
**Information technology — Coded
representation of immersive media —
Part 2:
Omnidirectional media format**

*Technologies de l'information — Représentation codée de média
immersifs — Partie 2: Format de média omnidirectionnel*





COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2019

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Fax: +41 22 749 09 47
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Foreword	vii
Introduction	viii
1 Scope	1
2 Normative references	1
3 Terms, definitions, abbreviated terms, and conventions	2
3.1 Terms and definitions	2
3.2 Abbreviated terms	6
3.3 Arithmetic operators and mathematical functions	7
3.4 Order of operation precedence	8
3.5 Range notation	9
3.6 Variables	9
3.7 Processes	9
4 Overview	9
4.1 Organization of this document	9
4.2 Overall architecture for omnidirectional media with projected video	10
4.2.1 Overview	10
4.2.2 Stitching, rotation, projection, and region-wise packing	11
4.3 Overall architecture for omnidirectional media with fisheye video	12
4.4 Conformance and interoperability	13
4.4.1 General	13
4.4.2 Media profiles	14
4.4.3 Presentation profiles	15
4.4.4 Summary of referenceable code points	16
5 Omnidirectional video projection and region-wise packing	19
5.1 Coordinate system	19
5.2 Omnidirectional projection formats	20
5.2.1 General	20
5.2.2 Equirectangular projection for one sample location	20
5.2.3 Cubemap projection for one sample location	21
5.3 Conversion from the local coordinate axes to the global coordinate axes	23
5.4 Region-wise packing formats	24
5.4.1 General	24
5.4.2 Conversion of one sample location for rectangular region-wise packing	24
6 Fisheye omnidirectional video	25
6.1 General	25
6.2 FisheyeVideoEssentialInfoStruct syntax structure	26
6.2.1 Syntax	26
6.2.2 Semantics	26
6.3 FisheyeVideoSupplementalInfoStruct syntax structure	29
6.3.1 Syntax	29

6.3.2	Semantics	30
7	Omnidirectional media storage and metadata signalling in the ISOBMFF	33
7.1	Generic extensions to the ISOBMFF	33
7.1.1	Stereoscopic video track grouping	33
7.1.2	Indication of <code>track_group_id</code> uniqueness	34
7.1.3	Updated semantics of <code>track_IDs</code> of the track reference box	34
7.1.4	Indication of a track not intended to be presented alone	34
7.1.5	Timed metadata tracks	34
7.1.6	Compatible scheme type box	35
7.1.7	Multiple transformations for a single transformed media track	35
7.1.8	The ' <code>codecs</code> ' parameter for a transformed media track	35
7.1.9	Track type box	36
7.1.10	Clarifications on the stereo video box	36
7.2	Generic extensions to ISO/IEC 14496-15	37
7.2.1	Alternative extraction source track grouping	37
7.2.2	Tile base track association with coverage information box and timed metadata data track	37
7.3	OMAF-specific extensions to the ISOBMFF	37
7.3.1	Sync samples in timed metadata tracks	37
7.4	OMAF-specific extensions to ISO/IEC 14496-15	37
7.4.1	Coverage information box in a tile base track	37
7.5	Structures and semantics that are common for video tracks and image items	38
7.5.1	Semantics of sample locations within a decoded picture	38
7.5.2	Projection format structure	41
7.5.3	Region-wise packing structure	41
7.5.4	Rotation structure	48
7.5.5	Content coverage structure	48
7.5.6	Sphere region structure	49
7.6	Restricted video schemes for omnidirectional video	51
7.6.1	Scheme types	51
7.6.2	Projected omnidirectional video box	54
7.6.3	Fisheye omnidirectional video box	55
7.6.4	Region-wise packing box	55
7.6.5	Rotation box	56
7.6.6	Coverage information box	56
7.7	Timed metadata for sphere regions	56
7.7.1	General	56
7.7.2	Sample entry	57
7.7.3	Sample format	58
7.7.4	Initial viewing orientation	58
7.7.5	Recommended viewport	59
7.7.6	Timed text sphere location metadata	60
7.8	Signalling of region-wise quality ranking	61
7.8.1	General	61
7.8.2	Spherical region-wise quality ranking	61
7.8.3	2D region-wise quality ranking	63
7.9	Storage of omnidirectional images	65
7.9.1	General	65
7.9.2	Frame packing item property	65
7.9.3	Projection format item property	65
7.9.4	Essential fisheye image item property	66
7.9.5	Supplemental fisheye image item property	67
7.9.6	Region-wise packing item property	67
7.9.7	Rotation item property	68

7.9.8	Coverage information item property.....	68
7.9.9	Initial viewing orientation item property	69
7.10	Storage of timed text for omnidirectional video	69
7.10.1	General	69
7.10.2	OMAF timed text configuration box	70
7.10.3	IMSC1 tracks.....	72
7.10.4	WebVTT tracks	73
8	Omnidirectional media encapsulation and signalling in DASH	73
8.1	Architecture of DASH delivery in OMAF	73
8.2	Usage of DASH in OMAF	74
8.2.1	General	74
8.2.2	Signalling of stereoscopic frame packing	74
8.2.3	Carriage of timed metadata.....	74
8.3	DASH MPD descriptors for omnidirectional media.....	75
8.3.1	XML namespace and schema	75
8.3.2	Signalling of projection type information.....	75
8.3.3	Signalling of region-wise packing type	76
8.3.4	Signalling of content coverage	76
8.3.5	Signalling of spherical region-wise quality ranking	79
8.3.6	Signalling of 2D region-wise quality ranking.....	84
8.3.7	Signalling of fisheye omnidirectional video.....	88
9	Omnidirectional media encapsulation and signalling in MMT	89
9.1	Architecture of MMT delivery in OMAF	89
9.2	OMAF signalling in MPEG composition information.....	90
9.3	VR application-specific MMT signalling	90
9.3.1	General	90
9.3.2	MMT signalling.....	91
10	Media profiles	103
10.1	Video profiles.....	103
10.1.1	Overview	103
10.1.2	HEVC-based viewport-independent OMAF video profile	103
10.1.3	HEVC-based viewport-dependent OMAF video profile	106
10.1.4	AVC-based viewport-dependent OMAF video profile.....	109
10.2	Audio profiles.....	111
10.2.1	Overview	111
10.2.2	OMAF 3D audio baseline profile	111
10.2.3	OMAF 2D audio legacy profile.....	114
10.3	Image profiles.....	118
10.3.1	Overview	118
10.3.2	Common specifications for image profiles	119
10.3.3	OMAF HEVC image profile	120
10.3.4	OMAF legacy image profile.....	121
10.4	Timed text profiles	122
10.4.1	Overview	122
10.4.2	OMAF IMSC1 timed text profile	123
10.4.3	OMAF WebVTT timed text profile.....	123
11	Presentation profiles.....	124
11.1	OMAF viewport-independent baseline presentation profile.....	124
11.1.1	General (informative)	124

11.1.2 ISO base media file format constraints	124
11.2 OMAF viewport-dependent baseline presentation profile	124
11.2.1 General	124
11.2.2 ISO base media file format constraints	124
Annex A (normative) OMAF DASH schema	125
Annex B (normative) DASH integration of media profiles	128
Annex C (normative) CMAF integration of media profiles	134
Annex D (informative) Viewport-dependent omnidirectional video processing	136
Annex E (informative) DASH MPD examples	154
Annex F (informative) MMT signalling examples.....	158

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see: www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

A list of all parts in the ISO/IEC 23090 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

When omnidirectional media content is consumed with a head-mounted display and headphones, only the parts of the media that correspond to the user's viewing orientation are rendered, as if the user were in the spot where and when the media was captured. One of the most popular forms of omnidirectional media applications is omnidirectional video, also known as 360° video. Omnidirectional video is typically captured by multiple cameras that cover up to 360° of the scene. Compared to traditional media application formats, the end-to-end technology for omnidirectional video (from capture to playback) is more easily fragmented due to various capturing and video projection technologies. From the capture side, there exist many different types of cameras capable of capturing 360° video, and on the playback side there are many different devices that are able to playback 360° video with different processing capabilities. To avoid fragmentation of omnidirectional media content and devices, a standardized format for omnidirectional media applications is specified in this document.

This document defines a media format that enables omnidirectional media applications, focusing on 360° video, images, and audio, as well as associated timed text. What is specified in this document includes (but is not limited to):

- 1) a coordinate system that consists of a unit sphere and three coordinate axes, namely the X (back-to-front) axis, the Y (lateral, side-to-side) axis, and the Z (vertical, up) axis,
- 2) projection and rectangular region-wise packing methods that may be used for conversion of a spherical video sequence or image into a two-dimensional rectangular video sequence or image, respectively,
- 3) storage of omnidirectional media and the associated metadata using the ISO base media file format (ISOBMFF) as specified in ISO/IEC 14496-12,
- 4) encapsulation, signalling, and streaming of omnidirectional media in a media streaming system, e.g., dynamic adaptive streaming over HTTP (DASH) as specified in ISO/IEC 23009-1 or MPEG media transport (MMT) as specified in ISO/IEC 23008-1, and
- 5) media profiles and presentation profiles that provide interoperable and conformance points for media codecs as well as media coding and encapsulation configurations that may be used for compression, streaming, and playback of the omnidirectional media content.

Information technology — Coded representation of immersive media —

Part 2: Omnidirectional media format

1 Scope

This document specifies the omnidirectional media format for coding, storage, delivery, and rendering of omnidirectional media, including video, images, audio, and timed text.

In an OMAF player the user's viewing perspective is from the centre of the sphere looking outward towards the inside surface of the sphere.

NOTE 1 In this document, only 3 degrees of freedom (3DOF) is supported. In other words, purely translational movement of the user does not result in different omnidirectional media being rendered to the user. For 3DOF support with stereoscopic video, when the user rolls his/her head, there could be a stereoscopic rendering issue.

