized into MMT assets, for each media type, like audio, video and additional data. These MMT assets are independent from each other and identified by a particular asset-ID, by which the MPUs can be assigned. Hence, the entire MMT package consists of several MMT assets, asset delivery characteristics (ADC) and composition information (CI), as shown in Figure 4.4.

As the data-model of the MMT MPU is based on ISOBMFF it is liable to its data structure. The MPU brand in ISOBMFF is `'mpuf'`. Consequently, each single MPU is packed into a single ISOBMFF file [80]. With reference to [88] and [80], correspondingly, two new box types are defined: The MPU box `'mmpu'`, in place of `'sidx'` and the MMT hint box `'mmth'`. The `'mmpu'` box identifies MPU related media content, like the sequence number and asset-ID, to assign the MPU file to the corresponding asset. The `'mmth'` box

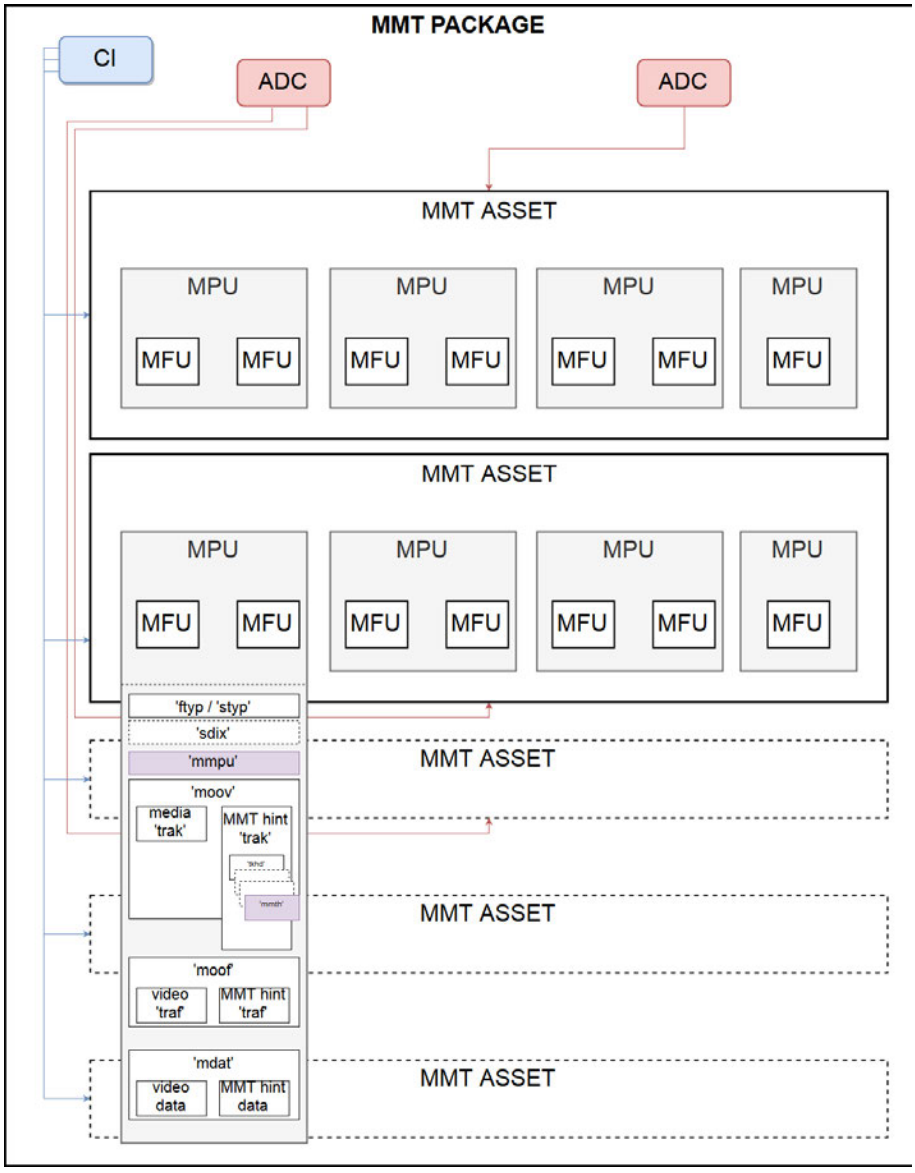


Figure 4.4: MMT Data Model and ISOBMFF structure, based on [88].

is part of the MMT hint `'trak'`, defining the number of MFUs, identifying the type of media data, i.e. timed or non-timed media data and informing about the media-samples. The MMT hint `'trak'` is placed in the `'moov'` box. As stated in [80], it also provides conversion information between encapsulated MPUs to MMTP payloads and packets. Like in the predefined ISOBMFF structure, the media data is placed in the `'mdat'` box, together with MMT hint data. The `'moov'` box contains the media track references and initialization information. For the exact ISOBMFF integration of MMT, the reader is referred to [80] and [88].

With MMTP, an own application layer protocol is defined, specifying the MMTP packet with the MMTP packet header and the corresponding payload. The MMTP payload is similar to the RTP payload structure. It can be used for real-time delivery, by the delivery mode MPU or for non-live on demand download by the generic file delivery (GFD) mode [69][80]. The MMT signalling can be delivered through the MMTP payload as binary format, providing information about the media delivery and consumption or timing-information. The signalling can be used for uni- and bidirectional communication within the delivery system. The explicit signalling messages can be obtained from [80] and [88].

4.1.4 MPEG-I

With the increasing development and spread of spherical video and gaming applications, many particular formats and technologies have been introduced. Most of them are based on existing technologies, but due to the multitude of developers and manufacturers many technologies are designed with deficient compatibility to others. Moreover, there is a need for long term specifications of spherical video and audio. Consequently, MPEG started the development of the standardization MPEG-I ISO/IEC 23090, for the coded representation of immersive media, which is expected to be launched in 2019. At this time, according to this first version and the technical report of it Champel et al. [33], MPEG-I consists of nine single parts:

- Part 1 – Technical Report on Immersive Media
- Part 2 – Omnidirectional Media Format (OMAF)
- Part 3 – Immersive Video
- Part 4 – Immersive Audio
- Part 5 – Point Cloud Compression
- Part 6 – Immersive Media Metrics
- Part 7 – Immersive Media Metadata
- Part 8 – Network-Based Media Processing (NBMP)

It comprises an overview on the technical architectures and general definitions for immersive media in Part-1, the standardization for an application format for spherical multimedia content in Part-2, coding standardizations for immersive video and audio in

Part-3 and 4 respectively, and new compression approaches by point-wise description of 3D media in Part-5. Furthermore, it defines new metrics for the evaluation of immersive media applications in Part-6. Part-7 specifies the immersive media related metadata, which may differ from those of conventional media applications. Processing of immersive or spherical media applications is much more complex and resource-consumptive than conventional media processing. For this reason, in Part-8 formats and interfaces are defined, to relay resource exhaustive immersive media processing tasks to the network [33].

The following chapters will concentrate on MPEG-OMAF, based on [51], with its supported projection schemes, profiles and the signalling in ISOBMFF and MPEG-DASH. OMAF was published in January 2019. It defines an application format for a system standardization with delivery, storage and rendering of spherical multimedia, i.e. 360° video and audio. Due to the extent and early state of the standardization and the focus of this work on streaming architectures, the other parts of MPEG-I won't be treated explicitly.

According to [33], MPEG-I makes some general clarifications and definitions on 360° video, not only for quality evaluation of omnidirectional media. Regarding the definition of resolution for evaluation purposes, in VR scenarios with HMDs, the notion of number of pixels per degree (pix/deg) is preferred, instead of absolute pixels. Moreover, the viewport of an entire 360° video (4π steradian space) is assumed to cover around 12-14%, resulting in only 1080 pixel vertical-resolution for each eye, for an 8K 360° source video. These assumptions apply to conventional delivery and playback technologies, at this state of time, and are caused primarily by the resolutions of today's cameras and playback systems.

4.1.5 MPEG - Omnidirectional Media Format (OMAF)

The omnidirectional media format is Part 2 of the MPEG-I standardization ISO/IEC 23090-2 and Part-20 of the MPEG-A standardization ISO/IEC 23000-20, specified in [51]. As aforementioned, OMAF is a system standardization for storage, delivery and rendering of 360° videos. It specifies sphere-to-planar mapping methods and the signalling of writing into a file using ISOBMFF, for encapsulation and transmission of the resulting media-file using MPEG-DASH or MMT. Moreover, it specifies profiles and encoding configurations for the use of HEVC and H.264. Viewport-dependency is supported by the use of tiles in HEVC, file encapsulation in ISOBMFF and segmentation in MPEG-DASH. Furthermore, the integration of the scalable profiles of AVC and HEVC were under consideration, but seem to be disregarded at this time of research.

This section is structured as follows: First, the determined supported mapping methods and other mapping-methods under consideration will be introduced. The frame-packing and region-wise packing will be explained thereafter. One goal of MPEG-OMAF is to provide available profiles and to determine the used video and audio codecs, which will be explained subsequently. Afterwards, the implementation of viewport dependency will be explained. At last, the signalling of OMAF in ISOBMFF and DASH will be delineated, which is an important part for standardized implementations of content creation tools and players. The overall system architecture of OMAF for 360° streaming, based on

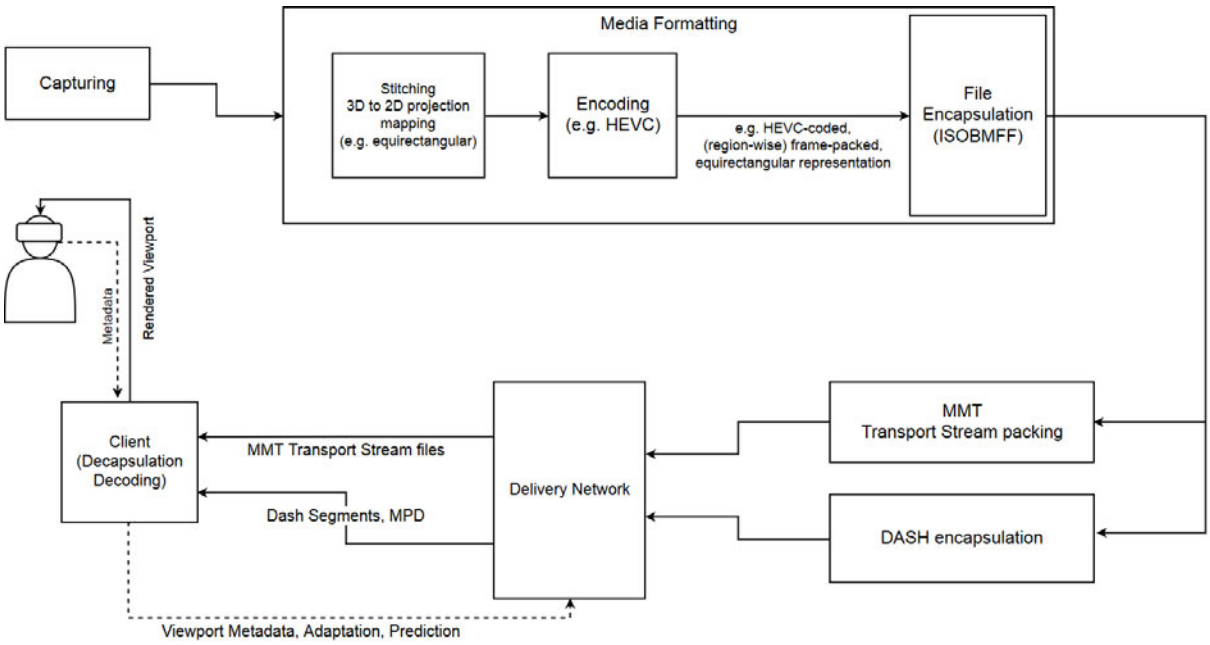


Figure 4.5: System architecture of MPEG-OMAF.

the descriptions of [35], [115] and [51], is illustrated in Figure 4.5.

4.1.5.1 OMAF - Projection mapping

According to [51],[115] and [35], in the first version of OMAF two projection types are supported: The equirectangular and cubemap projection. Other mapping methods are under consideration, some of those are discussed in Section 2.2. For projection mapping a coordinate systems needs to be determined, which in OMAF consists of the x,y,z axes and a unit sphere. The center of the sphere is The origin of the coordinate system is exactly at the center of the sphere, where the viewers perspective is located at, looking towards the inner surface. As stated in [115], the sphere coordinates can be described by (Φ,Θ). The conversion into sphere coordinates can be processed, by transformation of cartesian coordinates to global coordinates using rotations over pitch, yaw and roll and conversion from global coordinates to sphere coordinates. Global coordinates x_2, y_2, z_2 can be converted into sphere coordinates (Φ,Θ) as:

$$\Phi = \text{atan2}(y_2, x_2)\frac{180^\circ}{\pi} \tag{4.1}$$

$$\Theta = \text{asin}(z_2)\frac{180^\circ}{\pi} \tag{4.2}$$

In OMAF equirectangular projection the inner of the sphere is also unfold onto a 2D map, as described earlier in this work. The cubemap projection, as described in Section 2.2, is done by a six-sided cube, each of which representing one part of the sphere of the same size. Rearranging the cube-sides or parts of an equirectangular mapping in 2D representation, to match image content and similarities to coding dependencies is also specified in OMAF. it is called region-wise-packing and can enhance the coding efficiency. To avoid seaming and stitching artifacts, a guard band can be added at the borders of

the single packing regions, so that a small overlapping part is transmitted and signalled as part of the region-wise-packing syntax [115]. For representation to the human eyes, spherical video content needs to be split into left and right part, one for each eye. This arrangement needs also to be frame-packed. In OMAF it can be done by side-by-side packing, top-down packing or temporal interleaving. Additionally, OMAF supports the representation of content as a fisheye sphere. In this case no projection or region-wise packing needs to be done, but the playback system needs to be capable of processing the fisheye representation. This information can be send as metadata.

4.1.5.2 OMAF - Profiles and encoding configurations

The Profiles of OMAF predetermine encoding configurations and properties of the 360° video processing and transmission. They define specifications and constraints for ISOBMFF items and tracks, according to their media type, i.e. video, audio, image or timed-text. The OMAF profiles define the ISOBMFF sample entry types, possible extensions, i.e. box-types and the single samples of the tracks, i.e. elementary streams, according to the enabled configurations [51][115]. According to [51] a total of nine profiles is offered, 3 for video and 2 for audio, image and timed-text respectively. Additionally, 3 profiles for the CMAF integration of the media profiles are defined. The major difference between the video profiles are viewport adaptivity and the used video codec. Viewport dependency is enabled for two profiles. Other profiles and configurations are under consideration, so presumably other main or high-quality profiles might be introduced in following versions of OMAF. An overview of all OMAF profiles is given in Table 4.1.

Profile	**Brand**	**Profile & Level**	**Scheme & Type**	**Resolution/ Sampling-rate**	**Additional specification**
OMAF Video-profiles					
HEVC viewport independent baseline profile	HEVI	Main 10 Level 5.1	'podv' & 'erpv'	up to 4096x2160 @ 60fps HDR	no region-wise packing
HEVC viewport dependent baseline profile	HEVD	Main 10 Level 5.1	'podv' & 'erpv' or 'ercm'	up to 4096x2160 @ 60fps HDR	region-wise packing available
AVC viewport dependent baseline profile	AVDE	High Progressive-only 5.1	'podv' & 'erpv' or 'ercm'	up to 4096x2160 @ 25fps HDR	region-wise packing available
OMAF Audio-profiles					
3D audio MPEG-H baseline profile	oabl	Low-complexity Level 1,2 or 3	-	48000 Hz	3D metadata included in codec
2D audio legacy profile AAC	oa2d	HE-AACv2 Level 4	-	48000 Hz	no 3D metadata required

OMAF Image-profiles					
HEVC image profile	heoi	Main still 5.1	-	-	-
JPEG image profile	jpoi	-	-	-	-
OMAF timed text profiles					
IMSC1 timed text profile	ttml	text or image profile	-	-	-
WebVtt	ttwv	-	-	-	-
OMAF presentation profiles					
Viewport independent baseline presentation profile	ompp	-	-	-	min. one video-track HEVI, min. 1 audio-track oabl
Viewport dependent baseline presentation profile	ovdp	-	-	-	min. one video-track HEVD, min. 1 audio-track oabl
CMAF Media Profiles for OMAF					
CMAF Media Profile for the HEVC-based viewport-independent OMAF video profile	cvid	-	-	-	-
CMAF Media Profile for the HEVC-based viewport-dependent OMAF video profile	chev	-	-	-	-

CMAF Media Profile for OMAF 3D audio baseline profile	cabl	-	-	-	-

Table 4.1: OMAF media-profiles based on [51],[115] and [35].

For ISOBMFF signalling, the scheme type first defines, whether projection is used, i.e. `'podv'`, or no projection is applied and fisheye representation is chosen, i.e. `'fodv'`. The two introduced projection types, equirectangular and cubemap are determined by `'erpv'` and `'ercm'`, respectively. The selected HEVC profiles allow up to 4K resolution at 60 fps, with HDR enabled. The exact region-wise packing is not predetermined by the profile, but just enabled or disabled, e.g. enabled for `'hevd'`. The presentation profiles define global media profiles, specifying video configurations together with adequate audio configurations.

4.1.5.3 OMAF - Viewport adaptivity

Viewport dependency in OMAF can be achieved by different technologies of different efficiency and complexity. Conventionally, processing and transmission of a video has no viewport-adaptivity feature. In this case, the entire bitstream is encoded and transmitted in one single layer. For spherical video, the bitstream would represent a sequence of frame packed cubemap or equirectangular map pictures. Here, basically, the only possibilities of bitrate-reduction are at video encoding level, e.g. HEVC and distribution architecture-level, e.g. MPEG-DASH.

According to [115], one way to enable viewport dependency is the so-called region-wise quality ranking (RWQR). It consists of several encoded single-layer bitstreams, each of which representing the entire 360° video. The single bitstreams are encoded with one particular region encoded at high quality (HQ), the rest is encoded at lower quality, as depicted in Figure 4.6. Together with parallely, or afore transmitted metadata, informing about the respective region quality ranking, the client can request the best track, according to the users viewport. In ISOBMFF, the region-wise quality ranking method can be signalled in the sphere region quality ranking box `'srqr'` or the 2D region quality ranking box `'2dqr'`, in the sample entry. When using RWQR tiling is disabled. Hence, entire spherical video sequences are transmitted.

As stated in [51] and [115], OMAF integrates tile-based viewport adaptivity by the so-called motion-constrained tile set (MCTS), proposed by [119] for viewport-adaptive streaming of omnidirectional video. This approach is HEVC-based and uses bitstreams at two different quality levels, with tiling at encoding-level. Motion-constrained relates to the fact, that motion vectors can cross tile boundaries within the tile set. The borders of a tile-set may not be crossed, so no temporal motion-vectors between temporal-neighbouring tiles are allowed. The quality divergence can be achieved at encoding-level by change of bitrates or by change of resolution. The tile division of the single layer bitstreams should be equal for the two representations, when changing the quality by bitrate, whereas the tiles within one representation may be of different size. On an origin or edge server,

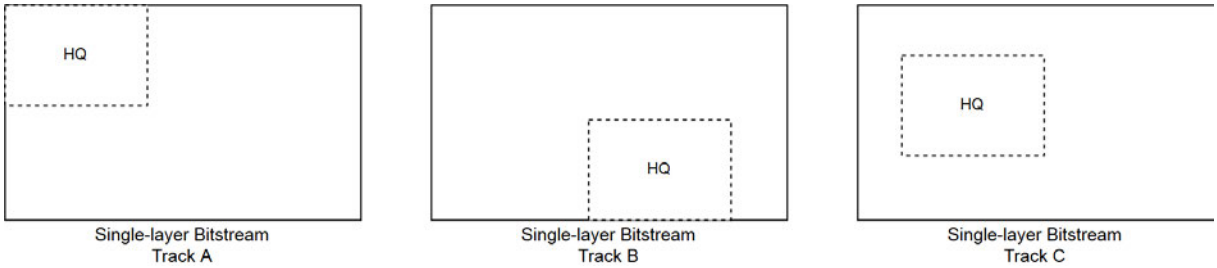


Figure 4.6: Viewport dependency by OMAF region-wise quality ranking, based on [115].

an intermediate extractor, e.g. a running script, extracts the tile-tracks of the higher quality bitstream, representing the viewport. All tiles covering the non-visible area of the omnidirectional video, are chosen from the lower quality bitstream. Next, the single tiles are put together, composing the whole spherical video, as a conventional HEVC bitstream, to maintain compatibility to existing HEVC-decoders. The decision of tile-switching, according to head-movements is based on sensory-feedback and metadata transmitted by the clients. The temporal independency of decoding certain regions, results in the need for switching points, that should conform to the SAPs. Moreover, to allow seamless switching and abolish inter-coding dependencies, intra-coded random-access points (RAPs) need to be embedded, without coding dependencies to other frames. It should be noted, that additional RAPs ordinarily impair the coding efficiency. This principle of using MCTSs at different qualities is illustrated in Figure 4.7.