NOTE 2 Omnidirectional video could contain graphics elements generated by computer graphics but encoded as video.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 10918-1, *Information technology — Digital compression and coding of continuous-tone still images — Part 1: Requirements and guidelines*

ISO/IEC 14496-1, *Information technology — Coding of audio-visual objects — Part 1: Systems*

ISO/IEC 14496-3:2009, *Information technology — Coding of audio-visual objects — Part 3: Audio*

ISO/IEC 14496-10:2014, *Information technology — Coding of audio-visual objects — Part 10: Advanced video coding*

ISO/IEC 14496-12, *Information technology — Coding of audio-visual objects — Part 12: ISO base media file format*

ISO/IEC 14496-14, *Information technology — Coding of audio-visual objects — Part 14, MP4 file format*

ISO/IEC 14496-15:2017, *Information technology — Coding of audio-visual objects — Part 15, Carriage of network abstraction layer (NAL) unit structured video in the ISO base media file format*

ISO/IEC 14496-30:2018, *Information technology — Coding of audio-visual objects — Part 30: Timed text and other visual overlays in ISO base media file format*

ISO/IEC 23000-19:2018, *Information technology — Multimedia application format (MPEG-A) — Part 19: Common media application format (CMAF) for segmented media*

ISO/IEC 23003-4:2015, *Information technology — MPEG audio technologies — Part 4: Dynamic range control*

ISO/IEC 23008-1:2017, *Information technology — High efficiency coding and media delivery in heterogeneous environments — Part 1: MPEG media transport (MMT)*

ISO/IEC 23008-2:2017, *Information technology — High efficiency coding and media delivery in heterogeneous environments — Part 2: High efficiency video coding*

ISO/IEC 23008-3:2015, *Information technology — High efficiency coding and media delivery in heterogeneous environments — Part 3: 3D audio*

ISO/IEC 23008-12, *Information technology — High efficiency coding and media delivery in heterogeneous environments — Part 12: Image file format*

ISO/IEC 23009-1, *Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats*

ISO/IEC 23091-2¹, *Information technology — Coding-independent code points — Part 2: Video*

ISO/IEC 23091-3, *Information technology — Coding-independent code points — Part 3: Audio*

W3C Recommendation, *TTML profiles for Internet media subtitles and captions 1.0 (IMSC1)*

WebVTT: *The web video text tracks format*, W3C (Working Draft, 08 August 2017)

W3C Recommendation, *XML schema part 1: Structures*

W3C Recommendation, *XML schema part 2: Datatypes*

IETF BCP 47, *Tags for Identifying Languages*

IETF RFC 6381, *MIME Codecs and Profiles*

3 Terms, definitions, abbreviated terms, and conventions

3.1 Terms and definitions

For the purposes of this document, the terms and definitions in ISO/IEC 14496-12, ISO/IEC 23008-12, ISO/IEC 23009-1 and the following apply. If terms defined in ISO/IEC 14496-12, ISO/IEC 23008-12 and ISO/IEC 23009-1 are also defined in this document, the definitions in this document are applicable.

NOTE In particular, the terms coded image, coded image item, derived image, derived image item, image item, reconstructed image, and source image item are defined in ISO/IEC 23008-12.

The terminological databases for use in standardization maintained by ISO and IEC at the following addresses:

- IEC Electropedia: available at <http://www.electropedia.org/>
- ISO Online browsing platform: available at <http://www.iso.org/obp>

3.1.1

azimuth

first of the two sphere coordinates describing the location of a point on the sphere

Note 1 to entry: Azimuth and elevation are specified in subclause 5.1.

3.1.2

azimuth circle

circle on the sphere connecting all points with the same azimuth value

Note 1 to entry: An azimuth circle is always a *great circle* (3.1.17).

3.1.3

circular image

image captured with a *fish-eye lens* (3.1.14)

3.1.4

closed scheme type

scheme type (3.1.35) that clearly specifies which transformations are allowed and does not allow future extensions

¹ Under preparation. Stage at time of FDIS ballot: ISO/IEC DIS 23091-2, 40.99.

3.1.5**composition-aligned sample**

sample in a track that is associated with another track, the sample has the same composition time as a particular sample in the another track, or, when a sample with the same composition time is not available in the another track, the closest preceding composition time relative to that of a particular sample in the another track

3.1.6**constituent picture**

such part of a spatially frame-packed stereoscopic picture that corresponds to one view, or a picture itself when frame packing is not in use or the temporal interleaving frame packing arrangement is in use

3.1.7**content coverage**

one or more sphere regions that are covered by the content represented by the track or by an image item

3.1.8**elevation**

second of the two sphere coordinates describing the location of a point on the sphere

Note 1 to entry: Azimuth and elevation are specified in subclause 5.1.

3.1.9**elevation circle**

circle on the sphere connecting all points with the same elevation value

Note 1 to entry: When the elevation is zero, an elevation circle is also a *great circle* (3.1.17). This coincides with the equator on Earth.

3.1.10**extractor track**

track that has *untransformed sample entry type* (3.1.44) equal to 'hvc2', 'avc2', or 'avc4' and contains one or more 'scal' track references

3.1.11**field of view**

extent of the observable world in captured/recorded content or in a physical display device

3.1.12**file decoder**

collective term for file/segment decapsulation and decoding of video, audio or image bitstreams

3.1.13**file decoding process**

process specified as a part of a media profile specification that takes as input a set of ISO BMFF tracks or items and derives either of the following:

- decoded pictures or audio samples, and rendering metadata for them;
- a fully rendered audio scene in the reference system

3.1.14**fisheye lens**

wide-angle camera lens that usually captures an approximately hemispherical *field of view* (3.1.11) and projects it as a *circular image* (3.1.3)

3.1.15**fisheye video**

video captured by *fisheye lenses* (3.1.14)

3.1.16**global coordinate axes**

coordinate axes that are associated with audio, video, and images representing the same acquisition position and intended to be rendered together

Note 1 to entry: Coordinate axes are specified in subclause 5.1.

Note 2 to entry: The origin of the global coordinate axes is usually the same as the centre point of a device or rig used for omnidirectional audio/video acquisition as well as the position of the observer's head in the three-dimensional space in which the audio and video tracks are located.

Note 3 to entry: In the absence of the initial viewing orientation metadata (see subclause 7.7.4 for tracks or subclause 7.9.9 for image items), the initial viewing orientation should be inferred to be equal to (0, 0, 0) for (centre_azimuth, centre_elevation, centre_tilt) relative to the global coordinate axes.

3.1.17

great circle

intersection of the sphere and a plane that passes through the centre point of the sphere

Note 1 to entry: A great circle is also known as an orthodrome or Riemannian circle.

Note 2 to entry: The centre of the sphere and the centre of a great circle are co-located.

3.1.18

guard band

area in a *packed picture* (3.1.23) that is not rendered but may be used to improve the rendered part of the packed picture to avoid or mitigate visual artifacts such as seams

Note to entry: Guard bands are associated with *packed regions* (3.1.24) as described in subclause 7.5.3.

3.1.19

local coordinate axes

coordinate axes obtained after applying rotation to the *global coordinate axes* (3.1.16)

3.1.20

OMAF player

collective term for

- file/segment reception or file access;
- file/segment decapsulation;
- decoding of audio, video, image, or timed text bitstreams;
- rendering of audio, images, or timed text; and
- viewport selection

3.1.21

omnidirectional media

media such as image or video and its associated audio that enable rendering according to the user's *viewing orientation* (3.1.45), if consumed with a head-mounted device, or according to user's desired *viewport* (3.1.46), otherwise, as if the user was in the spot where and when the media was captured

3.1.22

open-ended scheme type

scheme type (3.1.35) that allows future extensions

3.1.23

packed picture

picture that is represented as a coded picture in the coded video bitstream

Note 1 to entry: A packed picture may result from *region-wise packing* (3.1.31) of a *projected picture* (3.1.25).

3.1.24

packed region

region in a *packed picture* (3.1.24) that is mapped to a *projected region* (3.1.26) as specified by the *region-wise packing* (3.1.31) signalling

3.1.25

projected picture

picture that has a representation format specified by an omnidirectional video projection format

Note 1 to entry: Omnidirectional projection formats are specified in subclause 5.2.

3.1.26**projected region**

region in a *projected picture* (3.1.25) that is mapped to a *packed region* (3.1.24) as specified by the region-wise packing signalling

3.1.27**projection**

inverse of the process by which the samples of a *projected picture* (3.1.25) are mapped to a set of positions identified by a set of azimuth and elevation coordinates on a unit sphere

3.1.28**quality ranking region**

region that is associated with a quality ranking value and is specified relative to a decoded picture or a sphere

3.1.29**quality ranking 2D region**

quality ranking region (3.1.28) that is specified relative to a decoded picture

3.1.30**quality ranking sphere region**

quality ranking region (3.1.28) that is specified relative to a sphere

3.1.31**region-wise packing**

inverse of the process of transformation, resizing, and relocating of *packed regions* (3.1.24) of a *packed picture* (3.1.23) to remap to *projected regions* (3.1.26) of a *projected picture* (3.1.25)

3.1.32**rendering**

process of generating audio-visual content for playback from the decoded audio-visual data according to the user's *viewing orientation* (3.1.45), if consumed with a head-mounted device, or according to user's desired *viewport* (3.1.46), otherwise

3.1.33**sample**

all the data associated with a single time or single element in one of the three sample arrays that represent a picture

Note 1 to entry: When the term sample is used in the context of a track, it refers to all the data associated with a single time of that track, where a time is either a decoding time or a composition time. When the term sample is used in the context of a picture, e.g., in the phrase "luma sample", it refers to a single element in one of the three sample arrays that represent the picture.