In the example of Figure 4.7 two HEVC bitstreams of equal content and resolution, but with different qualities are created. Each frame is divided into a 4x2 MCTS. Each single MCTS is included into a single tile-track, referred to as sub-picture track, formed by the temporal following tiles of each MCTS, i.e. 'sub-picture track 1' for all 'tiles 1' etc. This is done at file encapsulation level. Here, the OMAF player receives tile-tracks 1,2,5,6 of higher quality for the viewport and tile-tracks 3,4,7,8 for the rest, lasting for a particular period of time. This selection is done via extractor tracks, that form a particular viewport, by pointing to single sub-picture tracks of the two qualities.

The OMAF player, i.e. omaf-capable playback device or intermediate extractor, chooses the respective version of each sub-picture track. Using these sub-picture tracks and the extractor track it can parse the single tile combinations, for reconstruction of a bitstream. This reconstructed bitstream can further be decoded with a conventional HEVC or AVC decoder.

According to [51], four different of these MCTS viewport-dependent transmission approaches are supported in OMAF. The first approach can be realized as described above. In this case the different tiled bitstreams are of different bitrate, i.e. quality, but have the same resolution. Here, the tile grid of the single bitstreams remains the same. To realize this approach for the AVC viewport-dependent profile, slices need to be used instead of tiles, as depicted in Figure 4.8. Considering that, the eight tiles of the HEVC approach above are replaced by eight single slices, constructing a motion-constrained slice set for each frame. The same inter-coding-dependency constrains need to be observed. The principle of using a sub-picture track, for re-consolidating the single slices, together with an extractor track, basically remains the same.

The HEVC-based MCTS approach is also supported for bitstreams of different res-

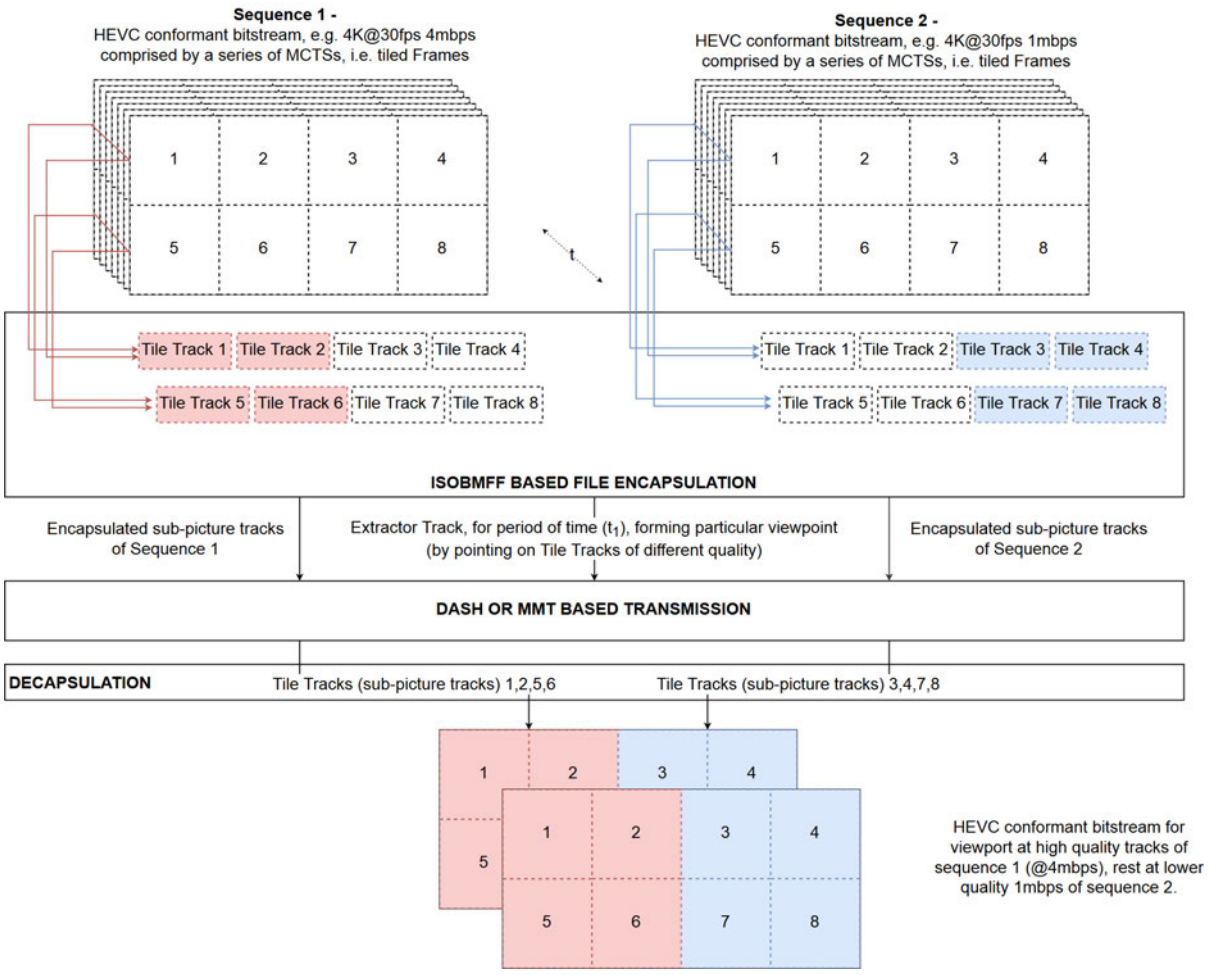


Figure 4.7: HEVC-based MCTS viewport adaptivity scheme of MPEG-OMAF for two sequences of the same resolution at different bitrates, based on [55] and [51].

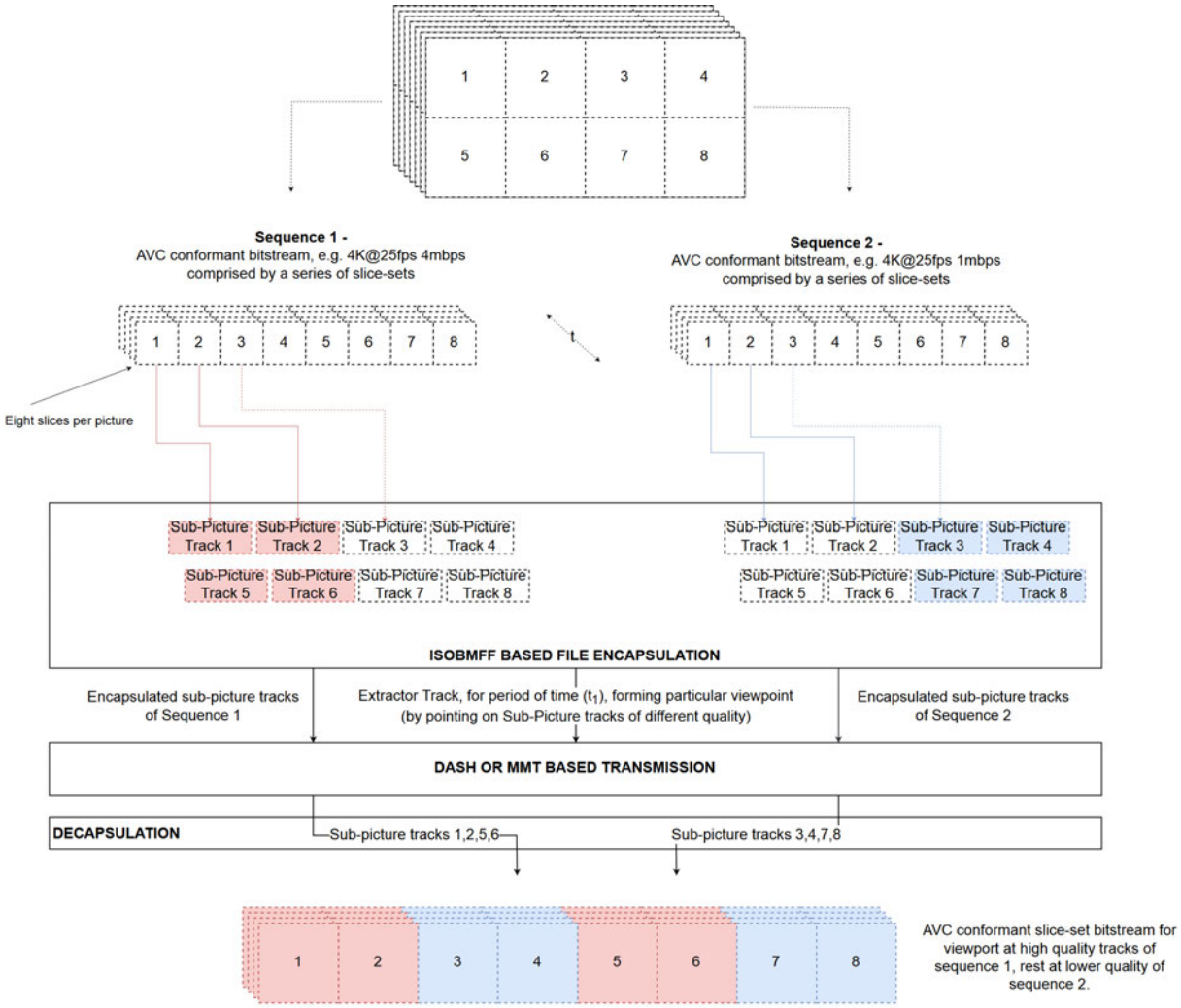


Figure 4.8: AVC-based viewport adaptivity scheme of MPEG-OMAF using slices for two content-similar tracks of the same resolution at different bitrates, based on [51].

olutions. As the degradation artifacts are others, compared to a low-bitrate sequence, for example blurriness compared to blocking artifacts, this approach may be suitable for other use cases. Sub-picture tracks and extractor tracks need to be created, similarly. An OMAF player should be able to merge and rearrange the two sub-picture tracks of different resolutions, in order to construct a conforming bitstream of one single resolution. By changing the tiling grid, i.e. creating inhomogeneous tile-sizes, the number of bitstreams, i.e. available resolutions, and the rearrangement of the different sub-picture tracks, together with the resulting extractor tracks, higher resolution than 4K for the whole sphere can be achieved. That results in effective higher resolution for the region covering the viewport, even for 4K-only capable decoders. The creation of viewport combinations in the extractor tracks, is the key-processing step for these viewport-adaptivity approaches with higher resolutions. These schemes for achieving 5K or 6K effective ERP or cubemap resolution are stated in precisely in Section D.6 of [51]. In this work, they will not be further explained, but may be used for the resulting 360° streaming-scenario, described in chapter 5. Moreover, apart from these MCTS approaches using tiles for all bitstreams, OMAF proposes the use of an un-tiled single layer bitstream for the lowest resolution or quality, together with HEVC-coded MCTSs for the higher resolutions or qualities. Sub-picture tracks are constructed likewise at file encapsulation level. For each viewport representation extractor tracks can be formed. That is, the lowest quality, a full-spherical representation is continuously sent and the distinct viewport representations can be decoded by selecting the different sub-picture tracks autonomously or by extracting them directly from the extractor tracks.

It should be noted that these viewport-adaptivity approaches were initially considered to be realized by the scalable profiles SHVC and SVC. The base-layer and enhancement-layer could be used for the content-representations of the respective quality. The enhancement layer could be used for higher quality representations, with tiles covering the current viewport. The base-layer could be transmitted continously, covering the whole sphere, for abrubt head-movement. This use of SHVC and SVC appears to be disregarded and is not further stated in the final OMAF standardization [51].

4.1.5.4 OMAF - ISOBMFF, SEI and DASH integration

As ISOBMFF is easily extendable to OMAF signalling, new MIME-types and box-types are defined in OMAF, some of which are presented in the following, according to [51],[115], [52]. Due to the focus of this work on video streaming, storing and signalling of omnidirectional still images, via the OMAF still image profiles, are not further discussed. Captions and timed text in OMAF can generally be integrated with fixed position or viewport-dependently, signalled by the timed text configuration box `'otcf'`, informing about the mode, i.e. fixed-position or viewport-dependent/ always-visible and the corresponding coordinate information. The initial viewing orientation with the sample entry type `'invo'` and other metadata, like the recommended viewport and timed text sphere location metadata can be signalled as a timed metadata track by a `'cdsc'` track reference. The recommended viewport can either be predefined e.g. by a director (type 1) or statistically derived (type 2).

First, to distinguish between conventional video and spherical video, the entry-type `'resv'` defines the restriction, that only capable decoders can extract the content correctly. A restricted scheme info box is supplied to the sample description, informing about

the omnidirectional video. The exact codec profile is delivered by an *original format box* `'frma'` for indication of the HEVC packaging `'hvc1'` or `'hvc2'` or the H.264 packaging `'avc1'`. The OMAF profiles are signalled in the `'ftyp'` box, as `'hevd'`, `'hevi'` or `'avde'`, corresponding to the described profiles of Section 4.1.5.2. As proposed in [52] and [115] for streaming scenarios, multiple ISOBMFF files can be created, each of which containing the same content as one single valid HEVC or H.264 bitstream, i.e. a set of MCTS tracks, at a certain quality. A flag of the *TrackHeaderBox* indicates that the single ISOBMFF tracks are not intended for separate presentation, but in combination with others.

As already noted, viewport-dependent profiles may produce streams and corresponding viewports of higher quality and should preferably be used for streaming scenarios. In this case, the client should receive one or more particular tracks of higher quality, i.e. the viewport and all others of reduced quality, according to the chosen OMAF profile. The respective quality of the related region can be signalled by the *sphere region quality ranking box* `'srqr'`. For each MCTS, a corresponding *mctsID* is generated. The corresponding MCTS tracks have a particular `track_ID`, which can be calculated from the `mctsID`. As proposed by [52], in addition to the single MCTS bitstreams packed as single ISOBMFF files, multiple ISOBMFF extractor track files are generated, building several possible tile combinations. The extractor track files build specific viewport combinations, each of which composed of one extractor track and multiple MCTS tracks, i.e. tiles of a specific quality level. All tracks are added to the `'moov'` box of the extractor file. These tile combinations produce a rectangular 360° frame of a fixed resolution, corresponding to the chosen OMAF profile. The projection and spherical indications are signalled in specific boxes.

According to [115], the following boxes are defined for 360° video signalling in OMAF. The distinction between projection or fisheye is indicated by `'podv'` for projection and `'fodv'` for fisheye, i.e. no sphere-to-planar mapping. Equirectangular or cubemap projection is signalled by `'erpv'` and `'ercm'` respectively, other projection types under consideration may be added in the future. When region-wise packing is used, it can be indicated by `'rwpk'`. The rotation for projection can be signalled by the rotation box `'rotn'`, where three integer values define the rotation of yaw, pitch and roll. For signalling of the frame-packing method, the existing stereo video box `'stvi'` in the sample entry can be used. The indication which area of the sphere is covered by a which track is important for the OMAF player, to choose the appropriate track, i.e. tile that covers the viewport. The coverage information box `'covi'` in the sample entry can be used for this indication. The OMAF ISOBMFF integration into DASH, based on [115], is depicted in Figure 4.9.

As stated in [52], OMAF-related metadata may be signalled as high level syntax via SEI messages of the H.264 or HEVC file. Information about the use of equirectangular projection, cubemap projection, sphere rotation, region-wise packing and the viewport location may be signalled as SEI. For specific SEI parameters of OMAF the reader is referred to 5.5.2 of [52] and [51]. According to [51], for DASH delivery of OMAF-specific information the MPD is extended by additional MPD DASH descriptors, accessible by the URN `'urn:mpeg:mpegI:omaf:2017'` (hereinafter referred to as namespace *a)* and `'urn:mpeg:mpegI:omaf:2018'` (hereinafter referred to as namespace *b)*, defining the namespaces for the XML elements and attributes. The projection type ('pf'), region-

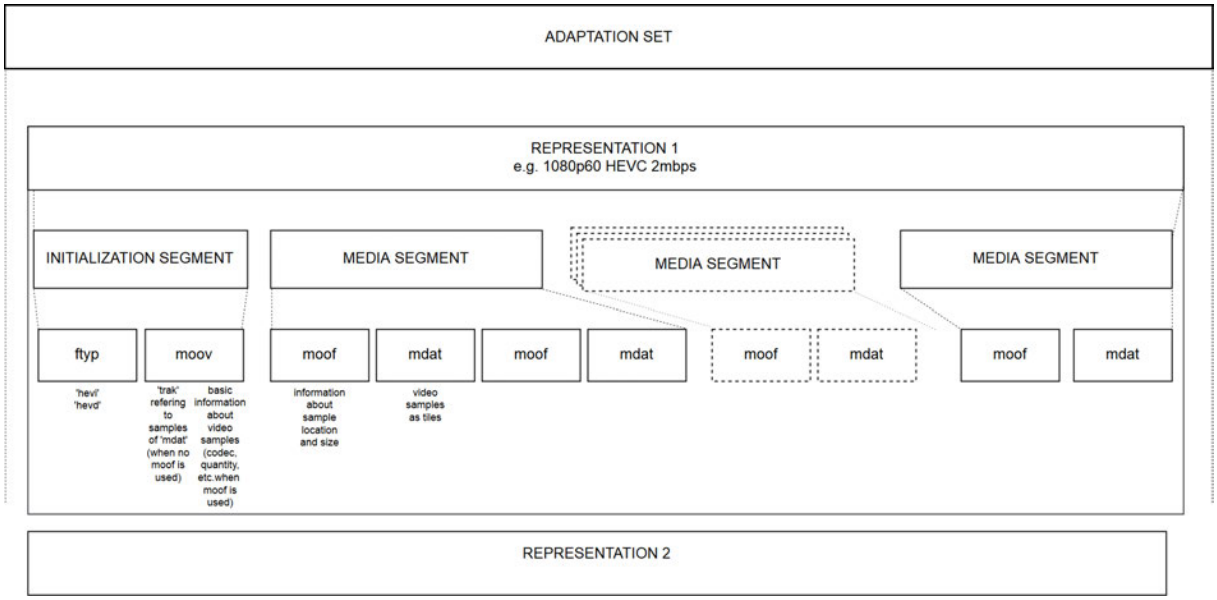


Figure 4.9: DASH segmentation and ISOBMFF integration, based on [115].

wise-packaging (`‘rwpk’`), content coverage ('cc'), spherical and 2D region-wise quality ranking (`‘srqr’` and `‘2dqr’`) and fisheye omnidirectional video (`‘fomv’`) DASH MPD descriptor, are accessible at namespace a. Exemplary, the scheme identifier for the projection type is defined by the URN `‘urn:mpeg:mpegI:omaf:2017:pf’`. Other attributes, like the association descriptor (`‘assoc’`), viewpoint information descriptor (`‘vwpt’`) and overlay information descriptor (`‘ovly’`) are accesible at namespace b.

According to [51], the particular OMAF video profiles should be instantiated at the Adaptation Set, as
`‘@codecs=‘resv.podv+erpv.hvc1.1.6.L93.B0’’`
`‘@mimeType=‘video/mp4 profiles="hevi"’’`,
exemplarily for the HEVC viewport independent OMAF profile, using equirectangular projection and the hvc1 MP4 container.

As stated in [52], the OMAF DASH workaround differs for the HEVI and HEVD profile. The viewport-independent profile, mainly for progressive download applications, has one single Adaptation Set per Period, containing several representations, at different bitrates and/or resolutions, according to the HEVI-specific requirements. For the HEVD profile, multiple Adaptation Sets per Period, each of which containing multiple representations are used. This principle is illustrated in Figure 4.10.

Each single Adaptation Set represents one tile per resolution. The single representations of an Adaptation Set, i.e. MCTSs only differ by their respective bitrate. Besides to the Adaptation Sets for each tile, different viewing directions are pre-constructed by additional Adaptation Sets. These point to the respective tiles at the different quality levels, where the viewport is covered by high-resolution tiles and the rest at lower resolution, as Adaptation Set 10 and 11 of Figure 4.10.

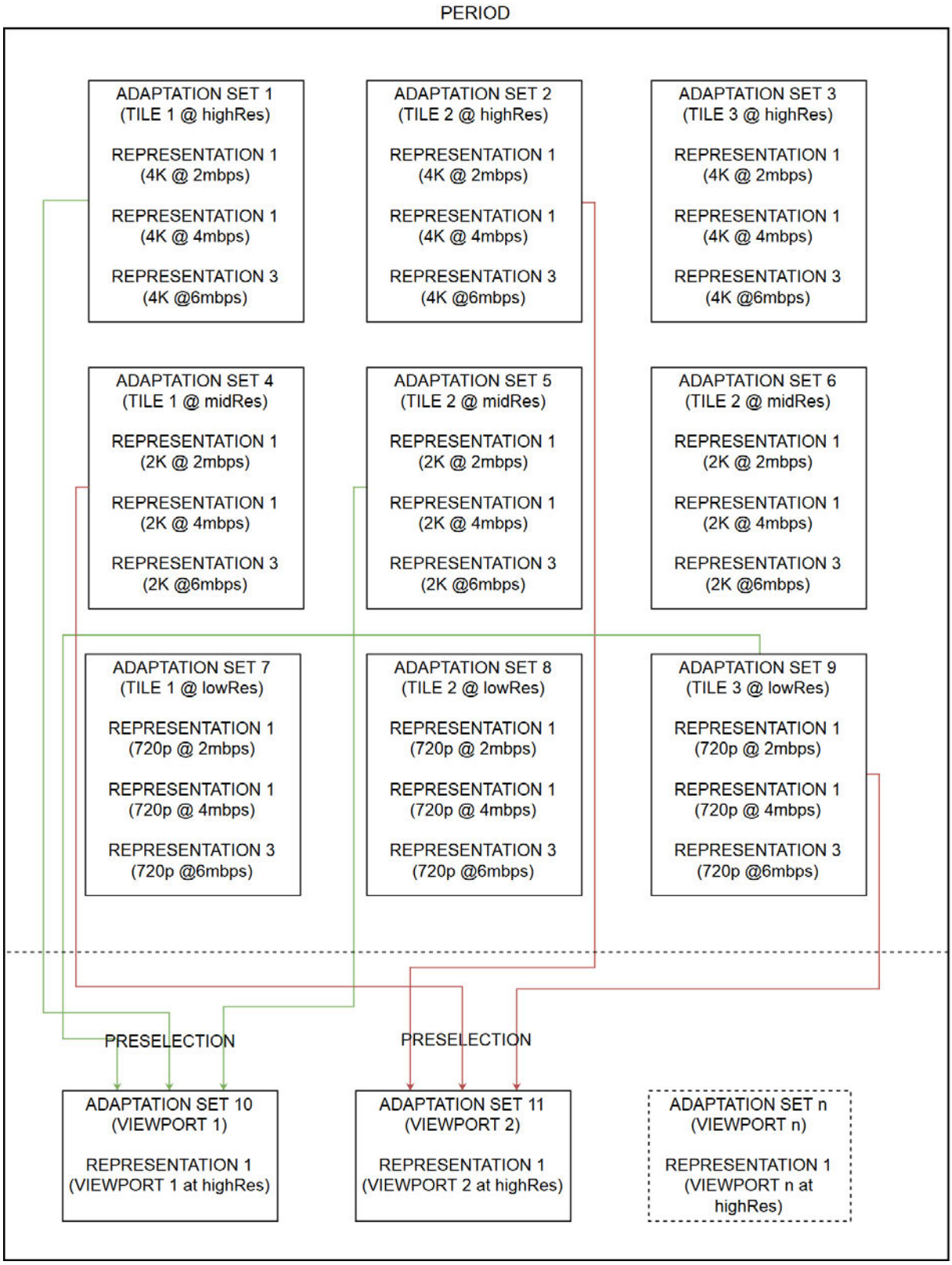


Figure 4.10: OMAF DASH Adaptation Set architecture for the HEVD profile, based on [52].

4.2 JCT-VC 360° video signalling standardization via SEI

As stated earlier, SEI messages can be used for signalling of supplemental information within h.264 or HEVC-encoded video. For this purpose, in 2016 JCT-VC initiated a corresponding standardization for SEI messages, as amendment to the HEVC standard ISO/IEC 23008-2:2017/Amd 3:2018 [31], published in 2018. According to [30], this amendment includes SEI message standardizations for HDR video signalling, omnidirectional video signalling, regional nesting and MCTS extraction information signalling. The following section concentrates on SEI messages for omnidirectional video and hereby MCTS extraction messages, HDR and nesting SEI will not further be treated. SEI messages are information that do not directly affect the decoding process. Either the decoder can interpret them correctly to obtain enhancement information for decoding, or the SEI can be skipped.

The standardized omnidirectional video SEI messages comprise the projection mapping and coordinate systems indications, the region-wise quality indication as stated in Section 4.1.5.4, the region-wise packing and the indication of a particular viewport. Additionally, the MCTS extraction SEI messages can be used for deriving a HEVC conforming bitstream by extraction of MCTSs.

As proposed in [29] and [30], the `'geometry_type'` informs about the coordinate system used for projection, i.e. spherical coordinates or cartesian coordinates. The `'projection_type'` SEI message indicates whether equirectangular projection, cubemap projection or rectilinear projection is set, i.e. type 0, 1, 2. The coordinate-system center position is defined by yaw, pitch and roll center values. The coordinate-system expansion, i.e. the ranges for the yaw and pitch values are determined by `'yaw_range'` and `'pitch_range'` respectively. The rectilinear projection type indicates a predetermined viewport, as an extracted region of an entire spherical video bitstream, with its corresponding position and size. In this case, first the region is extracted and afterwards rectilinear projection is used. The omnidirectional viewport SEI message, can be used for definition of at least one certain viewport, represented by spherical coordinates. If the bitstream is encoded with quality grades, a corresponding indication for the quality of a certain region is necessary, which can be signalled as region-wise quality indication SEI message. The RWP can also be signalled as SEI message, as already stated in Section 4.1.5.4.

As Boyce et al. stated in [30], the integration of MCTS extraction information into SEI is defined as follows. So-called extraction information sets are defined, containing identifiers for several MCTSs. That is, each extraction set covers information for a set of MCTSs. This extraction is done by replacing and modifying the VPSs, SPSs and PPSs structure, resulting in a new composition of several MCTSs, forming a conventional HEVC-conforming bitstream. For the exact process of MCTS extraction information set SEI message signalling and detailed information about omnidirectional SEI messages, the reader is refered to [30] and [31].

4.3 Related 360° video streaming approaches

Besides MPEG's 360° video standardization OMAF, many other system-approaches exist, some of which are described in papers or tutorials, others are not accessible, due to licensing issues. This section gives an overview of related works, besides the extensively described approach of MPEG OMAF.

In [26] Bassbouss et al. introduce an ISOBMFF-DASH-based streaming system for 16K omnidirectional video source-material, resulting in a 4K viewport. They only transmit the viewport area, without the use of tiling. As for interoperability reasons, the transformation and video-preprocessing is server-sided, less post-processing is needed at client side. For this reason, viewport combinations are pre-generated out of an equirectangular spherical source. However, as this approach is desgined for playback on conventional 2D screens, it is not suitable for continuous and free viewport selection, as playback on HMD would require.

Gankhuyag et al. [54] designed a motion-constrained AV1 encoder for omnidirectional tiled streaming. It can be considered as alternative to the HEVC tiling approach. Still, the paper concentrates on the encoding via AV1. The transmission-architecture is not regarded in detail and not in focus of the paper.

Another HEVC tile-based approach of 360° video transcoding and streaming, using MPEG-DASH is proposed by Kim et al. [70]. This approach has similarities to MPEG-OMAF using different resolutions for the viewport tiles and all other regions. For this reason they use two decoders, one for the viewport tiles, and another one for the entire spherical video at lower resolution, without tiling.

In the following, another approach by Ahamdi et al. [23] is described more detailed, as it is extensively described in the corresponding paper and fits the use case. Due to the large scope of possible approaches, not all can be considered and described in detail.

Ahmadi et al. [23] proposed an DASH HEVC tile-based streaming architecture for 360° video, with similarities to the proposed DASH-based OMAF system. The following synopsis is based on the corresponding descriptions of [23].

In the system, they establish multicast by so-called *Multimedia Broadcast Multicast Service* (MBMS) base stations, that employ point-to-multipoint bearers. Moreover, they use an approach of weighting tiles, depending on the user's viewing behaviour. That is, a tile weighting prediction technique to address viewport changes is introduced. A new rate adaptation algorithm, maximizing the average data-rate is proposed. These introduced technologies lead to a higher available data-rate for the viewport of 46%, by not increasing the overall needed bandwidth. The recorded source material is mapped using equirectangular and cubemap projection. Multiple quality representations of certain bitrates are created and afterwards temporally divided into several Segments and spatially divided into several tiles. These representations are then stored on a server together with an MPD. Additionally, a SRD is supplied, containing information about the dependencies between the spatially divided tiles. According to the current viewing orientation the decoder of the client selects the respective tile-tracks to decode. The MPD and service description are supplied to the *Broadcast Multicast Service Center* (BM-SC) and the MPD is updated periodically by the content provider, so it contains all representations available on the content server. They recommended to create one MPD for unicast and one for multicast. After the MPD request of the client and requesting of the respective

Segment, the MBMS service bearer is activated and MBMS delivery function of the BM-SC is triggered to send the desired tiles to all listening clients. The respective video tiles covering the current viewport are communicated during the session by the MBMS client to the BM-SC. Additionally to consumption information, transmitted as XML service description, extra information of the weighting of each video tile can be easily supplied into the XML. Using this weighting information and a rate adaptation algorithm the BM-SC specifies the appropriate tiles of each representation that needs to be transmitted. The MPD file is updated by the BM-SC according to the output of the rate adaptation algorithm, hence, only tiles determined by the algorithm will be set as available. Based on this updated MPD file and information about the channel conditions, the client selects the representation for each tile and starts downloading them. For finishing the stream, the client deregisters from the service and deactivates the MBMS service bearer.

As already stated, the tiling structure is strongly affecting the overall transmission efficiency. On the one hand, with smaller tile sizes, more inter-prediction dependencies need to be broken and more multicast sessions need to be established requiring more resource blocks, on the other hand, smaller tiles lead to more precise ROI resolution. The proposed tiling configuration of [23] is based on an analysis of a head traces dataset, with information about the bitrate overhead, the viewport change distribution and the ROI coverage. They decide to use a 4x8 tiling grid, resulting in only 3,78% bitrate overhead, compared to no tiling. In the proposed system, all tiles are streamed, but with different bitrate weights. Three different weighting approaches are proposed:

- Binary - Fixed viewport definition and no weighting; All tiles within the viewport have quality 1, all others quality 0.
- Pyramidal - Weighting relative to the distance of each tile to the current viewport, graded weighting with more than two weighting-grades.
- Probabilistic - Head traces-based probability weighting; Probability of viewport changes based on user's watching habits; Small-angle changes are regarded more likely; Probability of tile changes, i.e. viewing angle-changes is calculated.

The proposed rate adaptation algorithm, referred to as *Multicast Virtual Reality* (MVR) algorithm, results in transmitting video tiles with maximum weight at maximum available bandwidth. All other parts are streamed with quality according to their weights. When, in practice, the resources are limited, the quality is reduced accordingly to the available resources.

The authors of [23] state, their approach to address nearly twice the bitrate (46%) to the viewport tiles, than to all other tiles. Simultaneously, the viewport bitrate variations over a short period of time are minimized, to not affect the immersion negatively, by switching the quality too often. Moreover, the spectral efficiency, i.e. transmitted data rate (bit/s) divided by the allocated bandwidth (HZ), is maximized. For more detailed information about the tile-weighting, the Rate-adaptation algorithm and an evaluation of this system, the reader is further referred to [23].

4.4 Evaluation of proposed streaming technologies

Below, the proposed streaming systems are evaluated under consideration of efficiency, usability and feasibility. Moreover, the benefits and drawbacks of standardized approaches like OMAF are roughly compared to proprietary and open-source solutions.

OMAF uses the most popular projection mappings, like equirectangular and cubemap-projection, which are already approved and widely used. The integration of H.264, HEVC, ISOBMFF, DASH and MMT is comprehensible, since they are very efficient, popular and approved technologies. OMAF supports viewport-adaptivity, by using and extending existing technologies, without increasing the complexity and the computational costs to the limit, even though viewport-prediction is not considered. The computational power and processing is allocated to the client, together with a running script on a server preselecting the viewport tile combinations. The different profiles fit today's state of the art technologies, match multiple use cases and simplify the use of the proposed technologies. Moreover, 4K viewport support is enabled and 3D audio is supported, which correlates to today's state of technology.

The system of Bassbous et al. [26] is designed for 2D playback on conventional screens, without 'live' viewport-adaptation, albeit HbbTV integration and high resolution are introduced. This server-sided approach provides interoperability and easy access to several clients. Still, it does not fit the use case for playback on all kinds of playback-devices and fluent viewport-adaptation.

Since Gankhuyag et al. [54] propose an AV1 tiling-scheme, their approach is free of charge and might be comparable to HEVC tiling. Nevertheless, no entire architecture design is proposed and integration into common streaming systems is not stated precisely.

Kim et al. [70] proposed a HEVC tile-based approach with DASH integration, with many similarities to OMAF. Since it is not standardized and the documentation of OMAF is very extensive, the implementation and usability is likely to be more difficult than OMAF.

The multicast solution of Ahmadi et al. [23] is DASH-based and uses probabilistic-based tile weighting to build viewport combinations, which is very advanced. Moreover an own effective rate-adaptation algorithm is proposed. The implementation and administration of this system seems to be very complex and extensive.

The proprietary solutions often include, open-source technologies, which are easily available and royalty-free. These are independent from commercial entities in the media-technology industry and usually can be integrated into existing frameworks. In contrast, system integration and system design are often very complex and time-consumptive, regardless of the risk that the proprietary solution might be less efficient than the standardized one. Further, for integration and implementation of a certain proprietary solution the entire system needs to be regarded and a lot of research is necessary in advance. Existing interfaces need to be adjusted to fit the used protocols and data exchange. The administration either has to be simplified or the person in charge needs to be instructed and trained.

Doubtless, OMAF, as a standardized omnidirectional streaming system, will charge licence-fees and induces dependencies on other MPEG-based technologies. Open-source technologies, such as AV1 or VP9 are not supported and efficient technologies of other companies might not be regarded. The adaptation and integration of technological-

progress into the standard is a time-consumptive process, as particular standardization review procedures need to be maintained. For instance, the integration of new sphere-to-planar mappings would require a revision of the proposed approach and an extension of the ISO standard. On the other hand, OMAF can be easily and uniformly integrated, based on efficient, approved and widely used standards such as H.264, HEVC, MPEG-DASH and ISOBMFF. Presumably, the support lasts over a long period of time and new deployments will be updated in frequent periods. A broad dispersion of OMAF in the future is supposable, leading to integration into future technologies, like camera systems and playback technologies. As the big German broadcasting corporations widely use MPEG technologies such as codecs and streaming systems, the integration of HEVC tiling, ISOBMFF, DASH and OMAF into the streaming-network of the IRT will be the scope of the practical elaboration of this work, documented in the next Section.

Chapter 5

Implementing and testing different streaming systems

5.1 Demands and use case scenario

The IRT is responsible for research and development, and proposing recommendations for the public German broadcasters ARD and ZDF. For this reason, the final use case also relates to the demands of ARD and ZDF, i.e. their omnidirectional video applications. Distribution of video content over online platforms like *ARD-Mediathek* or *ZDF-Mediathek* is emerging. Likewise, such applications might be an adequate way to offer omnidirectional content. In this sense, the platforms serve as hosts for the provided content and shift the distribution liability to the Akamai CDN. The distribution network and server infrastructure of Akamai CDN will be examined in Section 5.2.1. The responsibilities of the broadcasters for allocation of 360° content hereby primarily concern the recording, pre-processing, formatting, and distribution. The playback itself is hardly to be exactly specified, as many users use different playback devices and no standard workflow is established yet. Generally three types of playback need to be distinguished:

- Playback on popular internet browsers, with manual viewport selection.
- Playback on smartphones used as HMDs, with automatic viewport selection.
- Playback on HMDs, with sensor-based automatic viewport selection.

There exist several ways to supply the users with the corresponding media-files adapted to each playback environment. One way would be to use a running script on the server, normalizing the spherical representation into a conventional planar video representation, decodable by conventional clients. Depending on the range and scale of the media content to be streamed, e.g. live-event, VOD etc., this requires a lot of computational power on the server side, to generate a viewport for each watching client. Another architecture, relocating more computational power to the client, could be realized by OMAF integration. In this way, multiple client environments can be served, assuming they have OMAF integration. At this very early state of 360° video technology, standards like OMAF are not yet integrated by many manufacturers and can not be considered as standard workflow. Nevertheless, assuming that OMAF integration becomes more and more common

in future, an OMAF-based system architecture might be an interesting solution for ZDF and ARD and could be tested by the IRT previously. Despite this option, the general integration of a HEVC-based MPEG-DASH streaming system for omnidirectional content seems expedient, as it corresponds to the conventional streaming infrastructure of the broadcasters.

Consequently, four main demands were defined, which are essential for the implementation of the omnidirectional streaming system.

1. **Usability for supplier:** Intuitive and simple operability and handling of the system.

2. **Quality:** Noticeable enhanced quality, compared to conventional streaming of 360° content.

3. **Integration:** Possibility of integration and implementation into existing infrastructure and hardware of the IRT, i.e. ZDF and ARD.

4. **Cross-platform support:** Playback and accessibility for a multitude of playback environments, e.g. monoscopic, stereoscopic, HMD, 2D-screen.

Some of these demands may contradict or influence each other, which is why not all can be realized entirely yet. Other aspects of importance are the financial cost, the disposability of appropriate omnidirectional content, the market development of the used streaming technology and the storage and computational resources allocated at the servers. Depending on the content to transmit, some of these aspects will be more important than others. A plausible use case scenario for future omnidirectional video streaming applications could be live-events. Supposing a sport event like the football world cup should be provided as spherical video. In this case, the content would have to be encoded, packetized, and transmitted live to ensure low-delay playback. Computational resources for viewport-dependent transmission would have to be aligned to the clients, to overcome immense server workload. For that reason, server-sided approaches with high computational complexity might not be appropriate.

However, for evaluation of the consulted implementations 360° video content needs to be created, due to copyright issues. As for the test, the content itself is less relevant, more importance should be attached to the technical properties of the test content. The generation of test sequences is stated in Section 5.2.2. The evaluation of adequate omnidirectional streaming systems of this chapter is divided into three main steps:

1. **Test content** generation, recording and post-processing, in Section 5.2.2.

2. **Implementation** of HEVC tile-based MPEG-DASH streaming approaches, of three different developers: GPAC Telecom ParisTech[1] [6], Nokia Technologies[2] [108], Fraunhofer Heinrich-Hertz-Institut[3] [90], in Section 5.3.1, 5.3.2, 5.3.3. With **Integration** of DASH elements into the Akamai CDN of the IRT and distribution of content and test-wise playback in particular environments for each section.

[1]GPAC Telecom ParisTech: Hereinafter referred to as GPAC.
[2]Nokia Technologies Corporation: Hereinafter referred to as: Nokiatech.
[3]Fraunhofer Heinrich-Hertz-Institut: Hereinafter referred to as Fraunhofer HHI or just HHI.

3. **Evaluation** of the different implementations.

It should be considered, that neither the approach of GPAC, nor the implementation of Nokiatech and the HHI enable a native support for immediate playback on HMDs. Albeit, Nokiatech provides the source code for playback integration into Android smartphones. The implementation of the HHI provides a content creation toolkit, and a JavaScript browser player. Moreover, the solution of GPAC is based on HEVC, using tiles and MPEG-DASH, but still has no standardized OMAF integration. The other two implementations use OMAF and may be pioneers for integration of OMAF-based streaming solutions. Additionally, the approach of Tiledmedia [13] will be roughly analyzed, by comparing the possible streaming properties and the playback to the other approaches.

5.2 Streaming infrastructure and available resources

5.2.1 Akamai CDN

The corporation Akamai is a provider for content delivery network systems, headquartered in Cambridge US. They are offering network services for multiple different content providers and large companies,and administer more than 50 terabit daily Web traffic per second [62]. It possesses a huge origin and edge server infrastructure scattered over the whole world, with 240.000 servers in over 130 countries and over 1.700 networks worldwide [62]. Besides media delivery and network administration, which are the key features for the mentioned use case, Akamai offers Cloud networking, Cloud security systems, and web-performance optimization services. The server of the IRT in this case, can be used as origin server, passing the content through the Akamai gateway to the edge server, which the respective client can access. It enhances the content distribution performance, administrates the data transmission between each client and server, and is hereby responsible for transmission quality and security. The server address of the corresponding origin test server, where the test content will be hosted is available at
`http://akamai-progressive.irt.de/masterarbeit_testfiles/`, with corresponding subfolders for each implementation. The content integration of the implemented streams, i.e. DASH MPDs, Segments etc. into the IRT test server and hereby Akamai CDN is done by a ftp administration tool.