3.1.34**sample entry type**

four-character code that is either the value of the `format` field of a `SampleEntry` directly contained in `SampleDescriptionBox` or the `data_format` value of an instance of `OriginalFormatBox`

3.1.35**scheme type**

type that parameterizes an encrypted, restricted, or incomplete media track

3.1.36**sphere coordinates**

azimuth (ϕ) and *elevation* (θ) that identify a location of a point on the unit sphere

3.1.37**sphere region**

region on a sphere, specified either by four *great circles* (3.1.17) or by two *azimuth circles* (3.1.2) and two *elevation circles* (3.1.9), or such a region on the rotated sphere after applying certain amount of yaw, pitch, and roll rotations

3.1.38**sub-picture**

picture that represents a spatial subset of the original video content, which has been split into spatial subsets before video encoding at the content production side

3.1.39

sub-picture bitstream

bitstream that represents a spatial subset of the original video content, which has been split into spatial subsets before video encoding at the content production side

3.1.40

sub-picture track

track that represents a *sub-picture bitstream* (3.1.39)

3.1.41

tilt angle

angle indicating the amount of tilt of a *sphere region* (3.1.37), measured as the amount of rotation of the sphere region along the axis originating from the sphere origin passing through the centre point of the sphere region, where the angle value increases clockwise when looking from the origin towards the positive end of the axis

3.1.42

time-parallel sample

sample in a track that is associated with another track, and either has the same decoding time as a particular sample in the other track, or, when a sample with the same decoding time in the other track is not available, the closest preceding decoding time relative to that of a particular sample in the other track

3.1.43

track sample entry type

sample entry type (3.1.34) of a `SampleEntry` directly contained in `SampleDescriptionBox`

3.1.44

untransformed sample entry type

track sample entry type (3.1.43) that would apply if no transformations had been performed to a transformed media track

3.1.45

viewing orientation

triplet of azimuth, elevation, and tilt angle characterizing the orientation that a user is consuming the audio-visual content; in case of image or video, characterizing the orientation of the *viewport* (3.1.46)

3.1.46

viewport

region of omnidirectional image or video suitable for display and viewing by the user

3.2 Abbreviated terms

2D	two-dimensional
CICP	coding-independent code points (specified in ISO/IEC 23091-1, 23091-2, and 23091-3)
CMAF	common media application format (specified in ISO/IEC 23000-19)
DASH	MPEG dynamic adaptive streaming over HTTP (specified in ISO/IEC 23009-1)
ERP	equiangular projection
FOV	field of view
ISOBMFF	ISO base media file format (specified in ISO/IEC 14496-12)
HEVC	high efficiency video coding (specified in ISO/IEC 23008-2)
HMD	head-mounted display
HOA	higher-order ambisonics

MCTS	motion-constrained tile set
MMT	MPEG media transport (specified in ISO/IEC 23008-1)
OMAF	omnidirectional media format (specified in ISO/IEC 23090-2)
SRD	spatial relationship description
URN	uniform resource name
VR	virtual reality

3.3 Arithmetic operators and mathematical functions

+	addition
−	subtraction (as a two-argument operator) or negation (as a unary prefix operator)
*	multiplication, including matrix multiplication
x^y	exponentiation, specifies x to the power of y (In other contexts, such notation is used for superscripting not intended for interpretation as exponentiation.)
/	integer division with truncation of the result toward zero EXAMPLE $7 / 4$ and $-7 / -4$ are truncated to 1 and $-7 / 4$ and $7 / -4$ are truncated to −1.
÷	denotes division in mathematical equations where no truncation or rounding is intended
$\frac{x}{y}$	denotes division in mathematical equations where no truncation or rounding is intended
$\sum_{i=x}^y f(i)$	summation of $f(i)$ with i taking all integer values from x up to and including y
$x \% y$	modulus (Remainder of x divided by y, defined only for integers x and y with $x \geq 0$ and $y > 0$.)
$\text{Asin}(x)$	trigonometric inverse sine function, operating on an argument x that is in the range of −1.0 to 1.0, inclusive, with an output value in the range of $-\pi/2$ to $\pi/2$, inclusive, in units of radians
$\text{Atan}(x)$	trigonometric inverse tangent function, operating on an argument x that is any real number, with an output value in the range of $-\pi/2$ to $\pi/2$, inclusive, in units of radians
$\text{Atan2}(y, x)$	$= \begin{cases} \text{Atan}\left(\frac{y}{x}\right) & ; \text{ if } x > 0 \\ \text{Atan}\left(\frac{y}{x}\right) + \pi & ; \text{ if } x < 0 \text{ \& \& } y \geq 0 \\ \text{Atan}\left(\frac{y}{x}\right) - \pi & ; \text{ if } x < 0 \text{ \& \& } y < 0 \\ +\frac{\pi}{2} & ; \text{ if } x == 0 \text{ \& \& } y \geq 0 \\ -\frac{\pi}{2} & ; \text{ otherwise} \end{cases}$
$\text{Cos}(x)$	trigonometric cosine function operating on an argument x in units of radians

Floor(x) largest integer less than or equal to x

Round(x) = Sign(x) * Floor(Abs(x) + 0.5)

$$\text{Sign}(x) = \begin{cases} 1 & ; x > 0 \\ 0 & ; x == 0 \\ -1 & ; x < 0 \end{cases}$$

Sin(x) trigonometric sine function operating on an argument x in units of radians

Tan(x) trigonometric tangent function operating on an argument x in units of radians

3.4 Order of operation precedence

The following notation is used to specify a range of values:

When order of precedence in an expression is not indicated explicitly by use of parentheses, the following rules apply:

- Operations of a higher precedence are evaluated before any operation of a lower precedence.
- Operations of the same precedence are evaluated sequentially from left to right.

Table 1 specifies the precedence of operations from highest to lowest; a higher position in the table indicates a higher precedence.

NOTE For those operators that are also used in the C programming language, the order of precedence used in this document is the same as used in the C programming language.

Table 1 – Operation precedence from highest (at top of table) to lowest (at bottom of table)

operations (with operands x, y, and z)
"x++", "x--"
"!x", "-x" (as a unary prefix operator)
x^y
"x * y", "x / y", "x ÷ y", " $\frac{x}{y}$ ", "x % y"
"x + y", "x - y" (as a two-argument operator), " $\sum_{i=x}^y f(i)$ "
"x << y", "x >> y"
"x < y", "x <= y", "x > y", "x >= y"
"x == y", "x != y"
"x & y"
"x y"
"x && y"
"x y"
"x ? y : z"
"x..y"
"x = y", "x += y", "x -= y"

3.5 Range notation

The following notation is used to specify a range of values:

$x = y..z$ x takes on integer values starting from y to z , inclusive, with x , y , and z being integer numbers and z being greater than y

3.6 Variables

This document derives variables that are named by a mixture of lower case and upper case letter and without any underscore characters. Variables starting with an upper case letter are derived for the current syntax structure and all depending syntax structures. Variables starting with an upper case letter may be used in the specification for dependent syntax structures without mentioning the originating syntax structure of the variable. Variables starting with a lower case letter are only used within the clause in which they are derived.

3.7 Processes

Processes are used to describe the decoding of syntax elements. A process has a separate specification and invoking. All syntax elements and upper case variables that pertain to the current syntax structure and depending syntax structures are available in the process specification and invoking. A process specification may also have a lower case variable explicitly specified as input. Each process specification has explicitly specified an output. The output is a variable that can either be an upper case variable or a lower case variable.

When invoking a process, the assignment of variables is specified as follows:

- If the variables at the invoking and the process specification do not have the same name, the variables are explicitly assigned to lower case input or output variables of the process specification.
- Otherwise (the variables at the invoking and the process specification have the same name), assignment is implied.

4 Overview

4.1 Organization of this document

This document is organized as follows:

- Subclauses 4.2 and 4.3 provide the overall architecture for projected omnidirectional video and fisheye omnidirectional video, respectively.
- Subclause 4.4 describes which types of referenceable features specified in this document implementations and external specifications may be used to achieve interoperability. It also defines OMAF media and presentation profiles.
- Clause 5 specifies a coordinate system used in this document and the equations for the equirectangular and cubemap omnidirectional projection formats, the conversion from the local coordinate axes to the global coordinate axes, and the rectangular region-wise packing.
- Clause 6 specifies syntax structures that are common for fisheye video and fisheye images.
- Clause 7 specifies extensions to the ISO base media file format for omnidirectional media as well as for timed metadata for sphere regions. It also specifies generic extensions to ISO/IEC 14496-12 and ISO/IEC 14496-15, which may be used also for other purposes than for omnidirectional media.
- Clause 8 specifies extensions to DASH for omnidirectional media.
- Clause 9 specifies extensions to MMT for omnidirectional media.
- Clause 10 specifies OMAF media profiles.
- Clause 11 specifies OMAF presentation profiles based on some of the OMAF media profiles specified in Clause 10.
- Annex A contains the OMAF DASH XML schema.
- Annex B specifies the DASH integration of all the OMAF media profiles for timed media specified in Clause 10.

- Annex C specifies the CMAF integration of some of the OMAF media profiles specified in Clause 10.
- Annex D describes some informative schemes for viewport-dependent omnidirectional video processing.
- Annex E contains some informative DASH MPD examples.
- Annex F contains some informative MMT signalling examples.

4.2 Overall architecture for omnidirectional media with projected video

4.2.1 Overview

Figure 1 shows a typical content flow process for an omnidirectional media application with projected video.

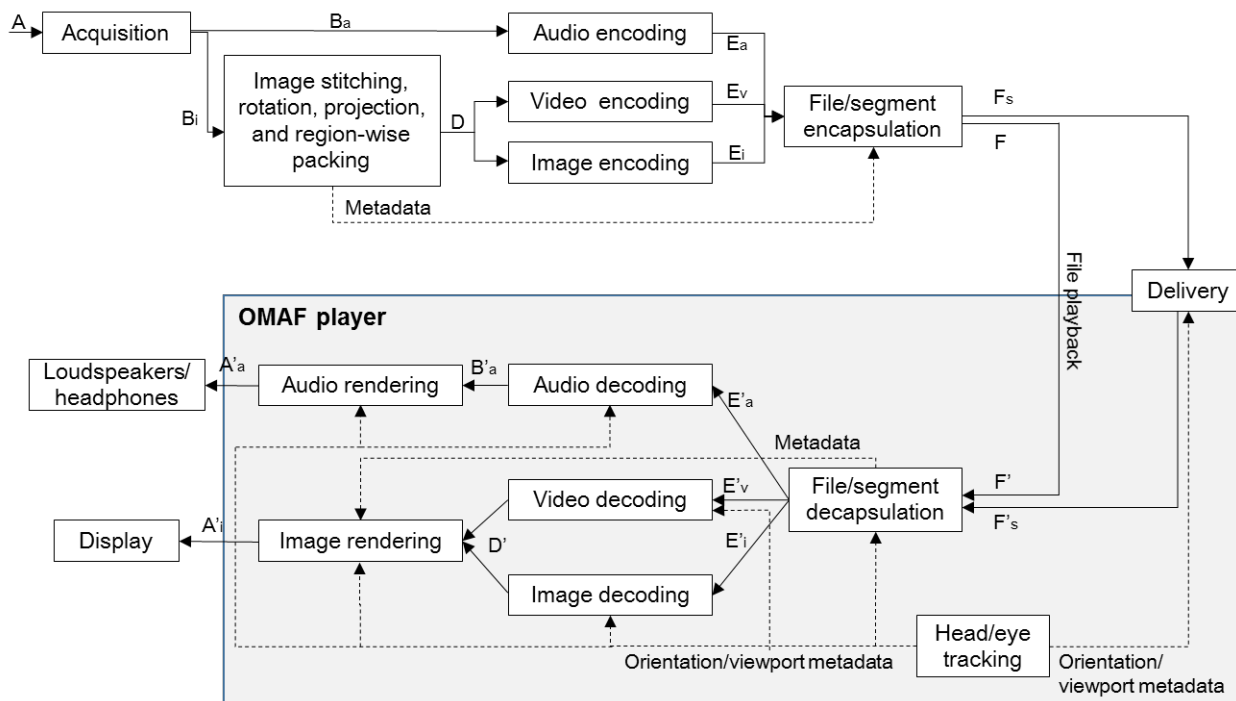


Figure 1 – Content flow process for omnidirectional media with projected video

The following interfaces are normatively specified in this document:

- E'_a, E'_v, E'_i: audio bitstream, video bitstream, coded image(s), respectively; see Clause 10.
- F/F': media file, including projection and region-wise packing metadata; see Clause 7. Moreover, media profiles specified in Clause 10 include the specification of the track formats for F/F', which may contain constraints on the elementary streams contained within the samples of the tracks.
- Clause 8 specifies the delivery related interfaces for DASH delivery.
- Clause 9 specifies the delivery related interfaces for MMT delivery.