5.2.2 360° audio-visual test content

The test sequences, for subsequent DASH processing and integration into the CDN, are recorded by the Insta360 Pro 2.0 camera, [9]. Table 5.1 gives an overview of all test sequences and corresponding recording parameters. All recordings are post-processed in the available tool Insta360 Stitcher[4]. The camera records the footage of each lens on each of the six single micro SD-cards. Additional information and instantaneously stitched files are written on an additional SD-card. Each single fisheye recording has a resolution of 3840x1920 at 120 Mbps variable bitrate in AVC with profile Main@L5.2, with bit-depth of 8 bits. The recordings were mapped into equirectangular representation and encoded into various technical representations, differing in codec, framerate, resolution, bit depth,

[4]Available at: https://www.insta360.com/download/insta360-pro2

Content name and description	Length	Format	Encoding parameters
`01_Binnenalster.mp4` Hamburg city impressions Binnenalster, townhall, cityscape	01:27 min	MP4 x.265 8 bit 4:2:0	7680x3840 30fps 223Mbps 2D
`02_Lombardsbruecke.mp4` Hamburg city impressions Lombardsbrücke, townhall, cityscape, traffic	01:01 min	MP4 x.265 8 bit 4:2:0	3840x1920 60fps 222Mbps 2D
`03_Aussenalster.mp4` Hamburg city impressions Außenalster, ride, pedestrians	05:47 min	MP4 x.265 8 bit 4:2:0	3840x1920 30fps 223Mbps 2D
`04_Lombardsbruecke.mov` Hamburg city impressions Lombardsbrücke, townhall, cityscape, traffic	01:01 min	MOV ProRes422HQ 4:2:2 10 bit	7680x3840 60fps 7890Mbps 2D
`05_Lombardsbruecke-02.mov` Hamburg city impressions Lombardsbrücke, townhall, cityscape, traffic	01:24 min	MOV ProRes422HQ 4:2:2 10 bit	3840x1920 120fps 3967Mbps 2D
`06_Lombardsbruecke-02.mp4` Hamburg city impressions Lombardsbrücke, townhall, cityscape, traffic	01:24 min	MP4 x.265 8 bit 4:2:0	7680x3840 60fps 223Mbps 2D
`07_Binnenalster.mp4` Hamburg city impressions Binnenalster, townhall, cityscape	01:27 min	MP4 x.265 8 bit 4:2:0	7680x3840 30fps 223Mbps 3D (top,bottom)

Table 5.1: Specification and encoding parameters of recorded test sequences, with the Insta360 Pro 2 and Insta360 Stitcher software for post-processing.

stereoscopy, etc. The content are three main motives, all cityscapes in Hamburg, Germany. Two still recordings, where the camera was put on a tripod, and one recording is a tracking shoot, of a longboard trip. So, steady and moved content is recorded. As the movement of the tracking shoot is shaky, some stitching and motion errors occur at faces of persons. The encoding was performed with more than one file, albeit the documentation in this section and the following Section 5.3 relates to one single clip. It was chosen to record with various camera settings, such as various framerates of e.g. 30, 60, 120 fps, resolutions and 2D or 3D of the same content.

After successful stitching of the six single recordings, i.e. perspectives, the files were transcoded into RAW YUV format by `ffmpeg` with the corresponding command line of Listing 5.1, to prepare them for Kvazaar encoding. In this case the RAW video was downsampled into 4K resolution, and reframed to match the source clip parameters of the corresponding guide [75]. Two exported example frames of the recorded test content `06_Lombardsbruecke-02.mp4` are given in Figure 5.1a and 5.1b. A screenshot of the

stitching process and exemplary fisheye representation of one single lens recording is depicted in 5.1c, noticing that six of these single lens fisheye recordings are used to create the full spherical equirectangular representation.

```
$ ffmpeg -i 06_Lombardsbruecke-02_8K60fps2D_HEVC.mp4 -c:v rawvideo -vf
    scale=3840:1920 -r 30 -pix_fmt yuv420p 06_RAW_Lombardsbruecke-02
    _4K30fps2D_HEVC.yuv
```

Listing 5.1: FFMPEG transcoding into RAW YUV format, for post-processing with the Kvazaar encoder [19].

5.3 Realization of different 360° streaming systems

5.3.1 GPAC Kvazaar HEVC tile-based adaptation guide implementation

Since viewport adaptive streaming of 360° videos by using tiles is an emerging topic, there are several implementation options. One way to realize HEVC tile-based adaptation in DASH is the proposed basic tutorial [75] of GPAC [17], that was implemented and roughly tested during this work. The following section documents the implementation steps of this approach and can be understood as one example of the stated guide. It is worth to note, that implementing this GPAC solution step by step is a challenging task, requiring many additional libraries, tools and Github[5] implementations to be integrated.

Since the HEVC encoding of the proposed approach is based on the Ultravideo Group Kvazaar Encoder [14], first the corresponding Kvazaar version 1.2.0 Github resources [19] and manual were used to compile a functioning Kvazaar build in Microsoft Visual Studio 2017. The following steps were done to create the Kvazaar executable, to be run on Windows10, based on the instructions of [3] and [19].

The instructions stated in [3], as shown in Figure 5.2, were executed, noticing that the `'pthreads.2'` folder should be placed at the same directory level, as the folder of the Kvazaar clone project. Visual Studio 2017 requires the external `'vsyasm.exe'`. Adjusting the build dependencies to `'vsyasm'` by right clicking on the `'kvazaar_lib'` in Visual Studio might be necessary. When these steps were implemented properly, the corresponding `'kvazaar_VS2013.sln'` can be executed and first the `'kvazaar_lib.lib'` file can be compiled. After successful compilation of the `lib` file, the executable can be generated by compiling the `'kvazaar_cli'`. The `'kvazaar.exe'` can afterwards be used in the OS shell by adding the corresponding environment variables in the system settings.

The next step is the Kvazaar HEVC encoding with tiles enabled. Since the encoder implementation can only process videos in the RAW `.yuv` format, the source footage needs to be converted into the RAW format. As sample footage, the equirectangular source clip, `06_Lombardsbruecke-02` recorded with the Insta 2 Pro is used. The transcoding into the `.yuv` format is done by using `ffmpeg` version 4.1.1. [5]. Since the tutorial describes the DASH setup for a 4K 30fps equirectangular stream, the 8K source-clip was downsampled into 3840x1920 30fps, at aspect ratio of the native clip 2:1. The `.yuv` video is then transcoded in Kvazaar, by the corresponding command line of Listing 5.2, for two different qualites, i.e. 1 Mbps and 6 Mbps.

[5] https://github.com/

(a)

(b)

(c)

Figure 5.1: a) Exported frame of equirectangular representation of recorded test sequence '06_Lombardsbruecke'. b) Excerpt and possible viewport of '06_Lombardsbruecke'. c) Screenshot of the stitching process with encoding parameters on the right band, and single lens fisheye representation. Content recorded with Insta360 Pro 2.0 camera and post-processed by Insta360 Stitcher software [8].

[redacted] commented on 20 Jul 2018

Compiling Kvazaar in Visual Studio2017

1. First need to download kvazaar source code and greatest source code
   Git clone https://github.com/ultravideo/kvazaar.git kvazaar
   Git clone https://github.com/ultravideo/greatest.git kvazaar//greatest

Copy the greatest source to the corresponding folder in kvazaar

2. Download pthreads-w32-2-9-1-release.zip to ftp://sourceware.org/pub/pthreads-win32
   Unzip into the Pre-built.2 folder
   Copy the three files in the include folder to *:\Program Files (x86)\Microsoft Visual Studio\2017\Enterprise\VC\Tools\MSVC\14.14.26428\include

Copy pthreadVC2.lib from the lib folder to build\kvazaar_lib. Note that it corresponds to the compiled version. 64-bit X64 32-bit X86

3. Then go to http://yasm.tortall.net/Download.html to download the corresponding version of vsyasm, Win32 VS2010 .zip or Win64 VS2010 .zip
   Decompression
   Put win32's vsyasm.exe in *:\Program Files (x86)\Microsoft Visual Studio\2017\Enterprise\VC\Tools\MSVC\14.14.26428\bin\Hostx86\x86
   Put Win64's vsyasm.exe in *:\Program Files (x86)\Microsoft Visual Studio\2017\Enterprise\VC\Tools\MSVC\14.14.26428\bin\Hostx86\x64
   Put vsyasm.props, vsyasm.targets, vsyasm.xml in C:\Program Files (x86)\MSBuild\Microsoft.Cpp\v4.0\V140\BuildCustomizations

4. After compiling, you need to copy pthreadVC2.dll from pthreads-w32-2-9-1-release\Pre-built.2\dll to kvazaar.exe directory.
   Need to pay attention to the corresponding version of the 64-bit X64 32-bit X86
   Finally run kvazaar.exe

Machine translation may be wrong

Figure 5.2: Step by step explanation of Kvazaar compiling in Visual Studio 2017, of [3].

```
~\Source>kvazaar -i 06_RAW_Lombardsbruecke-02_4K30fps2D_HEVC.yuv --input-
    res=3840x1920 --input-fps 30 -o 06_kva_Lombardsbruecke-02
    _4K30fps2D_HEVC_GPAC_1000kbps.hvc --bitrate 1000000 --period 30 --gop 8
    --no-open-gop --bipred -p 8 --mv-constraint frametilemargin --tiles 3x3
    -q 30 --slices tiles
```

Listing 5.2: Command for HEVC Kvazaar encoding, with tiles enabled, here for a bitrate of 5 Mbps.

The corresponding commands are defined as follows, with reference to the descriptions of [19].

- `-i, --input <filename>`: Input file in RAW yuv420p 8-bit format.
- `--input-res <res>`: Input resolution default is [auto], by detecting it from the file name. Otherwise defining the input resolution in `-<int>x<int>`: width times height pixels.
- `--o, --output <filename>`: Output file as `.hvc` format.
- `--tiles <int>x<int>`: Split picture into size of width x height [px] uniform tiles.
- `--slice<string>`: Defines how slices are used. Using the string `'tiles'` puts each desired tile in an independent slice.
- `--mv-constraint <string>`: Defines the motion vector constrains. The string `'frametilemargin'` defines the motion vector constrains corresponding to each tile, even stricter than the string `'frametile'`.
- `-q, --qp <integer>`: Defines the quantization parameter. When the `'bitrate'` is predefined as command, it will be omitted.
- `--bitrate <integer>`: Specifies the target bitrate. By default it is set to 0, so rate control is disabled. A following higher integer N defines the target N bits per second. E.g. 1 Mbps would be defined as `'--bitrate 1000000'`.
- `-p, --period <integer>`: Defines the period of following intra-pictures and is set to 64 by default. 0: Only first picture is intra. 1: Enables the all-intra mode and no inter-prediction is used. Using any other integer N: Every Nth picture is intra-coded.
- `-n, --frames <integer>`: Specifies the number of frames to code, if not the whole video needs to be coded. By default the whole video is encoded.
- `--input-fps <num>[/<denom>]`: Defines the framerate of the input video. By default it is set to 25 fps. In this case 30 fps is used.

Next, the two resulting output files, each of which as single `hvc` stream, need to be packed ISOBMFF-conformly into the MP4 container, by using MP4Box. After setting up the latest GPAC builds version 0.7.0 [7], the following command line of Listing 5.3 was used, according to the MP4Box documentation [10] and the adaptation guide [75], noticing that the log is also inserted in the listing, below the command.

```
mp4box -add 06_kva_Lombardsbruecke-02_4K30fps2D_HEVC_GPAC_1000kbps.hvc:
    split_tiles -fps 30 -new 06_mp4_Lombardsbruecke-02
    _4K30fps2D_HEVC_GPAC_1000kbps.mp4
HEVC import - frame size 3840 x 1920 at 30.000 FPS
HEVC Import results: 1840 samples (18404 NALUs) - Slices: 1845 I 1854 P
    12861 B - 1841 SEI - 1845 IDR
Stream uses forward prediction - stream CTS offset: 3 frames
Saving 06_mp4_Lombardsbruecke-02_4K30fps2D_HEVC_GPAC_1000kbps.mp4: 0.500
    secs Interleaving

mp4box -add 06_kva_Lombardsbruecke-02_4K30fps2D_HEVC_GPAC_6000kbps.hvc:
    split_tiles -fps 30 -new 06_mp4_Lombardsbruecke-02
    _4K30fps2D_HEVC_GPAC_6000kbps.mp4
HEVC import - frame size 3840 x 1920 at 30.000 FPS
HEVC Import results: 1840 samples (18404 NALUs) - Slices: 1845 I 1854 P
    12861 B - 1841 SEI - 1845 IDR
Stream uses forward prediction - stream CTS offset: 3 frames
Saving 06_mp4_Lombardsbruecke-02_4K30fps2D_HEVC_GPAC_6000kbps.mp4: 0.500
    secs Interleaving
```

Listing 5.3: Encapsulation of .hvc stream into ISOBMFF-conform format.

This leads to the desired number of tile-tracks as separate streams, of type `hvt1`, plus the base tile-track of type `hvc2/hev2` containing parameter sets and SEI messages. The first track corresponds to the base tile-track and the ensuing ISOBMFF tracks correspond to tile-tracks of tiles 1-9. The authors of [75] note, that removing the first tile-track leads to a corrupt file, albeit normally the single tile-tracks should be decodable independently of each other. Removing the second or any other tile-tracks by using the MP4Box command `'-rem <trackID>'` can be used to check whether tiling worked properly, resulting in entire green rectangular blocks at the corresponding locations. Besides the two different bitrates, all other encoding parameters, like resolution, tiling-grid, PPS and SPS should remain the same. For this purpose, MP4Box was used to create the corresponding DASH files out of the two created ISOBMFF packed HEVC streams, as follows in Listing 5.4:

```
MP4Box -dash 1000 -profile live -frag 1000 -rap -segment-name %s_segment -
    min-buffer 1000 -url-template -out dash_tiled-lombardsbruecke_1-6mbps.
    mpd 06_mp4_Lombardsbruecke-02_4K30fps2D_HEVC_GPAC_1000kbps.mp4 06
    _mp4_Lombardsbruecke-02_4K30fps2D_HEVC_GPAC_6000kbps.mp4
```

Listing 5.4: Creation of corresponding DASH MPD and segment files.

The command of Listing 5.4 results in one single MPD depicted in Listing 5.5. The MP4box `dash` commands are listed below, with reference to [4].

- `-dash <integer X>`: Creating segments of size X in milliseconds.
- `-profile <string>`: Defining the desired profile. In this case `'live'` is chosen.
- `-frag <integer X>`: Use movie fragments of roughly Y milliseconds. By default, fragments duration is 500 ms. Here the fragments should match the segment size, so it is correspondingly set to 1000 ms.
- `-rap`: Cutting segments to match access points, to not impair coding dependencies.

- `-segment-name <string NAME>`: Generating each segment in a dedicated file, called `NAME%d.EXT`. `%s` can be used to replace them by the name of each file being dashed (without file-extension).
- `-min-buffer`: Determines the minimum buffer time before playback.
- `-url-template`: The segments of different files will be referred to use the *Segment-Template* syntax in the MPD.
- `-out <String OUTPUT.mpd>`: Creates the corresponding MPD into the desired `OUTPUT` file.

```
<?xml version="1.0"?>
<!-- MPD file Generated with GPAC version 0.7.0-rev0-gbd5c9af-master at 2019-03-19T15:27:53.470Z-->
<MPD xmlns="urn:mpeg:dash:schema:mpd:2011" minBufferTime="PT1.000S" type="static" mediaPresentationDuration="PT0H1M1.333S" maxSegmentDuration="PT0H0M1.033S" profiles="urn:mpeg:dash:profile:isoff-live:2011">
 <ProgramInformation moreInformationURL="http://gpac.io">
  <Title>dash_tiled-lombardsbruecke_1-6mbps.mpd generated by GPAC</Title>
 </ProgramInformation>

 <Period duration="PT0H1M1.333S">
  <AdaptationSet segmentAlignment="true" bitstreamSwitching="true" maxWidth="3840" maxHeight="1920" maxFrameRate="30" par="2:1" lang="und">
   <EssentialProperty schemeIdUri="urn:mpeg:dash:srd:2014" value="1,0,0,0,0"/>
   <SegmentTemplate initialization="dash_tiled-lombardsbruecke_1-6mbps_set1_init.mp4"/>
   <Representation id="1" mimeType="video/mp4" codecs="hev2.1.6.L186.80" width="3840" height="1920" frameRate="30" sar="1:1" startWithSAP="1" bandwidth="9558">
    <SegmentTemplate timescale="30000" media="06_mp4_Lombardsbruecke-02_4K30fps2D_HEVC_GPAC_1000kbps_segment_track1_$Number$.m4s" startNumber="1" duration="30000"/>
   </Representation>
  </AdaptationSet>
  <AdaptationSet segmentAlignment="true" bitstreamSwitching="true" maxWidth="1280" maxHeight="640" maxFrameRate="30" par="2:1" lang="und">
   <SupplementalProperty schemeIdUri="urn:mpeg:dash:srd:2014" value="1,0,0,1280,640"/>
   <Representation id="1_2" mimeType="video/mp4" codecs="hvt1.1.6.L186.80" width="1280" height="640" frameRate="30" sar="1:1" startWithSAP="1" bandwidth="54815" dependencyId="1">
    <SegmentTemplate timescale="30000" media="06_mp4_Lombardsbruecke-02_4K30fps2D_HEVC_GPAC_1000kbps_segment_track2_$Number$.m4s" startNumber="1" duration="30000"/>
   </Representation>
   <Representation id="1_11" mimeType="video/mp4" codecs="hvt1.1.6.L186.80" width="1280" height="640" frameRate="30" sar="1:1" startWithSAP="1" bandwidth="101632" dependencyId="1">
    <SegmentTemplate timescale="30000" media="06_mp4_Lombardsbruecke-02_4K30fps2D_HEVC_GPAC_6000kbps_segment_track2_$Number$.m4s" startNumber="1" duration="30000"/>
   </Representation>
```

```
24  </AdaptationSet>
25  <AdaptationSet segmentAlignment="true" bitstreamSwitching="true" maxWidth
      ="1280" maxHeight="640" maxFrameRate="30" par="2:1" lang="und">
26   <SupplementalProperty schemeIdUri="urn:mpeg:dash:srd:2014" value="
      1,1280,0,1280,640"/>
27   <Representation id="1_3" mimeType="video/mp4" codecs="hvt1.1.6.L186.80"
      width="1280" height="640" frameRate="30" sar="1:1" startWithSAP="1"
      bandwidth="54859" dependencyId="1">
28    <SegmentTemplate timescale="30000" media="06_mp4_Lombardsbruecke-02
      _4K30fps2D_HEVC_GPAC_1000kbps_segment_track3_$Number$.m4s" startNumber="
      1" duration="30000"/>
29   </Representation>
30   <Representation id="1_12" mimeType="video/mp4" codecs="hvt1.1.6.L186.80"
       width="1280" height="640" frameRate="30" sar="1:1" startWithSAP="1"
      bandwidth="105010" dependencyId="1">
31    <SegmentTemplate timescale="30000" media="06_mp4_Lombardsbruecke-02
      _4K30fps2D_HEVC_GPAC_6000kbps_segment_track3_$Number$.m4s" startNumber="
      1" duration="30000"/>
32   </Representation>
33  </AdaptationSet>
34  <AdaptationSet segmentAlignment="true" bitstreamSwitching="true" maxWidth
      ="1280" maxHeight="640" maxFrameRate="30" par="2:1" lang="und">
35   <SupplementalProperty schemeIdUri="urn:mpeg:dash:srd:2014" value="
      1,2560,0,1280,640"/>
36   <Representation id="1_4" mimeType="video/mp4" codecs="hvt1.1.6.L186.80"
      width="1280" height="640" frameRate="30" sar="1:1" startWithSAP="1"
      bandwidth="52257" dependencyId="1">
37    <SegmentTemplate timescale="30000" media="06_mp4_Lombardsbruecke-02
      _4K30fps2D_HEVC_GPAC_1000kbps_segment_track4_$Number$.m4s" startNumber="
      1" duration="30000"/>
38   </Representation>
39   <Representation id="1_13" mimeType="video/mp4" codecs="hvt1.1.6.L186.80"
       width="1280" height="640" frameRate="30" sar="1:1" startWithSAP="1"
      bandwidth="91227" dependencyId="1">
40    <SegmentTemplate timescale="30000" media="06_mp4_Lombardsbruecke-02
      _4K30fps2D_HEVC_GPAC_6000kbps_segment_track4_$Number$.m4s" startNumber="
      1" duration="30000"/>
41   </Representation>
42  </AdaptationSet>
43  <AdaptationSet segmentAlignment="true" bitstreamSwitching="true" maxWidth
      ="1280" maxHeight="640" maxFrameRate="30" par="2:1" lang="und">
44   <SupplementalProperty schemeIdUri="urn:mpeg:dash:srd:2014" value="
      1,0,640,1280,640"/>
45   <Representation id="1_5" mimeType="video/mp4" codecs="hvt1.1.6.L186.80"
      width="1280" height="640" frameRate="30" sar="1:1" startWithSAP="1"
      bandwidth="133506" dependencyId="1">
46    <SegmentTemplate timescale="30000" media="06_mp4_Lombardsbruecke-02
      _4K30fps2D_HEVC_GPAC_1000kbps_segment_track5_$Number$.m4s" startNumber="
      1" duration="30000"/>
47   </Representation>
48   <Representation id="1_14" mimeType="video/mp4" codecs="hvt1.1.6.L186.80"
       width="1280" height="640" frameRate="30" sar="1:1" startWithSAP="1"
      bandwidth="888490" dependencyId="1">
49    <SegmentTemplate timescale="30000" media="06_mp4_Lombardsbruecke-02
      _4K30fps2D_HEVC_GPAC_6000kbps_segment_track5_$Number$.m4s" startNumber="
      1" duration="30000"/>
```

```
50      </Representation>
51    </AdaptationSet>
52
53    <AdaptationSet ...
54    </AdaptationSet>
55   -->
56    </Period>
57 </MPD>
```

Listing 5.5: Excerpt of exemplary MPD for generated adaptive HEVC tile-track DASH stream, created by using MP4Box, for base-track and tile-tracks 1 and 2.