The other interfaces in Figure 1 are not specified normatively in this document.

NOTE While the syntax and semantics of the bitstreams E_a, E_v, and E_i are the same as those for E'_a, E'_v, E'_i, respectively, the input interface to the file/segment encapsulation module is not normatively specified.

A real-world audio-visual scene (A) is captured by audio sensors as well as a set of cameras or a camera device with multiple lenses and sensors. The acquisition results in a set of digital image/video (B_i) and audio (B_a) signals. The cameras/lenses typically cover all directions around the centre point of the camera set or camera device, thus the name of 360-degree video.

Audio may be captured using many different microphone configurations and stored as several different content formats, including channel-based signals, static or dynamic (i.e. moving through the 3D scene) object signals, and scene-based signals (e.g., higher order ambisonics). The channel-based signals typically conform to one of the loudspeaker layouts defined in ISO/IEC 23091-3. In an omnidirectional media application, the loudspeaker layout signals of the rendered immersive audio program are binaraulized for presentation via headphones.

The images (B_i) of the same time instance are stitched, possibly rotated, projected, and mapped onto a packed picture (D). A description of this process is provided in subclause 4.2.2.

For audio, the stitching process is not needed, since the captured signals are inherently immersive and omnidirectional.

The packed pictures (D) are encoded as coded images (E_i) or a coded video bitstream (E_v). The captured audio (B_a) is encoded as an audio bitstream (E_a). The coded images, video, and/or audio are then composed into a media file for file playback (F) or a sequence of an initialization segment and media segments for streaming (F_s), according to a particular media container file format. In this document, the media container file format is the ISO base media file format specified in ISO/IEC 14496-12. The file encapsulator also includes metadata into the file or the segments, such as projection and region-wise packing information assisting in rendering the decoded packed pictures.

The metadata in the file includes information for the following:

- the projection format of the projected picture,
- the area of the spherical surface covered by the packed picture,
- the rotation for converting between the global coordinate axes and the local coordinate axes,
- region-wise packing, and
- region-wise quality ranking.

The segments F_s are delivered using a delivery mechanism to a player.

The file that the file encapsulator outputs (F) is identical to the file that the file decapsulator inputs (F'). A file decapsulator processes the file (F') or the received segments (F'_s) and extracts the coded bitstreams (E'_a , E'_v , and/or E'_i) and parses the metadata. The audio, video, and/or images are then decoded into decoded signals (B'_a for audio, and D' for images/video). The decoded packed pictures (D') are projected onto the screen of a head-mounted display or any other display device based on the current viewing orientation or viewport and the projection, spherical coverage, rotation, and region-wise packing metadata parsed from the file. Likewise, decoded audio (B'_a) is rendered, e.g. through headphones, according to the current viewing orientation. The current viewing orientation is determined by the head tracking and possibly also eye tracking functionality. Besides being used by the renderer to render the appropriate part of decoded video and audio signals, the current viewing orientation may also be used by the video and audio decoders for decoding optimization.

The process described above is applicable to both live and on-demand use cases.

4.2.2 Stitching, rotation, projection, and region-wise packing

For monoscopic 360-degree video, the source images of one time instance are stitched, possibly rotated, and projected to generate a projected picture representing one view. The breakdown of the image stitching, rotation, projection, and region-wise packing processes for monoscopic content is illustrated with Figure 2 and described as follows. Source images (B_i) are stitched, possibly rotated, and projected onto a unit sphere.

The image data on the unit sphere is further arranged onto a two-dimensional projected picture (C). The projected picture covers the entire sphere. Optionally, region-wise packing is then applied to map the projected picture onto a packed picture. If the region-wise packing is not applied, the packed picture is identical to the projected picture, and this picture is given as input to image/video encoding. Otherwise, projected regions are mapped onto a packed picture (D) by indicating the location, shape, and size of each packed region in the packed picture, and the packed picture (D) is given as input to image/video encoding. The packed picture may cover only a part of the entire sphere. In practice, the source images may be converted to a packed picture in one process without intermediate steps.

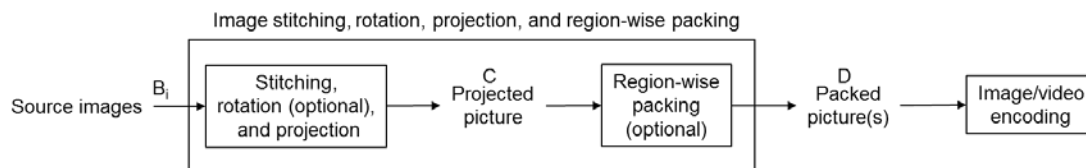


Figure 2 – Monoscopic image stitching, rotation, projection, and region-wise packing of source images of a single acquisition time instance

In the case of stereoscopic 360-degree video, the source images of one time instance are stitched, possibly rotated, and projected to generate a projected picture representing two views, one for each eye. Both views may be mapped onto the same packed picture, as described below in relation to Figure 3, and encoded by a traditional 2D video encoder. Alternatively, each view of the projected picture may be mapped to its own packed picture, in which case the image stitching, projection, and region-wise packing is like described above with Figure 2. A sequence of packed pictures of either the left view or the right view may be independently coded. Alternatively, packed pictures of the left view and the right view may be temporally interleaved in an alternating manner and encoded as a single bitstream, which is indicated to comply with the temporal interleaving frame packing arrangement. In another alternative, multiview coding is used to encode both views into the same bitstream with inter-view prediction.

The breakdown of the image stitching, rotation, projection, and region-wise packing processes for stereoscopic content where both views are mapped onto the same packed picture is illustrated with Figure 3 and described as follows. Source images (B_i) are stitched, possibly rotated, and projected onto two unit spheres, one for each eye. The image data on each unit sphere is further arranged onto a two-dimensional projected picture (C_L for left eye, C_R for right eye), which covers the entire sphere. Frame packing is applied to pack the left view picture and right view picture onto the same projected picture. Optionally, region-wise packing is then applied to pack the projected picture onto a packed picture, and the packed picture (D) is given as input to image/video encoding. If the region-wise packing is not applied, the packed picture is identical to the projected picture, and this picture is given as input to image/video encoding. The packed picture may cover only a part of the entire sphere.

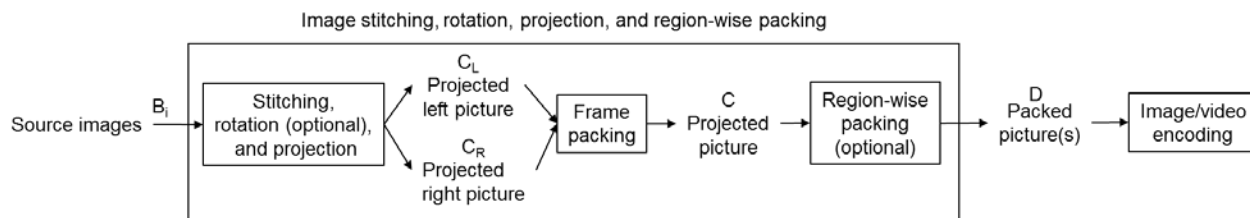


Figure 3 – Stereoscopic image stitching, rotation, projection, and region-wise packing of source images of a single acquisition time instance

The image stitching, rotation, projection, and region-wise packing processes may be carried out multiple times for the same source images to create different versions of the same content, e.g., for different sphere rotations (i.e., different rotations of local coordinate axes with respect to the global coordinate axes). Similarly, the region-wise packing process may be performed multiple times from the same projected picture to create more than one sequence of packed pictures to be encoded.

4.3 Overall architecture for omnidirectional media with fisheye video

Figure 4 shows a typical content flow process for an omnidirectional media application with fisheye video.

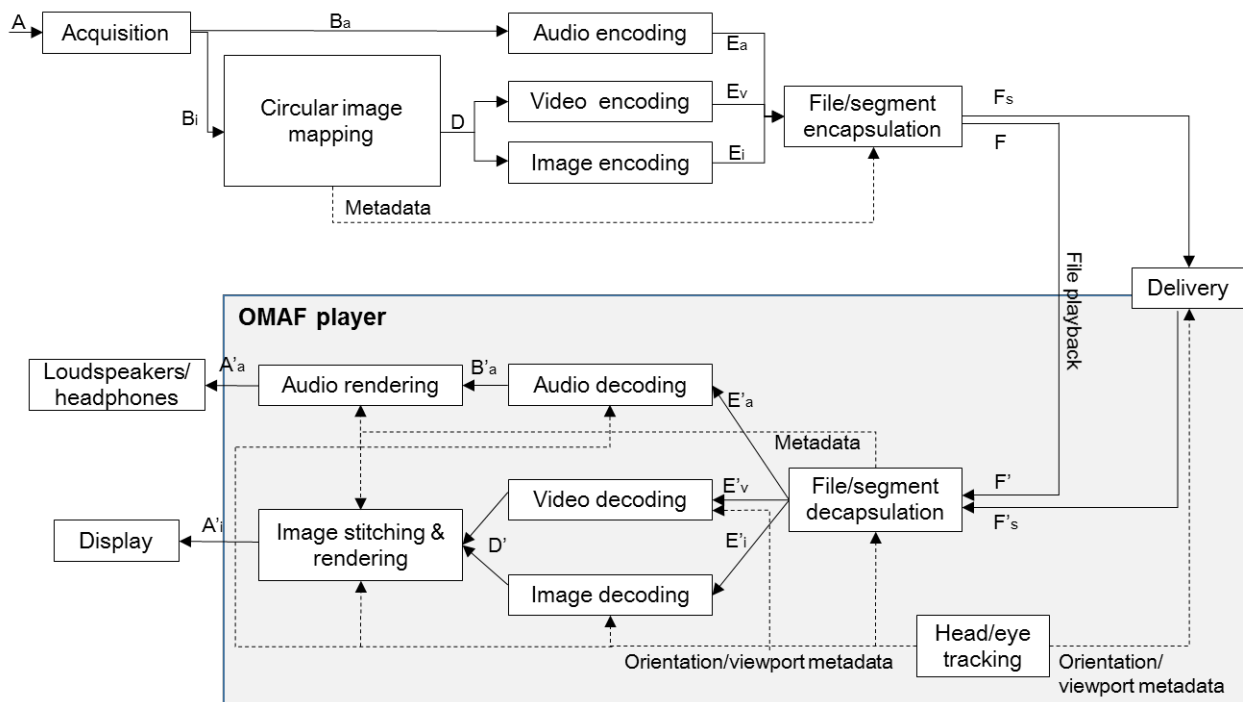


Figure 4 – Content flow process for omnidirectional media with fisheye video

A fisheye lens is a lens that achieves extremely wide angles of view and produces a circular image. Circular images captured by multiple fisheye lenses of a 360 camera typically cover all directions around the centre point of the camera set or camera device.