Each tile-track is now packed into one Adaptation Set, with two possible quality representations, i.e. 1000 Kbps and 6000 Kbps. The size of each tile-track is 1280x640, for three columns and rows. Resulting in 3840x1920 pixels, i.e. three-times 1280 x three-times 640. The SRD is auto-generated by the MP4Box `-dash` command, to describe the position of each tile in the respective resulting spherical video-frame. It is specified for each Adaptation Set by:

```
1 <EssentialProperty schemeIdUri="urn:mpeg:dash:srd:2014" value="1,0,0,0,0"/>
```

Listing 5.6: SRD base-track specification in the MPD XML.

as an example for the base-tile-track. The SRD for the single tile-tracks of tiles 1-9 are set by the following command, exemplarily for track 2:

```
1 <SupplementalProperty schemeIdUri="urn:mpeg:dash:srd:2014" value="
    1,0,0,1280,640"/>
```

Listing 5.7: SRD specification of tile-track 1 in the MPD XML.

The values are set correspondingly for each tile-track position in the resulting frame. As stated in the documentation, the playback can be done by the GPAC OSMO4 player [12], either by the GUI or by using the corresponding command of Listing 5.8.

```
1 ~\Source>$ mp4client http://akamai-progressive.irt.de/
    masterarbeit_testfiles/Nokiatech_Gpac/Gpac_eval/DASH01/dash_tiled-
    lombardsbruecke_1-6mbps.mpd#LIVE360
```

Listing 5.8: Starting the GPAC DASH stream, for test with OSMO4 player in 360° mode.

The DASH MPD was successfully created, with each tile-track nested as one single Representation. Albeit the playback in the OSMO4 player only works partially and does not interpret the MPD and its segments reliably, the architecture of the MPD is logical and well structured. When accessing the MPD on the server, the resulting stream in the OSMO4 player seems to be assembled by different tile-tracks, i.e. bitrate streams. This can be somewhat visualized by the player properties of Figure 5.3. The two different tile-tracks, i.e. Representations, with a certain position specified by the SRD, can be switched. In this case the two possible quality representations or 'auto' can be set. Noticing, that in the case of more streams of multiple bitrates, these would be listed accordingly in the properties of the player. This leads to the assumption that the MPD is created properly, and could be used adaptively by a proper player implementation. The GPAC OSMO4 player does not seem to operate reliably and support viewport adaptivity. Even when playing the provided test MPD, with tiles enabled, linked at [75], playback artifacts occurred.

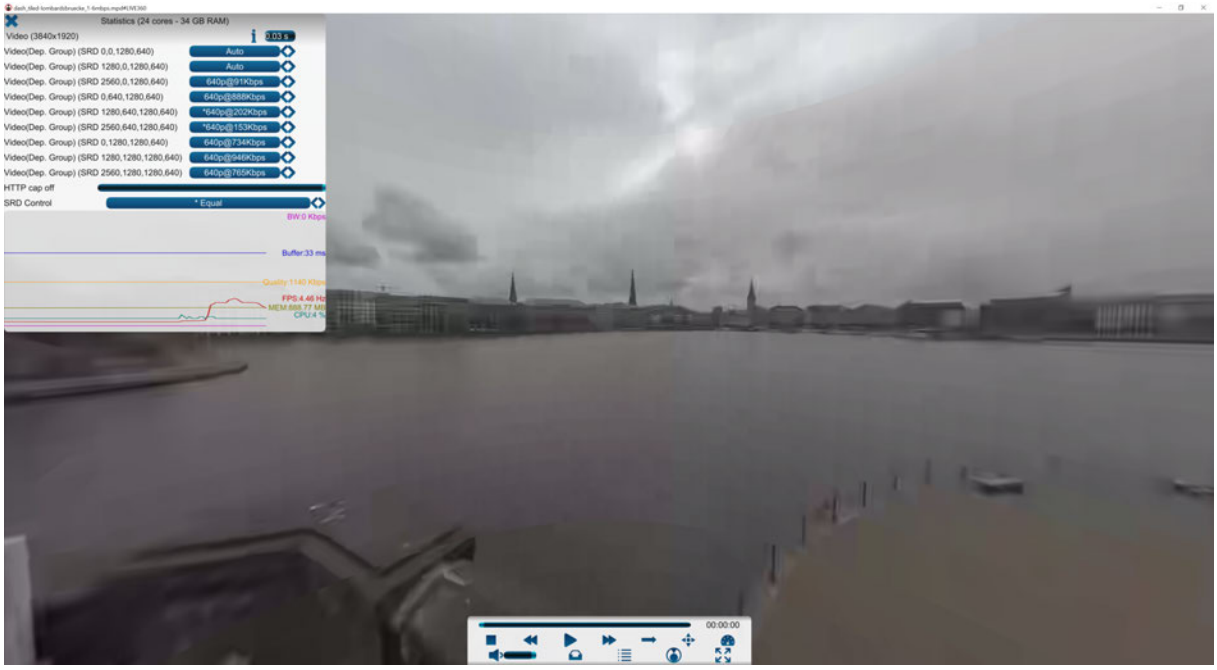


Figure 5.3: OSMO4 player playback of corresponding MPD, with configuration window and manual bitrate adjustments of single tracks. Bitrate of stream 1 is 1000 Kbps, stream 2 is 6000 Kbps.

In OSMO4 playback of the `dash_tiled-lombardsbruecke_1-6mbps.mpd` the chosen bitrate of each tile always corresponds to the lowest available bandwidth of the tile-tracks in the MPD. By changing the bitrate manually in the player properties to the highest bitrate for each tile-track, as visible in Figure 5.3, the image quality can be enhanced, although it is still not viewport-adaptive. It should be noted, that the resulting playback frame depicted in Figure 5.3 is not the refreshed playback frame, according to the manually adjusted bitrate for each track, but illustrates the player configurations. This appears to be an issue of the OSMO4 player, that chooses the lowest available bitrate by default. This issue was stated in forums of GPAC and does not seem to be fixed yet, according to the current state of research. It is assumed, that the available bandwidth is calculated wrong. This problem is stated in [11]. However, the GPAC OSMO4 player can be considered as test environment for research purposes, but may not meant to be an adaptive player solution. For detailed information on this issue and a possible browser integration of the GPAC solution, the reader is referred to the work of [85], where a similar example of the GPAC solution was examined and further integrated into a browser playback environment.

5.3.2 Nokiatech OMAF implementation

Another solution of HEVC tiled viewport-adaptive omnidirectional video streaming, is the OMAF implementation of Nokia Technologies [108], hereinafter referred to as *Nokiatech*. Three different OMAF creation modes are supported for the viewport-dependent profile, in this OMAF implementation. In the following, the test of a 360° OMAF HEVC viewport-dependent DASH stream is documented. In the implementation the following three viewport dependent modes are supported: Equal-resolution streams with extractor;

Effective 5K ERP; Effective 6K ERP all defined in the annex of the OMAF standard [51]. In this work, the first mode, equal-resolution streams with extractor tracks, was chosen for evaluation.

First, the provided Github resources [108], need to be compiled to build an executable OMAF generator and player. For compilation, additionally the *heif-Git* [110] and *libdash-Git* [20][21] are supplied into the same root-location as the Nokiatech OMAF-Git. It should be noted, that the *cmake*[6] library needs to be installed on the system, when using Windows10. Moreover the HEVC video extensions of Windows10 for GPU and CPU accelerated HEVC playback need to be installed, as its decoders are used. These can be obtained e.g. by the Microsoft Store. When all resources are prepared, the `'build-visualstudio.bat vs2017'` command was executed to build the implementation, located at `/omaf/build`. Afterwards, the respective executable files are located at the `'bin'` folder of the directory `'Creator'`. The OMAF player, named `'Monitor_Sample'` of Nokiatech needs to be compiled by Visual Studio. The corresponding Visual Studio 2017 solution file is located at `'omaf\Player\VideoPlayback\Windows\Monitor_Sample\Monitor_Sample.sln'`. When all the executables are built successfully, the environment-variables were defined for easier handling.

To create a DASH stream, first two versions of the source test video are were encoded at two different bitrates, with identical resolutions and encoding parameters. For encoding, as recommended, the HEVC Kvazaar encoder was used. The corresponding command lines and chosen parameters are depicted in Listing 5.9 below.

```
kvazaar -i 06_RAW_Lombardsbruecke-02_4K30fps2D_HEVC.yuv --input-res=3840
    x1920 --input-fps 30 --bitrate 500000 --gop 8 --no-open-gop --bipred -p
    8 --mv-constraint frametilemargin --tiles 5x2 --set-qp-in-cu --slices
    tiles -o 06_kva_Lombardsbruecke-02_4K30fps2D_HEVC_nokiatech_500kbps.265

kvazaar -i 06_RAW_Lombardsbruecke-02_4K30fps2D_HEVC.yuv --input-res=3840
    x1920 --input-fps 30 --bitrate 5000000 --gop 8 --no-open-gop --bipred -p
     8 --mv-constraint frametilemargin --tiles 5x2 --set-qp-in-cu --slices
    tiles -o 06_kva_Lombardsbruecke-02_4K30fps2D_HEVC_nokiatech_5000kbps.265
```

Listing 5.9: Kvazaar HEVC encoding at 500 Kbps and 5000 Kbps for Nokiatech OMAF preparation.

After successful encoding of the two .265 Kvazaar files, the MP4 packaging was done with MP4Box, as shown in Listing 5.10, for both files, i.e. bitrates.

```
~/360stream/nokia-omaf/source
mp4box -add 06_kva_Lombardsbruecke-02_4K30fps2D_HEVC_nokiatech_500kbps.265
    -fps 30 -new 06_packed_Lombardsbruecke-02
    _4K30fps2D_HEVC_nokiatech_500kbps.mp4

HEVC import - frame size 3840 x 1920 at 30.000 FPS
HEVC Import results: 1840 samples (20244 NALUs) - Slices: 2050 I 2060 P
    14290 B - 1841 SEI - 2050 IDR
Stream uses forward prediction - stream CTS offset: 3 frames
Saving 06_packed_Lombardsbruecke-02_4K30fps2D_HEVC_nokiatech_500kbps.mp4:
    0.500 secs Interleaving
```

Listing 5.10: Packaging of the Kvazaar files into the MP4 container.

[6] https://cmake.org/

The two generated files have a 2x5 tiling-grid and are encoded at 4K resoultion, 30 fps, with 0.5 Mbps and 5 Mbps for the non-viewport and viewport areas respectively. To create an OMAF-conform DASH directory, with segments, subsegments, initalization-file and MPD, the `'omafvd'` application can be used. This uses a `JSON` script as input for specification of the DASH parameters and file indexing. The code was named `'config.json'`, as proposed by [109]. Hence, the command is simply `'omafvd config.json'`. The `config` file is depicted in Listing 5.11. The `'omafvd.exe'` expects the files referred in the `JSON` file to be placed at the same folder. The example `JSON` file, provided by [108], was modified and adjusted to the desired DASH configuration.

```
{
    // "video" is obligatory
    "video": {
    "common" : {
            "projection" : "equirectangular", // or 'cubemap'; currently only the default OMAF cubemap format is supported
            "output_mode" : "MultiQ"          // Mode of VD: MultiQ == single resolution, multiple bitrates, "5K" == 5K unequal resolution (Annex D.6.2/example 2), "6K" == 6K unequal resolution (Annex D.6.3)
        },
        "bg": {
            "filename": "06_packed_Lombardsbruecke-02_4K30fps2D_HEVC_nokiatech_500kbps.mp4", //non viewport-tiles at 500 Kbps (bg = background)
            "quality": 50   // 1...255 where 1 is the best; if not given, defaults to 1
        },
        "fg": {
            "filename": "06_packed_Lombardsbruecke-02_4K30fps2D_HEVC_nokiatech_5000kbps.mp4", //viewport tiles at 5000 Kbps (fg = foreground)
            "quality": 1
    }
    // add as many as needed
    },
    // "dash" is alternative to "mp4". Creates all tracks separately in segments (for all inputs) and one for the extractor track
    // if both are given, only dash output is generated
    "dash": {
        "output_name_base" : "output_tiled-lombardsbruecke-02_500-5000kbps",     // Basename for the DASH output files
        "mpd": {
            "filename": "$Name$.mpd"  // optional, default is $Name$.mpd. $Name$ expands to the value of "output_name_base" or if it is not given, to "output_mode"
        },
        "media": {
            "subsegments_per_segment": 1, // optional, defaults to 1. If more than 1, 1 GOP is mapped to a subsegment, otherwise 1 GOP is mapped to a segment
            "segment_name": {
                // $Name$ expands to the value of "output_name_base" or if it is not given, to "output_mode"
                // $Segment$ expands to MPD's "$Number$" or "init" depending on the segment type. Using MPD's $Number$ directly here is not
```

```
    allowed.
                "video":          "$Name$.video.$Segment$.mp4",
                "audio":          "$Name$.audio.$Segment$.mp4",
                "extractor":      "$Name$.extractor.$Segment$.mp4"
            }
        }
    },
    // "mp4" is alternative to "dash". Creates a single mp4 with all tracks (one per tile) and the extractor track
    //"mp4": {
    //    "filename": "06_OMAF_Lombardsbruecke-02_4K30fps2D_HEVC_nokiatech_5000kbps.mp4"
    }

}
```

Listing 5.11: Parameters in Config.json for creation of the Nokiatech DASH directory, based on the example of [108].

An extract of the resulting DASH MPD is depicted below, in Listing 5.12. The first Adaptation Set, with `ID=1` contains the two Representations of the first tile-track. All following Adaptation Sets, contain the following two tile-tracks, each of which lasting for `'duration=9'`. One of the Representations, i.e. tile-tracks in each Adaptation Set is of high bitrate, i.e. `'fg'`, the other of lower quality, i.e. `'bg'`. Hence, the tile-track ID corresponds to the ID of the Adaptation Set. Supplemental OMAF-specific information are included in `'SupplementalProperty'` referring to the `'srqr'` MPEG URN. Further information about the quality ranking, and content coverage are included in the `'omaf:sphRegionQuality'` and `'omaf:cc shape_type="0"'` entries, respectively. The region-wise packing type and the projection are defined by `'omaf:packing_type="0"'` and `'omaf:projection_type="0"'`. As expected, projection type 0 is used, that refers to equirectangular projection. Although in the Adaptation Set entries the specified projection mapping, signalled by the SEI, refers to `'ercm'`, i.e cubemap projection, by `'codecs="resv.podv+ercm.hvc2.1.6.L153.80"'`. In the example `JSON` file, for OMAF configuration, equirectangular is predetermined. However, the last Adaptation Set with `'ID=11'` contains the extractor track, as one single Representation at full resolution.

```
<?xml version="1.0" encoding="utf-8"?>
<MPD xmlns:omaf="urn:mpeg:mpegI:omaf:2017"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns="urn:mpeg:dash:schema:mpd:2011"
     xmlns:xlink="http://www.w3.org/1999/xlink"
     xsi:schemaLocation="urn:mpeg:dash:schema:mpd:2011 http://standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-DASH_schema_files/DASH-MPD.xsd"
     minBufferTime="PT3S"
     type="static"
     mediaPresentationDuration="PT1M1.197S"
     profiles="urn:mpeg:dash:profile:isoff-live:2011">
  <EssentialProperty schemeIdUri="urn:mpeg:mpegI:omaf:2017:pf" omaf:projection_type="0" />
  <Period duration="PT1M1.197S">
    <AdaptationSet id="1" mimeType="video/mp4 profiles='hevd'" codecs="resv.podv+ercm.hvc1.1.6.L186.80" maxWidth="768" maxHeight="960" maxFramerate="30" segmentAlignment="1">
```

```
14      <EssentialProperty schemeIdUri="urn:mpeg:mpegI:omaf:2017:rwpk" omaf:
    packing_type="0" />
15      <SupplementalProperty schemeIdUri="urn:mpeg:mpegI:omaf:2017:cc">
16        <omaf:cc shape_type="1" default_view_idc="0">
17          <omaf:coverageInfo centre_azimuth="9437184" centre_elevation="
    2949120" centre_tilt="0" azimuth_range="4718592" elevation_range="
    5898240" />
18        </omaf:cc>
19      </SupplementalProperty>
20      <Representation id="output_tiled-lombardsbruecke-02_500-5000kbps.bg.
    tile1.video" bandwidth="50980" width="768" height="960" frameRate="30"
    startWithSAP="1">
21        <SegmentTemplate media="output_tiled-lombardsbruecke-02_500-5000
    kbps.bg.tile1.video.$Number$.mp4"
22                         initialization="output_tiled-lombardsbruecke-02
    _500-5000kbps.bg.tile1.video.init.mp4"
23                         duration="9"
24                         startNumber="1"
25                         timescale="30" />
26        <SupplementalProperty schemeIdUri="urn:mpeg:mpegI:omaf:2017:srqr">
27          <omaf:sphRegionQuality shape_type="1" remaining_area_flag="false"
     quality_ranking_local_flag="false" quality_type="0" default_view_idc="0
    ">
28            <omaf:qualityInfo quality_ranking="50" centre_azimuth="9437184"
     centre_elevation="2949120" centre_tilt="0" azimuth_range="4718592"
    elevation_range="5898240" />
29          </omaf:sphRegionQuality>
30        </SupplementalProperty>
31      </Representation>
32      <Representation id="output_tiled-lombardsbruecke-02_500-5000kbps.fg.
    tile1.video" bandwidth="507850" width="768" height="960" frameRate="30"
    startWithSAP="1">
33        <SegmentTemplate media="output_tiled-lombardsbruecke-02_500-5000
    kbps.fg.tile1.video.$Number$.mp4"
34                         initialization="output_tiled-lombardsbruecke-02
    _500-5000kbps.fg.tile1.video.init.mp4"
35                         duration="9"
36                         startNumber="1"
37                         timescale="30" />
38        <SupplementalProperty schemeIdUri="urn:mpeg:mpegI:omaf:2017:srqr">
39          <omaf:sphRegionQuality shape_type="1" remaining_area_flag="false"
     quality_ranking_local_flag="false" quality_type="0" default_view_idc="0
    ">
40            <omaf:qualityInfo quality_ranking="1" centre_azimuth="9437184"
    centre_elevation="2949120" centre_tilt="0" azimuth_range="4718592"
    elevation_range="5898240" />
41          </omaf:sphRegionQuality>
42        </SupplementalProperty>
43      </Representation>
44    </AdaptationSet>
45    <AdaptationSet id="2" mimeType="video/mp4 profiles='hevd'" codecs="resv
    .podv+ercm.hvc1.1.6.L186.80" maxWidth="768" maxHeight="960" maxFramerate
    ="30" segmentAlignment="1">
46      <EssentialProperty schemeIdUri="urn:mpeg:mpegI:omaf:2017:rwpk" omaf:
    packing_type="0" />
47      <SupplementalProperty schemeIdUri="urn:mpeg:mpegI:omaf:2017:cc">
```

```
48        <omaf:cc shape_type="1" default_view_idc="0">
49          <omaf:coverageInfo centre_azimuth="4718592" centre_elevation="
    2949120" centre_tilt="0" azimuth_range="4718592" elevation_range="
    5898240"/>
50        </omaf:cc>
51      </SupplementalProperty>
52      <Representation id="output_tiled-lombardsbruecke-02_500-5000kbps.bg.
    tile2.video" bandwidth="50980" width="768" height="960" frameRate="30"
    startWithSAP="1">
53        <SegmentTemplate media="output_tiled-lombardsbruecke-02_500-5000
    kbps.bg.tile2.video.$Number$.mp4"
54                         initialization="output_tiled-lombardsbruecke-02
    _500-5000kbps.bg.tile2.video.init.mp4"
55                         duration="9"
56                         startNumber="1"
57                         timescale="30"/>
58        <SupplementalProperty schemeIdUri="urn:mpeg:mpegI:omaf:2017:srqr">
59          <omaf:sphRegionQuality shape_type="1" remaining_area_flag="false"
     quality_ranking_local_flag="false" quality_type="0" default_view_idc="0
    ">
60            <omaf:qualityInfo quality_ranking="50" centre_azimuth="4718592"
     centre_elevation="2949120" centre_tilt="0" azimuth_range="4718592"
    elevation_range="5898240"/>
61          </omaf:sphRegionQuality>
62        </SupplementalProperty>
63      </Representation>
64      <Representation id="output_tiled-lombardsbruecke-02_500-5000kbps.fg.
    tile2.video" bandwidth="507850" width="768" height="960" frameRate="30"
    startWithSAP="1">
65        <SegmentTemplate media="output_tiled-lombardsbruecke-02_500-5000
    kbps.fg.tile2.video.$Number$.mp4"
66                         initialization="output_tiled-lombardsbruecke-02
    _500-5000kbps.fg.tile2.video.init.mp4"
67                         duration="9"
68                         startNumber="1"
69                         timescale="30"/>
70        <SupplementalProperty schemeIdUri="urn:mpeg:mpegI:omaf:2017:srqr">
71          <omaf:sphRegionQuality shape_type="1" remaining_area_flag="false"
     quality_ranking_local_flag="false" quality_type="0" default_view_idc="0
    ">
72            <omaf:qualityInfo quality_ranking="1" centre_azimuth="4718592"
    centre_elevation="2949120" centre_tilt="0" azimuth_range="4718592"
    elevation_range="5898240"/>
73          </omaf:sphRegionQuality>
74        </SupplementalProperty>
75      </Representation>
76    </AdaptationSet>
77
78    <!-- More Adaptation Sets following, corresponding to the tiling-grid
    -->
79
80    <AdaptationSet id="11" mimeType="video/mp4 profiles='hevd'" codecs="
    resv.podv+ercm.hvc2.1.6.L186.80" maxWidth="3840" maxHeight="1920"
    maxFramerate="30" segmentAlignment="1">
81      <EssentialProperty schemeIdUri="urn:mpeg:mpegI:omaf:2017:rwpk" omaf:
    packing_type="0"/>
```

```
82      <SupplementalProperty schemeIdUri="urn:mpeg:dash:preselection:2016"
    value="ext11,11 1 2 3 4 5 6 7 8 9 10"/>
83      <Representation id="output_tiled-lombardsbruecke-02_500-5000kbps.
    extractor" bandwidth="0" width="3840" height="1920" frameRate="30"
    startWithSAP="1">
84          <SegmentTemplate media="output_tiled-lombardsbruecke-02_500-5000
    kbps.extractor.$Number$.mp4" initialization="output_tiled-
    lombardsbruecke-02_500-5000kbps.extractor.init.mp4" duration="9"
    startNumber="1" timescale="30"/>
85      </Representation>
86    </AdaptationSet>
87  </Period>
88 </MPD>
```

Listing 5.12: Excerpt of exemplary MPD for generated Nokiatech OMAF DASH stream, generated by use of *omafvd* of [108].

The viewport-dependent DASH stream of multiple bitrates and the single locally-stored OMAF-conform MP4 file, can be played by the SDK `Monitor_Sample.exe` executing the respective commands of Listing 5.13 and 5.14 for Windows, or in Android by using Android Studio. Here, only the Windows playback has been tested.

```
1 monitor_sample storage://06_OMAF_Lombardsbruecke-02
    _4K30fps2D_HEVC_nokiatech_5000kbps.mp4
```

Listing 5.13: Command for playback of the OMAF files, by the Nokiatech sample player [109].

```
1 monitor_sample http://akamai-progressive.irt.de/masterarbeit_testfiles/
    Nokiatech_Gpac/Nokiatech_eval/DASH02/DASH02/output_tiled-lombardsbruecke
    -02_500-5000kbps.mpd
```

Listing 5.14: Command for playback of the DASH stream, by the Nokiatech sample player [109].

With the chosen OMAF profile, tiling grid, resolution, framerate and number of quality representations, i.e. the coding command lines mentioned above, the playback by the `Monitor_sample` player only works partially. Playback artifacts occur, assumed to be caused by wrong color component interpretation and encoding parameters, especially when using the viewport adaptive profile, with more than one quality representation. Notwithstanding, that this could have been caused by the self-configured Kvazaar HEVC encoding. For this purpose single bitrate and unadaptive DASH streams have also been tested, which were interpreted properly by the sample player. Since the DASH MPD has the tiling and segmentation structure, determined by the OMAF standard, it can be assumed, that another player implementation might be able to interpret the created DASH stream correctly. Since the source `.yuv`, the created OMAF-conform single files and the unadaptive streams can be decoded without artifacts, as depicted in Figure 5.4 it is assumed, that assembling the stream by different tile-tracks is a tricky task for the player. Although it is evident, at this point it should be noted, that this implementation is not intended to be a final version, but more a pre-version and test implementation at development state of the OMAF standard.

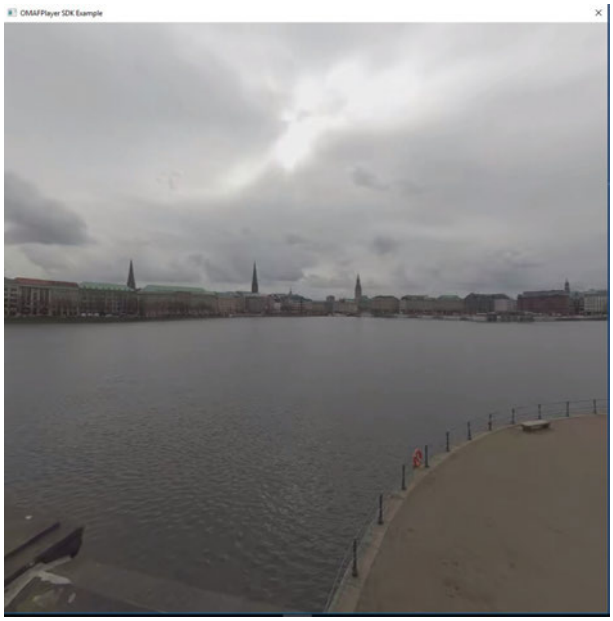


Figure 5.4: Screenshot of playback of viewport independent DASH MPD, created by the Nokiatech OMAF implementation [108] and Kvazaar HEVC encoder [19].

5.3.3 Fraunhofer HHI OMAF implementation

The *Fraunhofer Heinrich Herz Institut* (HHI) is developing a MPEG-OMAF implementation, as stated by the developers Podborski et al. in [90]. The current version was roughly tested thanks to a cooperation between the IRT and HHI. The resources were obtained by the corresponding Gitlab of the HHI and contain a JavaScript player, as HTML web page with a MPD URL input field and additional tools for content creation. Since the provided pre-built content creation tools are for Linux and MacOS, in this work, they were executed on a Virtual Box Ubuntu 18.04. These tools expect RAW `.yuv` ERP source video files as input to create a cubemap-projected viewport-dependent OMAF-conform DASH stream and single OMAF-conform files. The HEVC encoder implementation is the latest HM encoder [18]. In this version the tiling grid is set to 24 tiles for each transmitted stream, 12 in high-res, for the viewport, and 12 in low-res, for the rest. The high resolution tiles have a size of 768x768 pixels, the low resolution tiles are of 384x384 pixels. Correspondingly, the packed frame, has a resolution of 3840x2304 pixels. The tile-tracks are generated for each tile, resulting in 24 tile-tracks, so the assembled stream the client receives is divided into 12 low-res and 12-high-res tile-tracks, constructing a total of 48 triangular surfaces that form a 3D cube.

The content creation is based on the allocated programs `ffmpeg`, `hevc2omaf`, `TAPP360convert`, and `TAPPencoder`, to process the input `.yuv` file and is divided into five main execution steps:

- Step 1: Conversion of `.yuv` file from ERP into cubemap of highest available resolution.
- Step 2: Downscaling the high-res cubemap `.yuv` and creation of the low-res cubemap `.yuv` RAW files.

- Step 3: Split both, i.e. high- and low-res files, into 24 tiles (each), to create all required tiles.
- Step 4: Run HM and encode each tile as MCTS for chosen QPs.
- Step 5: Packaging of encoded HEVC bitstreams into OMAF DASH files and OMAF-conform single HEVC files.

For creation of the OMAF files, after installing Python version 2.7, the provided Python script and a configuration file for specification of the HM encoder parameters were used. The Python script handles the following input parameters, with reference to the script file of [90]:

- `'-s', '--steps': <string> '1-5'=` all; `'1'` = ERP to cubemap conversion only; `'5'` = Only packaging; `'3-5'`= First tiling, then encoding and packaging.
- `'-i', '--input': 'input filename' as <string>.`
- `'-o', '--OutputDir': <string>, default = 'out'.` Name of output folder, located at the same root as .py script.
- `'-p', '--FilePrefix': <string>.`
- `'--InputBitDepth': <int>, default=8.`
- `'-icf', '--InputChromaFormat': <int>, default=420.`
- `'-wdt', '--SourceWidth': <int>, default=8192.`
- `'-hgt', '--SourceHeight': <int>, default=4096.`
- `'-f', '--FramesToBeEncoded': <int>, default=-1.`
- `'-fr', '--FrameRate': <int>, default=30.`
- `'-q', '--QP': <int>, default=[32], nargs='+'.` Quantization parameter for quality adjustment.
- `'-c', '--HMconfig': <string> 'filename of the .cfg file'` for encoder parameters.
- `'-t', '--NumThreads': <int>, default=4.` Number of parallel processes.
- `'-gbs', '--GuardBandSize': <int>, default=0.` Size of OMAF guard band.
- `'-gbm', '--GuardBandMode': <string> 'smear',`
  `<string> 'mirror'>, default='smear'.` Mode of OMAF guard band.
- `'--hhienc',` HHI HEVC encoder instead of HM.

For evaluation, 6K RAW content was be used as input, to diminish the encoding times of the HM encoder. Hence, the source `.yuv` file was downsampled using `ffmpeg` accordingly, as shown in Listing 5.15. It should be noted, that the HHI OMAF creation tool expects the color component format to be yuv420p by default. Other pixel formats such as yuvj420p etc. can also be used, which has to be parametrized as `'-icf` input. As some color component artifacts occurred at the first encodings, `ffmpeg` was used to transform the yuvj420p pixel format into yuv420p, so no input color format had to be specified for the OMAF creation tools.

```
ffmpeg -s:v 7680x3840 -r 60 -pix_fmt yuvj420p -i 06_RAW_Lombardsbruecke-02
    _8K60fps2D_HEVC.yuv -vf scale=6144x3072 -c:v rawvideo -pix_fmt yuv420p
    06_RAW_Lombardsbruecke-02_6k60fps_yuv420p_new.yuv
```

Listing 5.15: FFMPEG downsampling into 6K RAW YUV format, for processing with the HHI content creation tools.

Afterwards, using the downsampled 6144x3072, 60 fps `.yuv` source clip, the following Python script was executed in Ubuntu to generate the OMAF DASH stream, as shown in Listing 5.16.

```
./create_omaf_files.py -i 06_RAW_Lombardsbruecke-02_6k60fps_yuv420p_new.yuv
    -s 1-5 -wdt 6144 -hgt 3072 -fr 60 -f 600 -q 32 -t 8 -o out-tiled-4 -c
    encoder_config_lombards.cfg
```

Listing 5.16: Execution of Python script, provided by [90], for content creation of OMAF DASH stream.

Since the HM encoder was used, with many complex HEVC features enabled, the encoding is very slow. For that reason, not the entire clip, but only 600 frames were encoded. The output of the HHI OMAF creation tool, contains two DASH folders, for two profiles, live and VOD, respectively. Each of which containing a MPD, and segments at two quality levels, as depicted in Listing 5.17. The corresponding tile-tracks are also created and located in subfolders, for the two resolutions, i.e. 384x384 and 768x768. Additionally one single OMAF-conform HEVC coded ISOBMFF file, encoded with the chosen QP, and the entire high-res and low-res RAW files, are created and separately stored. The MPD XML of Listing 5.17 contains the 48 Adaptation Sets, each of which containing one Representation, for each single tile-track. The HEVC codec and cubemap projection is specified in `'codecs="resv.podv+ercm.hvc1.2.L153.B0"'`. A SRD is not used in this case, as the position of each tile is specified by a `'coverageInfo'`. The OMAF brands discussed in Section 4.1.5 can be identified in this MPD. The 48 Adaptation Sets for each tile-track, are followed by 24 additional Adaptation Sets, each containing one Representation for one particular extractor track, assembling one particular viewport.

```
<?xml version="1.0" encoding="utf-8"?>
<MPD xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="urn:mpeg:
    dash:schema:mpd:2011" xmlns:omaf="urn:mpeg:mpegI:omaf:2017" xmlns:xlink=
    "http://www.w3.org/1999/xlink" xsi:schemaLocation="urn:mpeg:dash:schema:
    mpd:2011 http://standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-
    DASH_schema_files/DASH-MPD.xsd" type="static" profiles="urn:mpeg:dash:
    profile:isoff-live:2011" mediaPresentationDuration="PT4.95S"
    minBufferTime="PT1S">
  <EssentialProperty schemeIdUri="urn:mpeg:mpegI:omaf:2017:pf" omaf:
    projection_type="1"/>
  <Period duration="PT4.95S">
```

```
<AdaptationSet mimeType="video/mp4 profiles='hevd'" codecs="resv.podv+ercm.hvc1.2.L153.B0" segmentAlignment="1" subsegmentAlignment="1" id="1">
  <Viewpoint schemeIdUri="urn:mpeg:dash:viewpoint:2011" value="vp1"/>
  <EssentialProperty schemeIdUri="urn:mpeg:mpegI:omaf:2017:rwpk"/>
  <SupplementalProperty schemeIdUri="urn:mpeg:mpegI:omaf:2017:cc">
    <omaf:cc shape_type="0">
      <omaf:coverageInfo centre_azimuth="7602176" centre_elevation="1572864" azimuth_range="1310720" elevation_range="1310720"/>
    </omaf:cc>
  </SupplementalProperty>
  <Representation id="1_qp32" bandwidth="112121" qualityRanking="1" frameRate="60">
    <SegmentTemplate timescale="90000" duration="13500" startNumber="1" initialization="live/qp32/06RAWLombardsbruecke-026k60fpsyuv420pnew_out_384x384_qp32_seg0.mp4" media="live/qp32/Track_1_Seg_$Number$.mp4"/>
  </Representation>
</AdaptationSet>
<AdaptationSet mimeType="video/mp4 profiles='hevd'" codecs="resv.podv+ercm.hvc1.2.L153.B0" segmentAlignment="1" subsegmentAlignment="1" id="2">
  <Viewpoint schemeIdUri="urn:mpeg:dash:viewpoint:2011" value="vp1"/>
  <EssentialProperty schemeIdUri="urn:mpeg:mpegI:omaf:2017:rwpk"/>
  <SupplementalProperty schemeIdUri="urn:mpeg:mpegI:omaf:2017:cc">
    <omaf:cc shape_type="0">
      <omaf:coverageInfo centre_azimuth="4128768" centre_elevation="1572864" azimuth_range="1310720" elevation_range="1310720"/>
    </omaf:cc>
  </SupplementalProperty>
  <Representation id="2_qp32" bandwidth="139558" qualityRanking="1" frameRate="60">
    <SegmentTemplate timescale="90000" duration="13500" startNumber="1" initialization="live/qp32/06RAWLombardsbruecke-026k60fpsyuv420pnew_out_384x384_qp32_seg1.mp4" media="live/qp32/Track_2_Seg_$Number$.mp4"/>
  </Representation>
</AdaptationSet>

    <!-- 46 More Adaptation Sets following, corresponding to the tiling-grid of 24 tiles per resolution -->

  </AdaptationSet>
<AdaptationSet mimeType="video/mp4 profiles='hevd'" codecs="resv.podv+ercm.hvc2.2.L153.B0" segmentAlignment="1" subsegmentAlignment="1" id="49">
  <Viewpoint schemeIdUri="urn:mpeg:dash:viewpoint:2011" value="vp1"/>
  <EssentialProperty schemeIdUri="urn:mpeg:mpegI:omaf:2017:rwpk"/>
  <SupplementalProperty schemeIdUri="urn:mpeg:mpegI:omaf:2017:srqr">
    <omaf:sphRegionQuality quality_type="1" shape_type="0" remaining_area_flag="1" quality_ranking_local_flag="1">
      <omaf:qualityInfo quality_ranking="1" orig_width="4608" orig_height="3072" centre_azimuth="7602176" centre_elevation="1572864" azimuth_range="6553600" elevation_range="6553600"/>
      <omaf:qualityInfo quality_ranking="2" orig_width="2304" orig_height="1532"/>
```

```
40          </omaf:sphRegionQuality>
41        </SupplementalProperty>
42        <SupplementalProperty schemeIdUri="urn:mpeg:dash:preselection:2016"
    value="PreselTag1,49 25 38 31 26 44 32 45 39 41 47 42 37 3 4 5 6 9 10 11
     12 16 19 22 24 "/>
43        <Representation id="49" bandwidth="356906" frameRate="60">
44          <SegmentTemplate timescale="90000" duration="13500" startNumber="1"
     initialization="live/06RAWLombardsbruecke-026
    k60fpsyuv420pnew_out_Extractors0.mp4" media="live/Ext_1_Seg_$Number$.mp4
    "/>
45        </Representation>
46      </AdaptationSet>
47      <AdaptationSet mimeType="video/mp4 profiles='hevd'" codecs="resv.podv+
    ercm.hvc2.2.L153.B0" segmentAlignment="1" subsegmentAlignment="1" id="50
    ">
48        <Viewpoint schemeIdUri="urn:mpeg:dash:viewpoint:2011" value="vp1"/>
49        <EssentialProperty schemeIdUri="urn:mpeg:mpegI:omaf:2017:rwpk"/>
50        <SupplementalProperty schemeIdUri="urn:mpeg:mpegI:omaf:2017:srqr">
51          <omaf:sphRegionQuality quality_type="1" shape_type="0"
    remaining_area_flag="1" quality_ranking_local_flag="1">
52            <omaf:qualityInfo quality_ranking="1" orig_width="4608"
    orig_height="3072" centre_azimuth="4128768" centre_elevation="1572864"
    azimuth_range="6553600" elevation_range="6553600"/>
53            <omaf:qualityInfo quality_ranking="2" orig_width="2304"
    orig_height="1532"/>
54          </omaf:sphRegionQuality>
55        </SupplementalProperty>
56        <SupplementalProperty schemeIdUri="urn:mpeg:dash:preselection:2016"
    value="PreselTag2,50 26 47 27 32 25 41 33 31 28 48 39 38 5 6 10 11 12 13
     16 18 19 20 21 22 "/>
57        <Representation id="50" bandwidth="356906" frameRate="60">
58          <SegmentTemplate timescale="90000" duration="13500" startNumber="1"
     initialization="live/06RAWLombardsbruecke-026
    k60fpsyuv420pnew_out_Extractors1.mp4" media="live/Ext_2_Seg_$Number$.mp4
    "/>
59        </Representation>
60      </AdaptationSet>
61
62      <!-- 22 More Adaptation Sets following, each of which creating one
    particular viewport, by the respective extractor track -->
63
64    </Period>
65  </MPD>
```

Listing 5.17: Excerpt of exemplary MPD for generated HHI OMAF DASH stream, generated by the tools of [90].