The circular images (B_i) captured by fisheye lenses at the same time instance are not stitched, but directly mapped onto a picture (D) in the fisheye video format, which is specified in clause 6. Region-wise packing is not allowed for the fisheye video format.

The fisheye images including non-stitched circular images (D) are encoded as coded images (E_i) or a coded video bitstream (E_v). The coded images, video, and/or audio are then composed into a media file for file playback (F) or a sequence of an initialization segment and media segments for streaming (F_s), according to a particular media container file format. The file encapsulator also includes fisheye-specific metadata, which assist in rendering the decoded pictures, into the file or the segments.

The fisheye-specific metadata in the file includes:

- lens distortion correction (LDC) parameters with local variation of FOV,
- lens shading compensation (LSC) parameters with RGB gains,
- displayed field of view information, and
- camera extrinsic parameters (physical location).

In an OMAF player, the circular images in the decoded fisheye video are stitched using the signalled fisheye-specific parameters to an omnidirectional image and rendered according to the user's viewport.

4.4 Conformance and interoperability

4.4.1 General

This document specifies several referenceable features that implementations and external specifications may use to achieve interoperability. These referenceable features include the following:

- Specification of a coordinate system for omnidirectional media (subclause 5.1), and equations for a conversion process between sets of coordinate axes of this coordinate system (subclause 5.3).

- Omnidirectional projection formats (subclause 5.2).
- Media profiles, as defined in subclause 4.4.2. Individual media profiles are specified in Clause 10.
- Presentation profiles, as defined in subclause 4.4.3. Individual presentation profiles are specified in Clause 11.
- Mapping of media and presentation profiles to DASH and CMAF are specified in Annex B and Annex C, respectively.
- Region-wise packing formats (subclause 5.4).
- Scheme types for post-decoder processing of omnidirectional video (subclause 7.6.1).
- Specific types of sphere region timed metadata tracks (subclause 7.7).

However, the technologies and features specified in this document may be used independently. For example, boxes specified in this document may be used independently of other OMAF features.

This document specifies code points that implementations and external specifications may use for referring to respective OMAF features. The code points are summarized in subclause 4.4.4.

4.4.2 Media profiles

4.4.2.1 General

A media profile for timed media is defined as requirements and constraints for a set of one or more ISOBMFF tracks of a single media type. The conformance of a set of one or more ISOBMFF tracks to a media profile is specified as a combination of:

- Specification of which sample entry type(s) are allowed, and which constraints and extensions are required in addition to those imposed by the sample entry type(s).
- Constraints on the samples of the tracks, typically expressed as constraints on the elementary stream contained within the samples of the tracks.

A media profile for static media is defined as requirements and constraints for a set of one or more ISOBMFF items of a single media type. The conformance of a set of one or more ISOBMFF items to a media profile is specified as a combination of:

- Specification of which item type(s) are allowed, and which constraints and extensions are required in addition to those imposed by the item type(s).
- Constraints on the content of the items, typically expressed as constraints on the elementary stream contained within the items.

The elementary stream constraints of a media profile may be indicated by a requirement to comply with a certain profile and level of the media coding specification, possibly including additional constraints and extensions, such as a requirement of the presence of certain information for rendering and presentation.

NOTE The use of the elementary stream constraints specified for a media profile could be referenced by implementations and external specifications independently of the ISOBMFF requirements and constraints specified for the media profile. However, the encapsulation and use of a media profile for other encapsulation formats is outside of the scope of this document.

4.4.2.2 File decoding process and file decoder requirements for video and image media profiles

Each video or image media profile specified in clause 10 includes a file decoding process such that all file decoders that conform to the video or image media profile will produce numerically identical cropped decoded pictures when invoking the file decoding process associated with that video or image media profile for a set of ISOBMFF tracks conforming to the video media profile or a set of ISOBMFF image items conforming to the image media profile, respectively. A bitstream that conforms to the elementary stream constraints specified for the video or image media profile is reconstructed as an intermediate product of the file decoding process. The output of the file decoding process consists of all of the following:

- a list of decoded pictures;
- for projected omnidirectional video or image, all of the following rendering metadata:
 - the projection format of the associated projected pictures,

- the frame packing format of the associated projected pictures (when applicable),
- the region-wise packing information (when applicable),
- the rotation information (when applicable);
- for fisheye omnidirectional video or image, fisheye-specific rendering metadata.

Video media profiles may specify constraints on when rendering metadata is allowed to change.

A file decoder conforms to the file decoding process requirements of this document when it complies with both of the following:

- The file decoder includes a conforming decoder that produces numerically identical cropped decoded pictures to those produced by the file decoding process specified for the video or image media profile in Clause 10 (with the correct output order or output timing, as specified in the video or image coding specification of the video or image media profile, respectively).
- The file decoder outputs rendering metadata that is equivalent to that produced by the file decoding process specified for the video or image media profile in Clause 10 (with the correct association of the rendering metadata to particular cropped decoded pictures, as specified in this document).

Figure 5 illustrates an example file decoder, its outputs as described above, and its relation to other parts of an OMAF player implementation.

A player claiming conformance to a video or image media profile shall include a file decoder complying with the file decoding process of that video or image media profile as specified above. While the player operation, with the exception of the file decoding process, is not specified normatively in this document, specifications of a media profile may include an informative clause on expectations of a player operation, for example including recommendations for rendering.

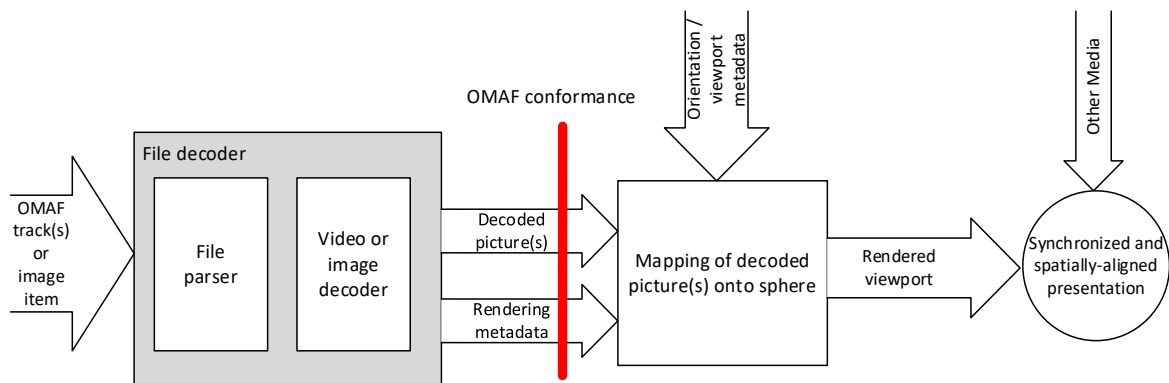


Figure 5 – Conformance points for OMAF video and image media profiles

4.4.2.3 File decoding process and file decoder requirements for audio media profiles

Each audio profile specified in Clause 10 includes a file decoding process such that all file decoders that conform to the audio profile will produce an output according to the specification of the file decoding process. For audio, the conformance of the output signal is defined by the file decoding process. The output of the file decoding process provides a signal that may be represented in the reference system.

4.4.3 Presentation profiles

A presentation profile is defined as requirements and constraints for an ISO BMFF file containing tracks or items of any number of media types. A specification of a presentation profile should refer to the specified media profiles and may include additional requirements or constraints. A file conforming to a presentation profile typically provides an omnidirectional audio-visual experience.

4.4.4 Summary of referenceable code points

4.4.4.1 Brands

ISO/IEC 14496-12 defines the concept of brands, which may be indicated in the `FileTypeBox`. This document specifies the `TrackTypeBox`, which may be used to indicate brands in a track-specific manner. Brands are used in this document to indicate conformance to OMAF media profiles, OMAF presentation profiles, and CMAF Media Profiles that are defined in this document.

The brands specified in this document are listed in Table 2.

Table 2 – Brands specified in this document

Brand identifier	Clause in this document	Name of the media profile or the presentation profile
hevi	10.1.2	HEVC-based viewport-independent OMAF video profile
hevd	10.1.3	HEVC-based viewport-dependent OMAF video profile
avde	10.1.4	AVC-based viewport-dependent OMAF video profile
oabl	10.2.2	OMAF 3D audio baseline profile
oa2d	10.2.3	OMAF 2D audio legacy profile
heoi	10.3.3	OMAF HEVC image profile
jpoi	10.3.4	OMAF legacy image profile
ttml	10.4.2	OMAF IMSC1 timed text profile
ttwv	10.4.3	OMAF WebVTT timed text profile
ompp	11.1	OMAF viewport-independent baseline presentation profile
ovdp	11.2	OMAF viewport-dependent baseline presentation profile
cvid	C.1.1	CMAF media profile for the HEVC-based viewport-independent OMAF video profile
chev	C.1.2	CMAF media profile for the HEVC-based viewport-dependent OMAF video profile
cabl	C.2.1	CMAF media profile for OMAF 3D audio baseline profile

4.4.4.2 Uniform resource names

The URNs specified in this document are listed in Table 3.

Table 3 – URNs specified in this document

URN	Clause in this document	Informative description
urn:mpeg:mpegI:omaf:2017	8.2.2	Namespace for the XML elements and attributes specified in this document
urn:mpeg:mpegI:omaf:2017:pf	8.3.2	Scheme identifier for the omnidirectional projection type DASH MPD descriptor
urn:mpeg:mpegI:omaf:2017:rwpk	8.3.3	Scheme identifier for the region-wise packing type DASH MPD descriptor
urn:mpeg:mpegI:omaf:2017:cc	8.3.4	Scheme identifier for the content coverage DASH MPD descriptor
urn:mpeg:mpegI:omaf:2017:srqr	8.3.5	Scheme identifier for the spherical region-wise quality ranking DASH MPD descriptor
urn:mpeg:mpegI:omaf:2017:2dqr	8.3.6	Scheme identifier for the 2D region-wise quality ranking DASH MPD descriptor
urn:mpeg:mpegI:omaf:2017:fomv	8.3.7	Scheme identifier for the fisheye omnidirectional video DASH MPD descriptor

4.4.4.3 Restricted scheme types

The restricted scheme types specified in this document are listed in Table 4.

Table 4 – Restricted scheme types specified in this document

Restricted scheme type	Clause in this document	Informative description
podv	7.6.1.2	Projected omnidirectional video; an open-ended restricted scheme type for omnidirectional video based on a projection.
erpv	7.6.1.3	Equirectangular projected video; a closed restricted scheme type for omnidirectional video based on the equirectangular projection and essentially no region-wise packing.
ercm	7.6.1.4	Packed equirectangular or cubemap projected video; a closed restricted scheme type for omnidirectional video based on either the equirectangular projection or the cubemap projection and any region-wise packing.
fodv	7.6.1.5	Fisheye omnidirectional video; an open-ended restricted scheme type for omnidirectional video based on video captured by one or more fisheye camera lenses and not based on a projection.

4.4.4.4 Sample entry types

The sample entry types specified in this document are listed in Table 5.

Table 5 – Sample entry types specified in this document

Sample entry type	Clause in this document	Informative description
invo	7.7.4	For use with a timed metadata track containing the initial viewing orientation timed metadata.
rcvp	7.7.5	For use with a timed metadata track containing the recommended viewport timed metadata.
ttsl	7.7.6	For use with a timed metadata track containing the timed text sphere location timed metadata.

4.4.4.5 Box types

The box types specified in this document are listed in Table 6.

Table 6 – Box types specified in this document

Box type	Clause in this document	Box name
csch	7.1.6	Compatible scheme type box
ttyp	7.1.9	Track type box
povd	7.6.2	Projected omnidirectional video box
prfr	7.6.2, 7.9.3	Projection format box, projection format item property
fodv	7.6.3	Fisheye omnidirectional video box
fovi	7.6.3, 7.9.4	Fisheye video essential information box, essential fisheye image item property
fvsi	7.6.3, 7.9.5	Fisheye video supplemental information box, supplemental fisheye image item property
rwpk	7.6.4, 7.9.6	Region-wise packing box, region-wise packing item property
rotn	7.6.5, 7.9.7	Rotation box, rotation item property
covi	7.6.6, 7.9.8	Coverage information box, coverage information item property
rosc	7.7.2	Sphere region configuration box
rvif	7.7.5	Recommended viewport information box
srqr	7.8.2	Sphere region quality ranking box

Box type	Clause in this document	Box name
2dqr	7.8.3	2D region quality ranking box
stvi	7.9.2	Frame packing item property NOTE <code>FramePackingProperty</code> has the same box type and syntax as <code>StereoVideoBox</code> specified in ISO/IEC 14496-12.
iivo	7.9.9	Initial viewing orientation item property
otcf	7.10.2	OMAF timed text configuration box

4.4.4.6 Track grouping types

The track grouping types specified in this document are listed in Table 7.

Table 7 – Track grouping types specified in this document

Track grouping type	Clause in this document	Track grouping name
ster	7.1.1	Stereoscopic video track grouping
alte	7.2.1	Alternative extraction source track grouping

5 Omnidirectional video projection and region-wise packing

5.1 Coordinate system

The coordinate system consists of a unit sphere and three coordinate axes, namely the X (back-to-front) axis, the Y (lateral, side-to-side) axis, and the Z (vertical, up) axis, where the three axes cross at the centre of the sphere.

The location of a point on the sphere is identified by a pair of sphere coordinates azimuth (ϕ) and elevation (θ).

Figure 6 specifies the relation of the sphere coordinates azimuth (ϕ) and elevation (θ) to the X, Y, and Z coordinate axes.

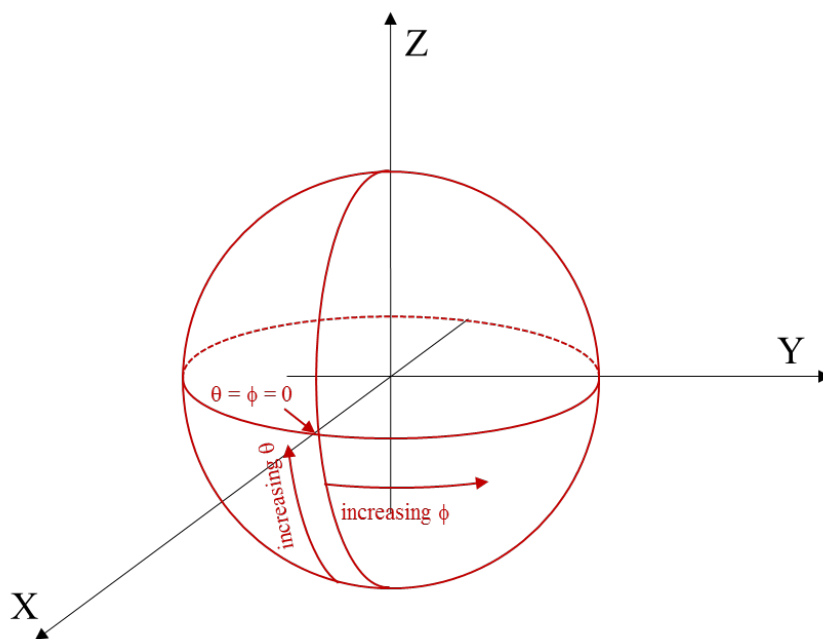


Figure 6 – Coordinate axes and their relation to the sphere coordinates

NOTE The specified coordinate system is the same as in ISO/IEC 23008-3.

The value ranges of azimuth is -180.0 , inclusive, to 180.0 , exclusive, degrees. The value range of elevation is -90.0 to 90.0 , inclusive, degrees.

5.2 Omnidirectional projection formats

5.2.1 General

Subclause 5.2 specifies the inverse of the projection process for remapping of one sample location of a monoscopic projected luma picture onto a position on the unit sphere identified by a pair of azimuth and elevation coordinates.

The omnidirectional projection formats specified in this document are identified by a 5-bit unsigned integer value. Table 8 specifies the omnidirectional projection format identifier values and provides references to the clauses in which the respective inverse projection processes are specified.

Table 8 – Omnidirectional projection formats

Identifier value	Omnidirectional projection	Clause
0	Equirectangular projection	5.2.2
1	Cubemap projection	5.2.3
2..31	Reserved	N/A

5.2.2 Equirectangular projection for one sample location

Inputs to this process are:

- pictureWidth and pictureHeight, which are the width and height, respectively, of a monoscopic projected luma picture, in relative projected picture sample units, and

- the centre point of a sample location (hPos, vPos) along the horizontal and vertical axes, respectively, where hPos and vPos are in relative projected picture sample units and may have non-integer real values.

Outputs of this process are:

- sphere coordinates (ϕ , θ) for the sample location in degrees relative to the coordinate axes specified in subclause 5.1.

The sphere coordinates (ϕ , θ) for the luma sample location, in degrees, are given by the following equations:

$$\phi = (0.5 - \text{hPos} \div \text{pictureWidth}) * 360$$

$$\theta = (0.5 - \text{vPos} \div \text{pictureHeight}) * 180$$

Figure 7 illustrates the azimuth and elevation ranges of a monoscopic projected picture with the equirectangular projection.

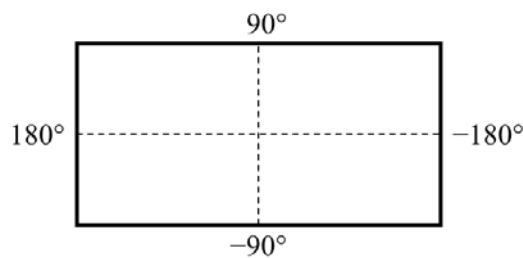


Figure 7 – Azimuth and elevation ranges of the monoscopic projected picture of the equirectangular projection

NOTE 1 Since an input to this process is the centre point of a sample location and the width and height of the sample are non-zero, the output ϕ is never equal to -180° or 180° and the output θ is never equal to -90° or 90° .

NOTE 2 The monoscopic projected picture represents the inside surface of the unit sphere observed from the origin of the coordinate system. Thus, the azimuth decreases from left to right.

5.2.3 Cubemap projection for one sample location

Inputs to this process are:

- pictureWidth and pictureHeight, which are the width and height, respectively, of a monoscopic projected luma picture, in relative projected picture sample units, and
- the centre point of a sample location (hPos, vPos) along the horizontal and vertical axes, respectively, where hPos and vPos are in relative projected picture sample units and may have non-integer real values.

Outputs of this process are:

- sphere coordinates (ϕ , θ) for the sample location in degrees relative to the coordinate axes specified in subclause 5.1.

Figure 8 illustrates the cube face arrangement in the projected picture of the cubemap projection format and the mapping of the cube faces onto the coordinate axes specified in subclause 5.1, where PX, NX, PY, NY, PZ, and NZ denote positive X, negative X, positive Y, negative Y, positive Z, and negative Z, respectively.