To test this created stream, the player was first set up on an Akamai Apache web server of the IRT. Here, unfortunately some errors occurred, as some functions of the angular.js were not supported. For that reason, as recommended in the provided git documentation, a `Nginx` web server was implemented for testing, configured to be accessible at `'localhost:5555'`. As only the Safari Browser has native HEVC support, this OMAF HHI player can only be executed on a Mac OS, to test the created DASH MPD. The playback of the created DASH stream works, even though the edges of the tiles are slightly visible. To prevent from these artifacts, OMAF guard bands have to be

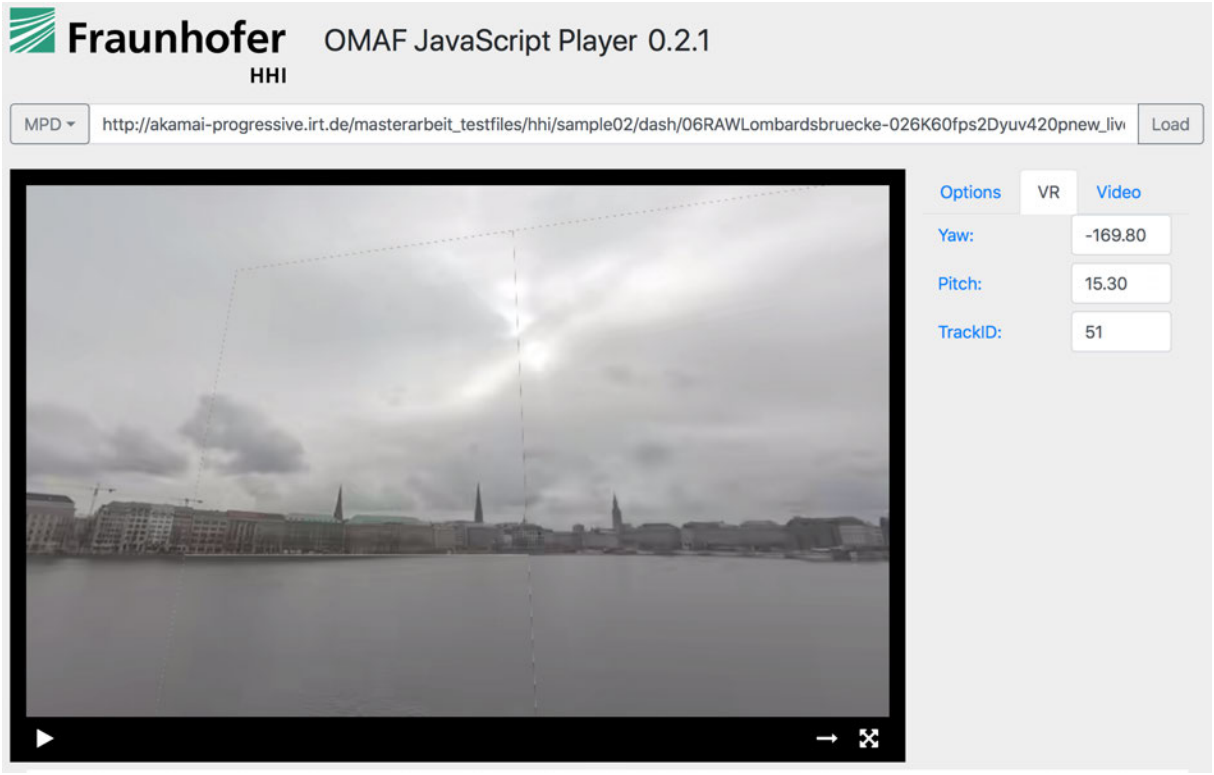


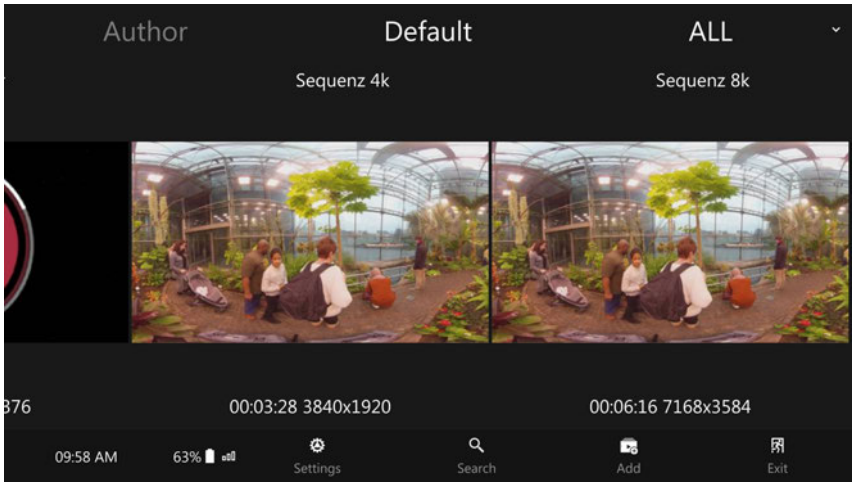
Figure 5.5: Playback screenshot of viewport dependent DASH MPD, created and played by the HHI OMAF implementation tools of [90], with deactivated guard bands.

set by the `'-gbs'` parameter, which by default is 0. Analyzing the network activities of the player, especially the GET requests, via the development tools of the Safari browser, leads to the assumption that viewport adaptivity works properly. That is, first the extractor tracks are requested, referring to the segments of tile-tracks for the viewport, which are requested from the `'qp32'` folder and those for the background are requested from the root folder of the DASH segments. Nonetheless, the playback on the used playback system[7] is stagnant and has short interruptions by black frames, especially when turning the viewport. Additionally, the limitations of the self-configuration in terms of of tiling and segmenting should be mentioned. It should be noted, that this implementation is at development state and can not be considered as finished solution, but more as proof of concept. A screenshot of the created OMAF stream is shown in Figure 5.5. An evaluation and comparison to the other implementations is stated in Section 5.4.

5.3.4 Tiledmedia

Tiledmedia is a private company, providing a partially standardized solution for omnidirectional tiled HEVC and H.264 streaming, using x264 and Intel's SVT-HEVC codec implementations. They were collaborating in the OMAF standardization process and the streaming solution is likely to become OMAF-standard in future, with the announcement of the second OMAF version in MPEG-I Phase 2. For evaluation of the stream, a stitched

[7] iMac Retina 5K, 27 inch, mid 2015, 3.3 GHz Intel Core i5, 32 GB 1600 MHz DDR3, AMD Radeon R9 M290 2 GB



(a)

(b)

Figure 5.6: Playback screenshots of a test sequence of *WDR* in the Android application of Tiledmedia.

8K 30fps 2D equirectangular source sequence of the WDR, recorded with the Insta Pro 2.0, was processed by Tiledmedia and provided through a proprietary Android application. In this case the chosen target bitrate of the transmitted HEVC stream was around 8 Mbps, displayed on a 2K Andorid smartphone. The playback was displayed fluently and viewport-adaptively. The quality switching, according to abrupt head-movement was slightly visible, but adaptation was performed very fast. Since this approach is a proprietary one, not all information regarding tiling structure, viewport-adaptivity and other processing specifications are open accessable and the descriptions of this section are of limited scale. As the examination of the transmission statistics of the stream is only possible through the application itself, or with additional technical effort, the specifications of this streaming solution were requested directly from the developers. The streams have been tested on a Samsung Gear with a Samsung Galaxy S7. Correspondingly, a screenshot of playback with the provided app is depicted in Figure 5.6a and 5.6b.

Currently, the maximum receivable resolution and framerate of the rendered viewport, are 8k mono and close to 6k stereoscopic, at 60 fps, assuming the device can display

it. On a Samsung Galaxy S10 it can be even higher, up to 16K, since it has an 8k decoder. The content is either transmitted as cubemap, or as cylindrical 2D mapping, either of 360° or 180° and with a choice of monoscopic or stereoscopic, and seamless switching between those. The maximum bitrate for transmission of the entire stream is tried to be kept below 20 Mbps, depending on the spatial and temporal complexity of the content. The configuration of the different quality representations, used for the tile-tracks is highly adjustable. Currently, e.g. for 8K source content, they use a low-res and a high-res representation, for the background and the viewport, respectively. In this case, the low-res background could be composed by 6 tiles of 512x512 pixels, i.e. 1536x1024, for the whole cubemap. The high-res viewport could be transmitted as a short GOP with, for instance 4x4 tiles of 512x512 pixels, i.e. 2048x2048 pixels, per cubeface. A long GOP with the same resolution and configuration, is also provided. It is assumed, that the low-res stream is transmitted continuously, and the high-res tiles are transmitted viewport-adaptively. More advanced bandwidth and viewport-adaptive stream switching is under development, to switch between different quality bitrate levels and also between different resolutions. Moreover, Tiledmedia uses CDN prefetching for neighbouring tiles. The playback is configurable for different playback devices and is already supported for all Android devices with a hardware HEVC decoder on Android OS version 6.0 or higher. Further, all iOS devices with a hardware 4K HEVC decoder and iOS version of 11.0 or higher are supported, i.e. including all iPhones 7 and higher. Support for PC playback on the Oculus RIFT or HTC VIVE has already been developed in principle, but is not commercially available and will only be released when the demand on such an application increases. As currently most browsers have no native HEVC support, browser playback is not supported. Further, an AV1 based implementation is under development.

5.4 Comparison of tested implementations

Different approaches for implementation of a tiled 360° stream were presented above, some of which are already OMAF-conform and use the proposed optimization technologies, like viewport-adaptivity, tiles and segmented transmission. This subsection summarizes the findings of the roughly tested implementations. It is important to note, that all of them are still under development at different states and the tests of Nokiatech and HHI refer to pre-versions, which are likely to be enhanced gradually. The first three approaches use MPEG-DASH and create MPDs, to segment the single tile-tracks in an adequate distribution format. Since the Tiledmedia implementation is not entirely standardized and no open-source solution, the exact transmission technology, e.g. whether DASH, HLS etc. is used, is not revealed. Hence, the comparison of the streaming architectures primarily refers to the OMAF solutions of Nokiatech and HHI, and the GPAC-based tiling guide. The tile-based encoding of the source content, into independently decodable parts of a whole frame, and the subsequent packaging of these sequences into a standardized transmission structure, e.g. MPEG DASH, are the key steps for creation of an omnidirectional stream. The resulting OMAF-conform MPDs of a certain OMAF profile, created by different implementations, basically should not deviate from each other, to maintain playback compatibility. For that reason the tiling structure and MPDs are compared to identify differences and OMAF-conformity.

Generally, all use very similar structures, packing either one or more Representation for

each single tile-track into one Adaptation Set. The OMAF Nokiatech DASH creation tool [108], creates multiple Adaptation Sets, each containing two Representations of different bitrates. Hence, one Adaptation Set with one particular ID contains all tiles covering one particular area. The MPD created by the GPAC guide [75] has a very similar structure, although it is not OMAF standard. The differences primarily are, that the Nokiatech approach uses the OMAF brands and does not determine the positions of the tiles by the SRD. Both, HHI and Nokiatech use the OMAF content coverage brand `'cc'` for positioning of the single tiles in the resulting image and specify the projection format by `'pf'` of the respective type, i.e. here Nokiatech uses `'type 1'` and HHI uses `'type 0'`. Since the HHI approach uses the HEVC-based viewport-dependent OMAF video profile with MCTS of different resolutions, the tiling grid and the MPD creation differs from the others. Here, each single tile is packed into one single Adaptation Set, with 24 additional extractor tracks. The MPD of the Nokiatech DASH stream only contains one single pre-selected viewport orientation, packed into one Adaptation Set, containing one single extractor track. Both OMAF solutions use cubemap projection and HEVC encoding, albeit of different codec implementations. Most brands and standardized labels of the OMAF MPDs, i.e. the specified profiles, codecs, mappings, region-wise packing or sphere region quality ranking are identical. Exemplarily, OMAF-conform entries specifying the region wise packing `rwpk`—, the quality ranking `'srqr'`, and the content coverage `'cc'` are picked and shown in Listing 5.18.

```
<EssentialProperty schemeIdUri="urn:mpeg:mpegI:omaf:2017:rwpk" />
<SupplementalProperty schemeIdUri="urn:mpeg:mpegI:omaf:2017:srqr">
...
<SupplementalProperty schemeIdUri="urn:mpeg:mpegI:omaf:2017:cc">
...
```

Listing 5.18: OMAF-conform schemes in MPD, of created DASH stream.

The DASH entry `'viewpoint'` is only used in the MPD created by the HHI tools. Since at this early state of the OMAF standard, not all profiles and tools are supported in each implementation yet, the test player implementations only work with their own proprietary created DASH MPD. Even for content, created by their own tools, the players have a very limited functionality and only work in particular environments. They can rather be considered as test players without the pretension to work with other approaches. Nevertheless, player interoperability for different OMAF profiles and implementations is necessary, and needs to be expedited in the future, as OMAF-conform DASH MPDs should be uniformly interpretable.

At this very early state of development, performance comparison between these approaches is very difficult, since different profiles and encoding parameters were used, and the players have a very limited functional range, working only in particular technical environments. The playback of the Tiledmedia stream, appears to be of high-performance, comparing the streams visually. Their player has a extensive functionality and exceeds the capabilities of other players. As described above, numerous different streaming parameters can be set with few limitations in terms of projection mapping, resolution and bitrate. Notwithstanding, the approaches of Nokiatech and HHI are promising to become efficient OMAF implementations, if the range of encoding parameters, profiles, and encoder implementations is expanded. Further, with future profiles and extensions of the OMAF standard, efficient competitive solutions compared to other proprietary ones will

be proposed, which can be integrated into the existing approaches.

Considering, that all of them are based on MPEG video coding, i.e. H.264 and HEVC, and MPEG streaming is the base for three of them, probably other approaches, based on other codecs and streaming systems might be developed in the future. Tile-based AV1 coding is already emerging, and might be used for similar omnidirectional viewport-adaptive streaming systems. The potential of the streaming architectures, and the OMAF standard, with regard to future usability, especially referring to the introduced use case scenario will be discussed and concluded in Chapter 6.

Chapter 6

Conclusion and outlook

The introduced technologies of the theoretical part of this work and the potential for streaming efficiency of the above tested approaches are summarized in the following. Meanwhile considering the introduced use case of 360° streaming at the public broadcasters. In the end, the technologies and findings are examined with respect to future prospects.

First, 360° video has been defined and content creation and representation have been analyzed, considering recent approaches. Albeit not all have been integrated into the tested implementations of section 5.3. Basic technologies for video coding and streaming have been discussed in the aftermath, with focus on MPEG technologies and HTTP streaming. Moreover, new approaches for streaming of spherical content, by using tiles and novel prediction algorithms, based on machine learning, were roughly described in Section 3.2.6 and 3.2.7. Technical descriptions and realizations of some of those technologies in 360° streaming systems were discussed subsequently, with focus on MPEG-OMAF in Section 4.1.5. Other technical approaches were also introduced in Section 4.3 and examined afterwards, with respect to usability, efficiency and feasibility. The practical part of this work is stated in Chapter 5, testing different implementations, two of them based on OMAF.

Bitrate reduction for streaming of omnidirectional video content is realizable through sphere to planar mapping optimization, efficient video coding, viewport-adaptive segmented transmission and prediction algorithms. Today, streaming of spherical content can be done in several ways. For instance, it can either be done viewport-independently, with efficient video coding, but reduced resolution and bitrate, due to bandwidth constraints. Furthermore, it can be realized viewport-dependently, with much higher efficiency and bitrate for the viewport. Albeit today, this implies reduced compatibility and usability or much more effort and costs yet. Since for live streaming low-complexity and low-delay are required, the stream can be transmitted entirely as conventional planar 2D sequence with respective SEI messages, informing about the spherical specifications. Another way would be with hardware i.e. live encoders and encapsulators, capable of tiling, and players capable of assembling the viewport out of tile-tracks. In this sense, the development of MPEG-OMAF is an important advance, so capable hardware and software might be released in the future and already existing partially standard-conform solutions and proprietary solutions might be adapted to fit the standard. If viewport adaptivity is desired for the drafted use case, today it can be realized in one of the following three ways:

1. By server-sided viewport allocation, and transmission of the stream as conventional decodable stream, i.e. H.264, HEVC etc.

2. By using OMAF encoding and a standardized OMAF player.

3. By other non-standardized, or partially-standardized approaches, e.g. third-party solutions, with proprietary stream creation and playback environments.

Since OMAF is still very young, no fully functional capable players have been released and the content creation implementations have a limited functionality yet. For this reason, if such a solution is desired, players need to be developed and content creation implementations need to be expanded. As server-sided viewport allocation is computationally expensive and a complex task, the amount of clients, receiving the content should be considered. Other third-party solutions may be the most efficient solution yet, with different scope of service, distribution properties and financial plans. It should be considered that for VOD streaming, content can be high efficiently pre-processed, using very time-consumptive but efficient encoding. For transmission environments with higher disposable bandwidths, especially regarding the emerging 5G internet standard, and applications with less priority to low delay, viewport dependency might not imminently be necessary. This also depends on the used codec and encoding parameters. With respect to the public broadcasters one could also propose to wait for more mature implementations, that provide viewport adaptivity, while simultaneously using conventional content allocation for VOD and cooperating with third-party companies for realization of complex live streaming of particular events.

Relating to the initially examined questions of maintaining high quality for omnidirectional video the OMAF standard can be an efficient solution. Exploiting the potential of viewport-adaptivity and video codecs and offering broad disposability are crucial features for effective omnidirectional video streaming. Since OMAF is based on popular and effective video codecs and streaming technologies, such as H.264, HEVC and MPEG-DASH the integration and interoperability are enhanced. Further, by offering viewport-adaptivity and effective high resolutions up to 8K, OMAF meets today's technical demands. Especially with future advances of OMAF implementations generally the usability, streaming efficiency and feasibility can further be improved. Although the tested implementations are of limited extent yet. That is support of other input formats besides RAW `.yuv` and better configuration of encoding parameters are necessary. Noticing, that the exact comparison and examination of the implementations were stated in Section 5.4.

Regarding the content providers, it should be determined for which content and use case 360° streaming is necessary at all, and which playback environment is adequate, i.e. TV, browser, HMD etc. For browser-based playback, the popularity and dispersion of HEVC needs to increase, to provide OMAF-conform omnidirectional content. Other different approaches, based on other codecs and distribution systems, such as AV1 or HLS, can be of enormous importance in the future, as the big companies in content creation and content distribution, such as Netflix, Amazon, Google and co. are interested in open-source solutions. Further, upcoming improvements in conventional video coding, e.g. JEM7, special treatment of spherical content, e.g. MPEG-I, and progress in machine learning algorithms, used for content and viewport prediction, provide novel approaches for efficient omnidirectional streaming.

Bibliography

[1] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016–2021 White Paper. "https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html", accessed: 2018-10-16.

[2] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2017–2022 White Paper. "https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-738429.html", accessed: 2019-03-25.

[3] Compilation error · Issue #203 · ultravideo/kvazaar. "https://github.com/ultravideo/kvazaar/issues/203", accessed: 2019-02-24.

[4] DASH Streaming Support | GPAC. "https://gpac.wp.imt.fr/2012/02/01/dash-support/", accessed: 2019-02-24.

[5] FFmpeg. "https://www.ffmpeg.org/", accessed: 2019-02-01.

[6] GPAC | Multimedia Open Source Project. "https://gpac.wp.imt.fr/", accessed: 2019-02-24.

[7] GPAC-Downloads. "https://gpac.wp.imt.fr/downloads/", accessed: 2019-02-24.

[8] Insta360 360 Camera - Insta360 Download. "https://www.insta360.com/download/insta360-pro2", accessed: 2019-03-16.

[9] Insta360 Pro 2 - Better from every angle. "https://www.insta360.com/product/insta360-pro2/?gclid=EAIaIQobChMIuJ7B8eDn3QIVV6WaCh0NfgoQEAAYASAAEgIM5_D_BwE", accessed: 2018-10-02".

[10] MP4 Box General Documentation | GPAC. "https://gpac.wp.imt.fr/mp4box/mp4box-documentation/", accessed: 2019-02-24.

[11] MP4client quality adaptation not working when playing SRD tiled video · Issue #827 · gpac/gpac. "https://github.com/gpac/gpac/issues/827", accessed: 2019-02-24.

[12] Osmo4 | GPAC. "https://gpac.wp.imt.fr/player/", accessed: 2019-02-24.

[13] Tiledmedia. "https://www.tiledmedia.com/", accessed: 2019-02-24.

[14] Ultra Video Group. "http://ultravideo.cs.tut.fi/", accessed: 2019-02-24.

[15] Virtual Reality - Prognose zum Umsatz weltweit bis 2021 | Statistik. "https://de.statista.com/statistik/daten/studie/318536/umfrage/prognose-zum-umsatz-mit-virtual-reality-weltweit/" and "https://www.superdataresearch.com/wp-content/uploads/2018/01/SuperData-Research-Virtual-Reality-XR-Report-1.png", accessed: 2019-03-20.

[16] WebVR concepts. "https://developer.mozilla.org/en-US/docs/Web/API/WebVR_API/Concepts", accessed: 2018-10-02.

[17] GPAC main code repository. GPAC Implementation, February 2019. "https://github.com/gpac/gpac", accessed: 2019-02-25.

[18] HEVC Test Model (HM) Documentation, March 2019. "https://hevc.hhi.fraunhofer.de/HM-doc/", accessed: 2019-03-02.

[19] Kvazaar - an open-source HEVC encoder. Implementation, February 2019. "http://github.com/ultravideo/kvazaar", accessed: 2019-02-27.

[20] MPEG-DASH Access Library: Official ISO/IEC MPEG-DASH Reference Implementation - bitmovin/libdash, February 2019. "https://github.com/bitmovin/libdash", accessed: 2019-02-24.

[21] MPEG-DASH Access Library: Official ISO/IEC MPEG-DASH Reference Implementation - nokia/libdash, March 2019. "https://github.com/nokia/libdash", accessed: 2019-03-02.

[22] Adeel Abbas and David Newman. AHG8: Rotated Sphere Projection for 360 Video. April 2017. [Online]. Available: http://phenix.it-sudparis.eu/jvet/doc_end_user/current_document.php?id=3060.

[23] Hamed Ahmadi, Omar Eltobgy, and Mohamed Hefeeda. Adaptive Multicast Streaming of Virtual Reality Content to Mobile Users. In *Proceedings of the on Thematic Workshops of ACM Multimedia 2017*, Thematic Workshops '17, pages 170–178, New York, NY, USA, 2017. ACM.

[24] Elena Alshina, Gary J Sullivan, Microsoft Corp, Jens-Rainer Ohm, and Jill Boyce. *JVET-G1001: Algorithm description of Joint Exploration Test Model 7 (JEM7).* July 2018.

[25] David Austerberry. *The technology of video and audio streaming.* Focal Press, Burlington, MA, 2nd edition, 2004.

[26] L. Bassbouss, S. Pham, and S. Steglich. Streaming and playback of 16k 360° videos on the web. In *2018 IEEE Middle East and North Africa Communications Conference (MENACOMM)*, pages 1–5, April 2018.

[27] Ali Borji, Ming-Ming Cheng, Huaizu Jiang, and Jia Li. Salient Object Detection: A Benchmark. *IEEE Transactions on Image Processing*, 24(12):5706–5722, December 2015. arXiv: 1501.02741.

[28] J. M. Boyce, Y. Ye, J. Chen, and A. K. Ramasubramonian. Overview of SHVC: Scalable Extensions of the High Efficiency Video Coding Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 26(1):20–34, January 2016.