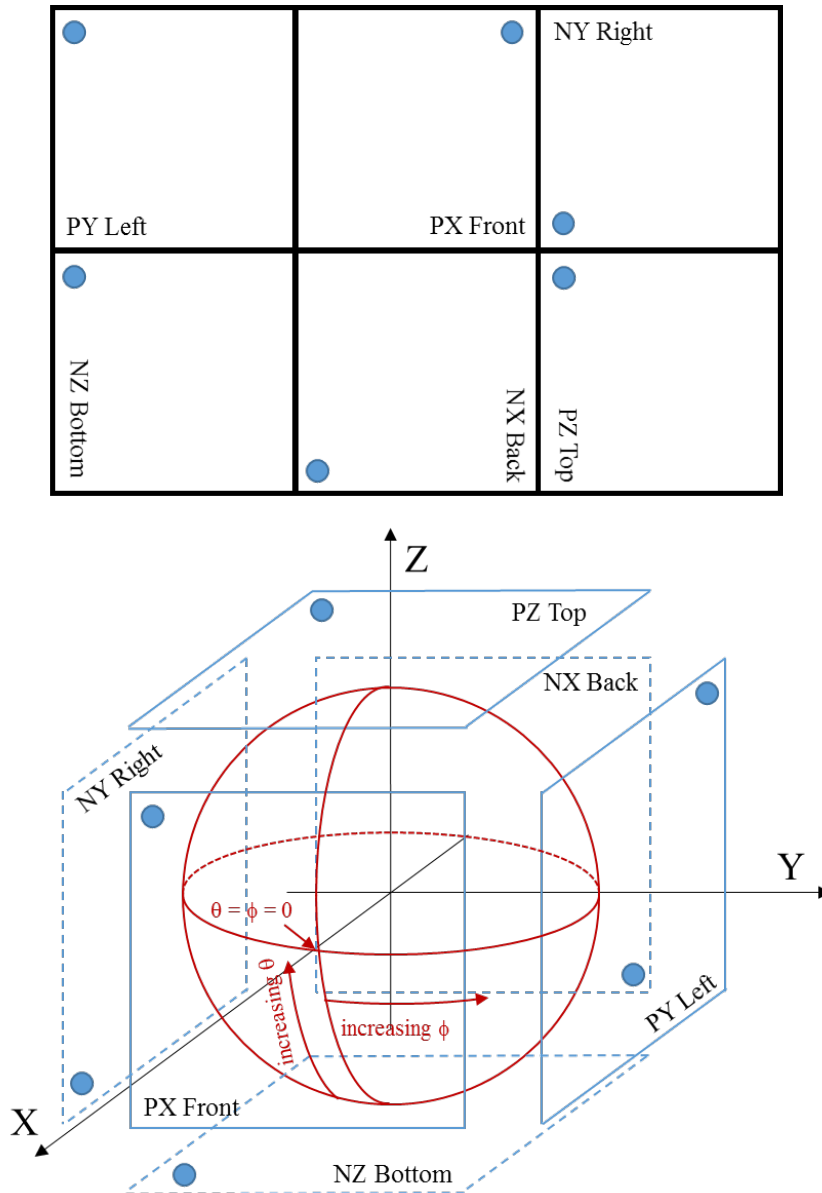


Figure 8 – Relation of the cube face arrangement of the projected picture to the sphere coordinates

The values of pictureWidth and pictureHeight shall be such that pictureWidth is a multiple of 3, pictureHeight is a multiple of 2, and pictureWidth / 3 is equal to pictureHeight / 2.

The sphere coordinates (ϕ , θ) for the luma sample location, in degrees, are given by the following equations:

```

lw = pictureWidth / 3
lh = pictureHeight / 2
tmpHorVal = hPos - Floor( hPos ÷ lw ) * lw
tmpVerVal = vPos - Floor( vPos ÷ lh ) * lh
hPos' = -( 2 * tmpHorVal ÷ lw ) + 1
vPos' = -( 2 * tmpVerVal ÷ lh ) + 1
w = Floor( hPos ÷ lw )
h = Floor( vPos ÷ lh )
if( w == 1 && h == 0 ) { // positive x front face
    x = 1.0
    y = hPos'
    z = vPos'
} else if( w == 1 && h == 1 ) { // negative x back face

```

```

    x = -1.0
    y = -vPos'
    z = -hPos'
} else if( w == 2 && h == 1 ) { // positive z top face
    x = -hPos'
    y = -vPos'
    z = 1.0
} else if( w == 0 && h == 1 ) { // negative z bottom face
    x = hPos'
    y = -vPos'
    z = -1.0
} else if( w == 0 && h == 0 ) { // positive y left face
    x = -hPos'
    y = 1.0
    z = vPos'
} else { // ( w == 2 && h == 0 ), negative y right face
    x = hPos'
    y = -1.0
    z = vPos'
}
 $\phi = \text{Atan2}(y, x) * 180 \div \pi$ 
 $\theta = \text{Asin}(z \div \sqrt{x^2 + y^2 + z^2}) * 180 \div \pi$ 

```

5.3 Conversion from the local coordinate axes to the global coordinate axes

Inputs to this process are:

- rotation_yaw (α_d), rotation_pitch (β_d), rotation_roll (γ_d), all in units of degrees, where rotation_yaw (α_d) and rotation_roll (γ_d), are in the range of -180.0 , inclusive, to 180.0 , exclusive, and rotation_pitch (β_d) is in the range of -90.0 to 90.0 , inclusive, and
- sphere coordinates (ϕ_d, θ_d) relative to the local coordinate axes.

Outputs of this process are:

- sphere coordinates (ϕ', θ') in degrees relative to the global coordinate axes.

This process specifies rotations around the three axes of the coordinate system of subclause 5.1, where yaw (α_d) expresses a rotation around the Z axis, pitch (β_d) rotates around the Y axis, and roll (γ_d) rotates around the X axis. Rotations are extrinsic, i.e., around X, Y, and Z fixed reference axes. The angles increase clockwise when looking from the origin towards the positive end of an axis, as illustrated in Figure 9.

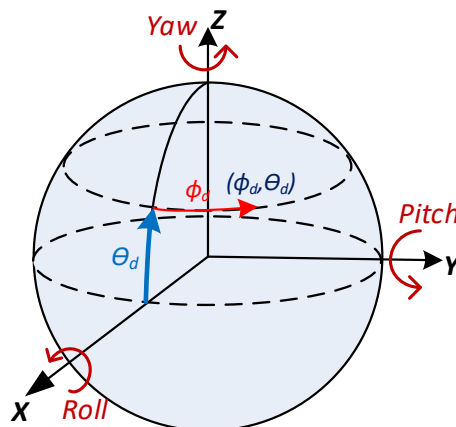


Figure 9 – Illustration of the directions of the yaw, pitch and roll rotations

When any of the yaw (α_d), pitch (β_d) and roll (γ_d) rotation angles is not equal to zero, an OMAF player needs to apply the sphere rotation process specified in this clause to convert the local coordinate axes to the global coordinate axes.

It is assumed that the global coordinate systems for different media types were made aligned during content production.

The outputs are derived as follows:

$$\begin{aligned}
 \phi &= \phi_d * \pi \div 180 \\
 \theta &= \theta_d * \pi \div 180 \\
 \alpha &= \alpha_d * \pi \div 180 \\
 \beta &= \beta_d * \pi \div 180 \\
 \gamma &= \gamma_d * \pi \div 180 \\
 x_1 &= \text{Cos}(\phi) * \text{Cos}(\theta) \\
 y_1 &= \text{Sin}(\phi) * \text{Cos}(\theta) \\
 z_1 &= \text{Sin}(\theta) \\
 x_2 &= \text{Cos}(\beta) * \text{Cos}(\alpha) * x_1 - \text{Cos}(\beta) * \text{Sin}(\alpha) * y_1 + \text{Sin}(\beta) * z_1 \\
 y_2 &= (\text{Cos}(\gamma) * \text{Sin}(\alpha) + \text{Sin}(\gamma) * \text{Sin}(\beta) * \text{Cos}(\alpha)) * x_1 + \\
 &\quad (\text{Cos}(\gamma) * \text{Cos}(\alpha) - \text{Sin}(\gamma) * \text{Sin}(\beta) * \text{Sin}(\alpha)) * y_1 - \\
 &\quad \text{Sin}(\gamma) * \text{Cos}(\beta) * z_1 \\
 z_2 &= (\text{Sin}(\gamma) * \text{Sin}(\alpha) - \text{Cos}(\gamma) * \text{Sin}(\beta) * \text{Cos}(\alpha)) * x_1 + \\
 &\quad (\text{Sin}(\gamma) * \text{Cos}(\alpha) + \text{Cos}(\gamma) * \text{Sin}(\beta) * \text{Sin}(\alpha)) * y_1 + \\
 &\quad \text{Cos}(\gamma) * \text{Cos}(\beta) * z_1 \\
 \phi' &= \text{Atan2}(y_2, x_2) * 180 \div \pi \\
 \theta' &= \text{Asin}(z_2) * 180 \div \pi
 \end{aligned}$$

5.4 Region-wise packing formats

5.4.1 General

Subclause 5.4 specifies the inverse processes of the region-wise packing for remapping of a luma sample location in a packed region onto a luma sample location of the corresponding projected region.

The inverse region-wise packing processes specified in this document are identified by a 4-bit unsigned integer value. Table 9 specifies the region-wise packing format identifier values and provides references to the clauses in which the respective inverse processes are specified.

Table 9 – Region-wise packing formats

Identifier value	Region-wise packing format	Clause
0	Rectangular region-wise packing	5.4.2
1..15	Reserved	N/A

5.4.2 Conversion of one sample location for rectangular region-wise packing

This clause specifies the inverse of the rectangular region-wise packing process for remapping of a luma sample location in a packed region onto a luma sample location of the corresponding projected region.

Inputs to this process are:

- sample location (x, y) within the packed region, where x and y are in relative packed picture sample units, while the sample location is at an integer sample location within the packed picture,
- the width and the height (projRegWidth, projRegHeight) of the projected region, in relative projected picture sample units,

- the width and the height (packedRegWidth, packedRegHeight) of the packed region, in relative packed picture sample units,
- transform type (transformType), and
- offset values for the sampling position (offsetX, offsetY) in the range of 0, inclusive, to 1, exclusive, in horizontal and vertical relative packed picture sample units, respectively.

NOTE offsetX and offsetY both equal to 0.5 indicate a sampling position that is in the centre point of a sample in packed picture sample units.

Outputs of this process are:

- the centre point of the sample location (hPos, vPos) within the projected region, where hPos and vPos are in relative projected picture sample units and may have non-integer real values.