[29] Jill M. Boyce. Omnidirectional projection indication SEI message geometry_type and projection_type changes. page 11, 27th Meeting: Hobart, April 2017. Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11. [Online]. Available: `https://www.researchgate.net/publication/327253566_JCTVC-AA0035_Omnidirectional_projection_indication_SEI_message_geometry_type_and_projection_type_changes`.

[30] Jill M. Boyce, Adarsh Ramasubramanian, R. Skupin, Gary J. Sullivan, and Alexis Tourapis. HEVC Additional Supplemental Enhancement Information (Draft 1). page 20, 26th Meeting: Geneva, CH, January 2017. Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11.

[31] Jill M. Boyce, Adarsh Ramasubramanian, R. Skupin, Gary J. Sullivan, and Alexis Tourapis. ISO/IEC 23008-2:2017/Amd 3:2018 Information technology — High efficiency coding and media delivery in heterogeneous environments — Part 2: High efficiency video coding AMENDMENT 3: Additional supplemental enhancement information. Geneva, CH, July 2018. International Organization for Standardization.

[32] Jill M. Boyce, Li Xiang, Karsten Suehring, and Vadim Seregin. JVET-J1010: JVET common test conditions and software reference configurations. 10th Meeting: San Diego, US, April 2018. Joint Video Exploration Team (JVET) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11.

[33] Mary-Luc Champel, Rob Koenen, Gauthier Lafruit, and Madhukar Budagavi. Proposed Draft 1.0 of TR: Technical Report on Architectures for Immersive Media, n17685. San Diego, US, April 2018. International Organization for Standardization ISO/IEC JTC1/SC29/WG11.

[34] Y. Chen, D. Murherjee, J. Han, A. Grange, Y. Xu, Z. Liu, S. Parker, C. Chen, H. Su, U. Joshi, C. Chiang, Y. Wang, P. Wilkins, J. Bankoski, L. Trudeau, N. Egge, J. Valin, T. Davies, S. Midtskogen, A. Norkin, and P. de Rivaz. An Overview of Core Coding Tools in the AV1 Video Codec. In *2018 Picture Coding Symposium (PCS)*, pages 41–45, June 2018.

[35] Byeongdoo Choi, Ye-Kui Wang, and Miska M. Hannuksela. WD on ISO/IEC 23000-20 Omnidirectional Media Application Format, n16189. Working Draft, International Organization for Standardization ISO/IEC JTC1/SC29/WG11, Geneva, CH, June 2016.

[36] X. Corbillon, G. Simon, A. Devlic, and J. Chakareski. Viewport-adaptive navigable 360-degree video delivery. In *2017 IEEE International Conference on Communications (ICC)*, pages 1–7, May 2017.

[37] Thomas Daede. AV1 Update, May 2017. "https://people.xiph.org/~tdaede/demuxed_av1_2017.pdf", accessed: 2018-07-11.

[38] Tarek El-Ganainy. Spatiotemporal Rate Adaptive Tiled Scheme for 360 Sports Events. *arXiv:1705.04911 [cs]*, May 2017. [Online]. Available: http://arxiv.org/abs/1705.04911.

[39] Tarek El-Ganainy and Mohamed Hefeeda. Streaming Virtual Reality Content. *arXiv:1612.08350 [cs]*, December 2016. [Online]. Available: http://arxiv.org/abs/1612.08350.

[40] N. Engelmann. *Virtual Reality Gaming: Potential der Technologie für die Welt der digitalen Spiele*. Tectum Wissenschaftsverlag, July 2018.

[41] Gorry Fairhurst. Unicast, Broadcast, and Multicast, October 2009. "https://erg.abdn.ac.uk/users/gorry/course/intro-pages/uni-b-mcast.html", accessed: 2018-10-16.

[42] Ching-Ling Fan, Jean Lee, Wen-Chih Lo, Chun-Ying Huang, Kuan-Ta Chen, and Cheng-Hsin Hsu. Fixation Prediction for 360 Video Streaming in Head-Mounted Virtual Reality. In *Proceedings of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video*, NOSSDAV'17, pages 67–72, New York, NY, USA, 2017. ACM.

[43] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. Technical report, 1999. [Online]. Available: https://www.rfc-editor.org/info/rfc2616.

[44] Roy T. Fielding, Mark Nottingham, and Julian Reschke. HTTP Semantics. Internet-Draft Version: draft-ietf-httpbis-semantics-03, Internet Engineering Task Force, October 2018.

[45] Alliance for Open Media and Yunqing Wang. av1/common/enums.h - aom - Git at Google, 2016. "https://aomedia.googlesource.com/aom/+/master/av1/common/enums.h", accessed: 2018-11-13.

[46] International Organization for Standardization. ISO/IEC 14496-10 Information technology – Coding of audio-visual objects – Part 10: Advanced Video Coding. Geneva, CH, December 2003.

[47] International Organization for Standardization. ISO/IEC FDIS 23009-1 Information technology – Dynamic adaptive streaming over HTTP (DASH) – Part 1: Media presentation description and segment formats. Geneva, CH, April 2012.

[48] International Organization for Standardization. ISO/IEC 23008-2 Information technology – High efficiency coding and media delivery in heterogeneous environments – Part 2: High efficiency video coding. Geneva, CH, December 2013.

[49] International Organization for Standardization. ISO/IEC 23009-1:2014 Information technology – Dynamic adaptive streaming over HTTP (DASH) – Part 1: Media presentation description and segment formats. Geneva, CH, May 2014.

[50] International Organization for Standardization. ISO/IEC 14496-12:2015 Information technology – Coding of audio-visual objects – Part 12: ISO base media file format. Geneva, CH, February 2015.

[51] International Organization for Standardization. ISO/IEC 23090-2 | Information technology – Coded representation of immersive media – Part 2: Omnidirectional media format. Geneva, CH, January 2019.

[52] VR Industry Forum. VR Industry Forum - GUIDELINES, Version 1.1draft009 2018-05-11, May 2018. "https://www.vr-if.org/wp-content/uploads/vrif2018.018.09-clean.pdf", accessed: 2018-11-24.

[53] C. Fu, L. Wan, T. Wong, and C. Leung. The Rhombic Dodecahedron Map: An Efficient Scheme for Encoding Panoramic Video. *IEEE Transactions on Multimedia*, 11(4):634–644, June 2009.

[54] G. Gankhuyag, J. Jeong, and Y. Kim. Motion-constrained AV1 Encoder for 360 VR Tiled Streaming. In *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 596–598, October 2018.

[55] R. Ghaznavi-Youvalari, A. Zare, H. Fang, A. Aminlou, Q. Xie, M. M. Hannuksela, and M. Gabbouj. Comparison of HEVC coding schemes for tile-based viewport-adaptive streaming of omnidirectional video. In *2017 IEEE 19th International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–6, October 2017.

[56] Adrian Grange, Peter de Rivaz, and Jonathan Hunt. VP9 Bitstream & Decoding Process Specification, March 2016. "https://storage.googleapis.com/downloads.webmproject.org/docs/vp9/vp9-bitstream-specification-v0.6-20160331-draft.pdf", accessed: 2018-07-11.

[57] O. Grau and G. Custance. *Virtual Art: From Illusion to Immersion*. Leonardo (Series) (Cambridge, Mass.). MIT Press, 2003.

[58] N. Greene. Environment Mapping and Other Applications of World Projections. *IEEE Computer Graphics and Applications*, 6(11):21–29, November 1986.

[59] Dan Grois, Tung Nguyen, and Detlev Marpe. Performance comparison of AV1, JEM, VP9, and HEVC encoders, (conference presentation). pages 1–12, September 2017.

[60] Ian P Howard, Brian J Rogers, et al. *Binocular vision and stereopsis*. Oxford University Press, USA, 1995.

[61] C. Huitema. Real Time Control Protocol (RTCP) attribute in Session Description Protocol (SDP). Technical report, 2003. [Online]. Available: https://www.rfc-editor.org/info/rfc3605.

[62] Akamai Technologies Inc. Facts & Figures | Akamai. "https://www.akamai.com/uk/en/about/facts-figures.jsp", accessed: 2019-03-20.

[63] Apple Inc. About the Common Media Application Format with HTTP Live Streaming | Apple Developer Documentation. "https://developer.apple.com/documentation/http_live_streaming/about_the_common_media_application_format_with_http_live_streaming", accessed: 2019-01-15.

[64] Apple Inc. Apple developer, HTTP live streaming. "https://developer.apple.com/streaming/", accessed: 2018-09-20.

[65] Apple Inc. Understanding the HTTP Live Streaming Architecture | Apple Developer Documentation. "https://developer.apple.com/documentation/http_live_streaming/understanding_the_http_live_streaming_architecture", accessed: 2018-10-25.

[66] Apple Inc. HTTP Live Streaming | Apple Developer Documentation, January 2016. "https://developer.apple.com/documentation/http_live_streaming", accessed: 2018-10-25.

[67] Apple Inc. Using HTTP Live Streaming, January 2016. "https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/StreamingMediaGuide/UsingHTTPLiveStreaming/UsingHTTPLiveStreaming.html", accessed: 2018-10-31.

[68] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259, November 1998.

[69] Kyuheon Kim, Kyungmo Park, Sunghee Hwang Hwang, and Jaeyeon Song. International Organization for Standardization ISO/IEC JTC1/SC29/WG11 Coding of moving pictures and audio - Draft of White paper on MPEG Media Transport (MMT). Technical report, Geneva, CH, February 2015.

[70] Y. Kim, J. Huh, and J. Jeong. Distributed Video Transcoding System for 8k 360 VR Tiled Streaming Service. In *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 592–595, October 2018.

[71] Evgeny Kuzyakov and David Pio. Under the hood: Building 360 video, October 2015. "https://code.fb.com/video-engineering/under-the-hood-building-360-video/", accessed: 2018-10-05.

[72] Evgeny Kuzyakov and David Pio. Next-generation video encoding techniques for 360 video and VR, January 2016. "https://code.fb.com/virtual-reality/next-generation-video-encoding-techniques-for-360-video-and-vr/", accessed: 2018-10-05.

[73] Evgeny Kuzyakov, David Pio, and Sean Liu. Facebook for Developers - Optimizing 360 Video for Oculus, February 2018. "https://developers.facebook.com/videos/f8-2016/optimizing-360-video-for-oculus/", accessed: 2018-10-05.

[74] T. Laude, Y. G. Adhisantoso, J. Voges, M. Munderloh, and J. Ostermann. A Comparison of JEM and AV1 with HEVC: Coding Tools, Coding Efficiency and Complexity. In *2018 Picture Coding Symposium (PCS)*, pages 36–40, June 2018.

[75] Le Feuvre, Jean - GPAC. HEVC Tile-based adaptation guide | GPAC. "https://gpac.wp.imt.fr/2017/02/01/hevc-tile-based-adaptation-guide/", accessed: 2019-02-24.

[76] Stefan Lederer and Florian Bacher. Datasets | ITEC – Dynamic Adaptive Streaming over HTTP. "http://www-itec.uni-klu.ac.at/dash/?page_id=207", accessed: 2018-12-05.

[77] Stefan Lederer, Christopher Müller, and Christian Timmerer. Dynamic Adaptive Streaming over HTTP Dataset. In *Proceedings of the 3rd Multimedia Systems Conference*, MMSys '12, pages 89–94, New York, NY, USA, 2012. ACM.

[78] H. Lee, Y. Lee, J. Lee, D. Lee, and H. Shin. Design of a mobile video streaming system using adaptive spatial resolution control. *IEEE Transactions on Consumer Electronics*, 55(3):1682–1689, August 2009.

[79] J. Li, Z. Wen, S. Li, Y. Zhao, B. Guo, and J. Wen. Novel tile segmentation scheme for omnidirectional video. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 370–374, September 2016.

[80] Y. Lim, S. Aoki, I. Bouazizi, and J. Song. New MPEG Transport Standard for Next Generation Hybrid Broadcasting System With IP. *IEEE Transactions on Broadcasting*, 60(2):160–169, June 2014.

[81] Nikolai Longolius. *Web-TV AV-Streaming im Internet, Echtzeitübertragung von Ton & Bild im Internet*. O'Reilly Germany, Köln, 1. edition, 2011.

[82] D. Marpe, T. Wiegand, and G. J. Sullivan. The H.264/MPEG4 advanced video coding standard and its applications. *IEEE Communications Magazine*, 44(8):134–143, August 2006.

[83] Pascal Massimino. AOM - AV1 How does it work?, July 2017. "ttps://parisvideotech.com/wp-content/uploads/2017/07/AOM-AV1-Video-Tech-meet-up.pdf", accessed: 2018-09-12.

[84] Torsten Milde. *Videokompressionsverfahren im Vergleich. JPEG, MPEG, H.261, XCCC, Wavelets, Fraktale*. Dpunkt.Verlag GmbH, Heidelberg, Germany, January 1999.

[85] Gert Moesen. Viewport dependent MPEG-DASH streaming of 360 degree natively tiled HEVC video in web browser context. University of Hasselt. 2018. [Online]. Available: https://uhdspace.uhasselt.be/dspace/bitstream/1942/26918/1/862be5ce-7a5f-4bf8-8e9e-8012633524b8.pdf.

[86] D. Mukherjee, J. Bankoski, A. Grange, J. Han, J. Koleszar, P. Wilkins, Y. Xu, and R. Bultje. The latest open-source video codec VP9 - An overview and preliminary results. In *2013 Picture Coding Symposium (PCS)*, pages 390–393, December 2013.

[87] Omar A. Niamut, Emmanuel Thomas, Lucia D'Acunto, Cyril Concolato, Franck Denoual, and Seong Yong Lim. MPEG DASH SRD: Spatial Relationship Description. In *Proceedings of the 7th International Conference on Multimedia Systems*, MMSys '16, pages 5:1–5:8, New York, NY, USA, 2016. ACM. event-place: Klagenfurt, Austria.

[88] Kyungmo Park, Youngkwon Lim, Shuichi Aoki, Gerard Fernando, and Jin Young Lee. Text of ISO/IEC 2nd CD 23008-1 MPEG Media Transport. Geneva, CH, January 2013. International Organization for Standardization, ISO/IEC JTC1/SC29/WG11 Coding of moving pictures and audio.

[89] Emil Peerson. San Francisco 2, February 2015. [Online]. Available: http://www.humus.name/index.php?page=Textures&start=16.

[90] Dimitri Podborski, Jangwoo Son, Gurdeep Singh Bhullar, Cornelius Hellge, and Thomas Schierl. HTML5 MSE Playback of MPEG 360 VR Tiled Streaming, 2019. [Online]. Available: https://arxiv.org/abs/1903.02971.

[91] Feng Qian, Lusheng Ji, Bo Han, and Vijay Gopalakrishnan. Optimizing 360 Video Delivery over Cellular Networks. In *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges*, ATC '16, pages 1–6, New York, NY, USA, 2016. ACM.

[92] Ed R. Pantos and W. May. HTTP Live Streaming. Technical report, 2017. "https://www.rfc-editor.org/info/rfc8216", accessed: 2018-10-25.

[93] Iain E. G. Richardson. *H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia*. Wiley, Aberdeen, February 2004.

[94] Oliver Röder. *Grundlagen der Stereoskopie: Analyse der Aufnahme und Projektion von 3-D Bildern*. VDM, Verlag Dr. Müller, Saarbrücken, 2007.

[95] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. Technical report, 2003. [Online]. Available: https://www.rfc-editor.org/info/rfc3550.

[96] H. Schulzrinne, A. Rao, R. Lanphier, M. Westerlund, and Ed M. Stiemerling. Real-Time Streaming Protocol Version 2.0. Technical report, 2016. [Online]. Available: https://www.rfc-editor.org/info/rfc7826.

[97] H. Schwarz, D. Marpe, and T. Wiegand. Overview of the Scalable Video Coding Extension of the H.264/AVC Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9):1103–1120, September 2007.

[98] Andre Seixas Dias, Blasi Saverio, Fiona Riveira, Izquierdo Ebroul, and Marta Mrak. An overview of recent video coding developments in MPEG and AOMEDIA. Amsterdam, September 2018. 1BBC, Research & Development Department, UK 2Queen Mary University of London, Multimedia and Vision Group, UK.

[99] J. Shi, Q. Yan, L. Xu, and J. Jia. Hierarchical Image Saliency Detection on Extended CSSD. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(4):717–729, April 2016.

[100] David Singer and Kilroy Hughes. Text of Common Media Application Format for Segmented Media | N16186. pages 1–187, Geneva, CH, June 2016. International Organization for Standardization ISO/IEC JTC1/SC29/WG11.

[101] David Singer and Thomas Stockhammer. White paper on an Overview of the ISO Base Media File Format, October 2018. "https://mpeg.chiariglione.org/sites/default/files/files/standards/docs/N18093_ISOFF%28TS%29.pptx", accessed: 2018-12-14.

[102] Robert H. Spector. Visual Fields. In H. Kenneth Walker, W. Dallas Hall, and J. Willis Hurst, editors, *Clinical Methods: The History, Physical, and Laboratory Examinations*. Butterworths, Boston, 3rd edition, 1990.

[103] K. K. Sreedhar, A. Aminlou, M. M. Hannuksela, and M. Gabbouj. Viewport-Adaptive Encoding and Streaming of 360-Degree Video for Virtual Reality Applications. In *2016 IEEE International Symposium on Multimedia (ISM)*, pages 583–586, December 2016.

[104] W. Richard Stevens. *TCP/IP Illustrated (Vol. 3): TCP for Transactions, HTTP, NNTP, and the Unix Domain Protocols*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1996.

[105] Yu-Chuan Su and Kristen Grauman. Learning Compressible 360 Video Isomers. *arXiv:1712.04083 [cs]*, December 2017. arXiv: 1712.04083.

[106] G. J. Sullivan, J. Ohm, W. Han, and T. Wiegand. Overview of the High Efficiency Video Coding (HEVC) Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1649–1668, December 2012.

[107] Vivienne Sze, Madhukar Budagavi, and Gary J. Sullivan, editors. *High Efficiency Video Coding (HEVC): Algorithms and Architectures*. Integrated Circuits and Systems. Springer International Publishing, 2014.

[108] Nokia Technologies. Nokia OMAF implementation, February 2019. "https://github.com/nokiatech/omaf", accessed: 2019-02-25.

[109] Nokia Technologies. Nokia OMAF implementation. Usage instructions for OMAF Creator Viewport dependent mode, February 2019. "https://github.com/nokiatech/omaf/wiki/Usage-instructions-for-OMAF-Creator---Viewport-dependent-mode", accessed: 2019-02-25.

[110] Nokia Technologies. Nokiatech High Efficiency Image File Format Implementation, March 2019. "https://github.com/nokiatech/heif", accessed: 2019-03-01.

[111] T. C. Thang, Q. Ho, J. W. Kang, and A. T. Pham. Adaptive streaming of audiovisual content using MPEG DASH. *IEEE Transactions on Consumer Electronics*, 58(1):78–85, February 2012.

[112] Pankaj Topiwala, Wei Dai, Madhu Krishnan, Adeel Abbas, Sandeep Doshi, and David Newman. Performance comparison of AV1, HEVC, and JVET video codecs on 360 (spherical) video. In *Applications of Digital Image Processing XL*, volume 10396, page 1039609. International Society for Optics and Photonics, September 2017.

[113] L. Trudeau, N. Egge, and D. Barr. Predicting Chroma from Luma in AV1. In *2018 Data Compression Conference*, pages 374–382, March 2018.

[114] International Telecommunication Union. *Recommendation ITU-R BT.500-13 (01/2012), Methodology for the subjective assessment of the quality of television pictures*. Recommendation BT.500. Geneva, 2012.

[115] Ye-Kui Wang. An Overview of Omnidirectional MediA Format (OMAF), December 2017. [Online]. Available: https://mpeg.chiariglione.org/sites/default/files/files/standards/docs/OMAFoverview2017-1212-10.zip.

[116] John Watkinson. *MPEG Handbook*. Focal Press, 1st edition edition, 2001.

[117] Tobias Young and Tom Patterson. Equirectangular (0-degree) Map Projection Image, physical map. 15-degree graticule. [Online]. Available: https://map-projections.net/img/flat-ocean-w/rectang-0.jpg.

[118] Matt Yu, Haricharan Lakshman, and Bernd Girod. Content Adaptive Representations of Omnidirectional Videos for Cinematic Virtual Reality. In *Proceedings of the 3rd International Workshop on Immersive Media Experiences*, ImmersiveME '15, pages 1–6, New York, NY, USA, 2015. ACM.

[119] Alireza Zare, Alireza Aminlou, Miska M. Hannuksela, and Moncef Gabbouj. HEVC-compliant Tile-based Streaming of Panoramic Video for Virtual Reality Applications. In *Proceedings of the 24th ACM International Conference on Multimedia*, MM '16, pages 601–605, New York, NY, USA, 2016. ACM.

Ich versichere, die vorliegende Arbeit selbstständig ohne fremde Hilfe verfasst und keine anderen Quellen und Hilfsmittel als die angegebenen benutzt zu haben. Die aus anderen Werken wörtlich entnommenen Stellen oder dem Sinn nach entlehnten Passagen sind durch Quellenangaben eindeutig kenntlich gemacht.

Ort, Datum Benedikt Meyer-Schwickerath