The outputs are derived as follows:

```

if( transformType == 0 || transformType == 1 || transformType == 2 || transformType == 3 ) {
    horRatio = projRegWidth ÷ packedRegWidth
    verRatio = projRegHeight ÷ packedRegHeight
} else if ( transformType == 4 || transformType == 5 || transformType == 6 ||
transformType == 7 ) {
    horRatio = projRegWidth ÷ packedRegHeight
    verRatio = projRegHeight ÷ packedRegWidth
}
if( transformType == 0 ) {
    hPos = horRatio * ( x + offsetX )
    vPos = verRatio * ( y + offsetY )
} else if ( transformType == 1 ) {
    hPos = horRatio * ( packedRegWidth - x - offsetX )
    vPos = verRatio * ( y + offsetY )
} else if ( transformType == 2 ) {
    hPos = horRatio * ( packedRegWidth - x - offsetX )
    vPos = verRatio * ( packedRegHeight - y - offsetY )
} else if ( transformType == 3 ) {
    hPos = horRatio * ( x + offsetX )
    vPos = verRatio * ( packedRegHeight - y - offsetY )
} else if ( transformType == 4 ) {
    hPos = horRatio * ( y + offsetY )
    vPos = verRatio * ( x + offsetX )
} else if ( transformType == 5 ) {
    hPos = horRatio * ( y + offsetY )
    vPos = verRatio * ( packedRegWidth - x - offsetX )
} else if ( transformType == 6 ) {
    hPos = horRatio * ( packedRegHeight - y - offsetY )
    vPos = verRatio * ( packedRegWidth - x - offsetX )
} else if ( transformType == 7 ) {
    hPos = horRatio * ( packedRegHeight - y - offsetY )
    vPos = verRatio * ( x + offsetX )
}

```

6 Fisheye omnidirectional video

6.1 General

Without the *projection* and *region-wise packing* processes specified in Clause 5, multiple circular images captured by fisheye cameras may be directly projected onto a picture, which consists of fisheye omnidirectional video. In an OMAF player, the decoded fisheye omnidirectional video may be stitched and rendered according to the user's intended viewport using the signalled fisheye video parameters, including:

- region information of circular images in the coded picture,
- field of view and camera parameters of fisheye lens,
- lens distortion correction (LDC) parameters with local variation of FOV, and
- lens shading compensation (LSC) parameters with RGB gains.

Essential fisheye video parameters for enabling stitching and rendering at the OMAF player are specified in `FisheyeVideoEssentialInfoStruct` in subclause 6.2. For high quality rendering and efficient delivery of fisheye video, supplemental fisheye video parameters are specified in `FisheyeVideoSupplementalInfoStruct` in subclause 6.3.

6.2 FisheyeVideoEssentialInfoStruct syntax structure

6.2.1 Syntax

```
aligned(8) class FisheyeVideoEssentialInfoStruct() {
    unsigned int(3) view_dimension_idc;
    bit(21) reserved = 0;
    unsigned int(8) num_circular_images;
    for (i=0; i< num_circular_images; i++) {
        unsigned int(32) image_centre_x;
        unsigned int(32) image_centre_y;
        unsigned int(32) rect_region_top;
        unsigned int(32) rect_region_left;
        unsigned int(32) rect_region_width;
        unsigned int(32) rect_region_height;
        unsigned int(32) full_radius;
        unsigned int(32) scene_radius;
        signed int(32) camera_centre_azimuth;
        signed int(32) camera_centre_elevation;
        signed int(32) camera_centre_tilt;
        unsigned int(32) camera_centre_offset_x;
        unsigned int(32) camera_centre_offset_y;
        unsigned int(32) camera_centre_offset_z;
        unsigned int(32) field_of_view;
        bit(16) reserved = 0;
        unsigned int(16) num_polynomial_coefs_distortion;
        for (j=0; j< num_polynomial_coefs_distortion; j++)
            signed int(32) polynomial_coef_k_distortion;
    }
}
```

6.2.2 Semantics

`view_dimension_idc` indicates that the syntax element values of this `FisheyeVideoEssentialInfoStruct` syntax structure have been constrained as specified in Table 10. The indicated constraints shall be obeyed in the syntax element values of this `FisheyeVideoEssentialInfoStruct` syntax structure.

Table 10 – Constraints implied by view_dimension_idc

view_dimension_idc	Constraints
0	num_circular_images is equal to 2. The values of camera_centre_azimuth, camera_centre_elevation, camera_centre_tilt, camera_centre_offset_x, camera_centre_offset_y, and camera_centre_offset_z are such that the circular images have aligned optical axes and face opposite directions. The sum of field_of_view values is greater than or equal to $360 * 2^{16}$.
1	num_circular_images is equal to 2. The values of camera_centre_azimuth, camera_centre_elevation, camera_centre_tilt, camera_centre_offset_x, camera_centre_offset_y, and camera_centre_offset_z are such that the circular images have parallel optical axes that are orthogonal to the line intersecting the camera centre points. The camera with i equal to 0 is the left view.
2	num_circular_images is equal to 2. The values of camera_centre_azimuth, camera_centre_elevation, camera_centre_tilt, camera_centre_offset_x, camera_centre_offset_y, and camera_centre_offset_z are such that the circular images have parallel optical axes that are orthogonal to the line intersecting the camera centre points. The camera with i equal to 0 is the right view.
3 to 6, inclusive	Reserved.
7	No additional constraints are implied for the syntax element values within this FisheyeVideoEssentialInfoStruct syntax structure.

num_circular_images specifies the number of circular images in each coded picture this structure applies to. Typically, the value is equal to 2, but other non-zero values are also possible.

image_centre_x is a fixed-point 16.16 value that specifies the horizontal coordinate, in luma samples, of the centre of the circular image in each coded picture this structure applies to, relative to the top-left corner of the coded picture.

image_centre_y is a fixed-point 16.16 value that specifies the vertical coordinate, in luma samples, of the centre of the circular image in each coded picture this structure applies to, relative to the top-left corner of the coded picture.

rect_region_top, rect_region_left, rect_region_width, and rect_region_height specify the coordinates of the top-left corner and the width and height of the rectangular region that contains the circular image, which may or may not be cropped. These values are specified in units of luma samples.

`full_radius` is a fixed-point 16.16 value that specifies the radius, in luma samples, from the centre of the circular image to the edge of the circular image that corresponds to the maximum field of view of the fisheye lens, specified by `field_of_view`.

The active pixel area of a circular image is defined by the intersection of the rectangular region, defined by `rect_region_top`, `rect_region_left`, `rect_region_width`, and `rect_region_height`, and the circle defined by `image_centre_x`, `image_centre_y`, and `full_radius`.

`scene_radius` is a fixed-point 16.16 value that specifies the radius, in luma samples, from the centre of the circular image to the closest edge of the area in the image where there are no obstructions from the camera body itself, and that the enclosed area is the suggested area for stitching as recommended by the content provider.

`camera_centre_azimuth` and `camera_centre_elevation` specify the centre of the sphere region corresponding to the centre of the circular image. `camera_centre_azimuth` shall be in the range of $-180 * 2^{16}$ to $180 * 2^{16} - 1$, inclusive. `camera_centre_elevation` shall be in the range of $-90 * 2^{16}$ to $90 * 2^{16}$, inclusive.

`camera_centre_tilt` specifies the tilt angle of the centre of the sphere region corresponding to the centre of the circular image. `camera_centre_tilt` shall be in the range of $-180 * 2^{16}$ to $180 * 2^{16} - 1$, inclusive.

`camera_centre_offset_x`, `camera_centre_offset_y` and `camera_centre_offset_z` are fixed-point 16.16 values that indicate the XYZ offset values, in millimeters, of the focal centre of the fisheye camera lens corresponding to the circular image, from the focal centre origin of the overall fisheye camera configuration as illustrated in Figure 10.

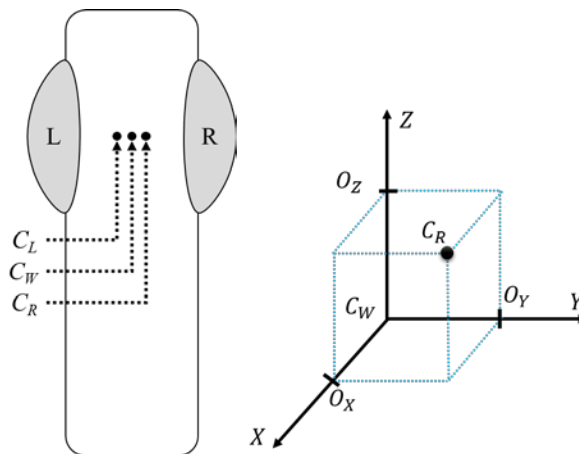


Figure 10 – Illustration of `camera_centre_offset_x` (O_X), `camera_centre_offset_y` (O_Y), and `camera_centre_offset_z` (O_Z) of the focal centre (C_L or C_R) of the fisheye camera lens (L or R, respectively) relative to the focal centre origin of the overall fisheye camera configuration (C_W)

`field_of_view` is a fixed-point 16.16 value that specifies the field of view of the fisheye lens corresponding to the circular image, in degrees. A typical value for a hemispherical fisheye lens is 180.0 degrees.

`num_polynomial_coefs_distortion` and `polynomial_coef_k_distortion` are lens distortion parameters that define the curve function between the normalized radius (r) of luma sample x_p in the circular image and the angle (θ) in the camera sphere where X_p and X_c are the corresponding points of x_p and x_c as shown in Figure 11. The normalized radius (r) of x_p is calculated as the distance (d) divided by `full_radius` where the distance (d) is the distance between x_p and the centre of the circular image, x_c in units of luma samples.

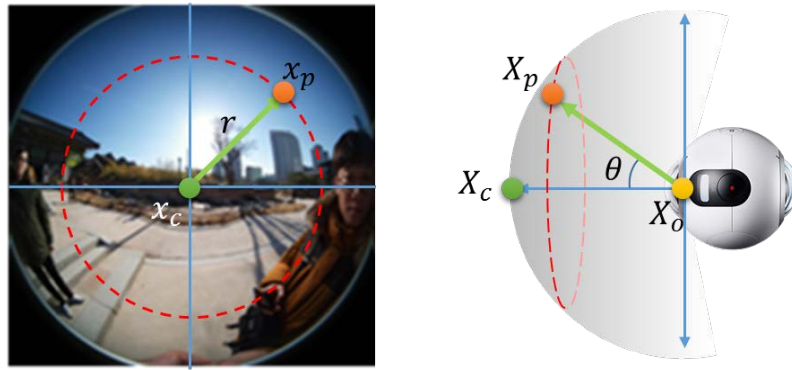


Figure 11 – Illustration of relation between radius (r) and theta (θ) of distortion function

`num_polynomial_coefs_distortion` is an integer that specifies the number of polynomial coefficients corresponding to the circular image. This is the maximum order of the polynomial plus 1.

The instances of polynomial `polynomial_coef_k_distortion` are fixed-point 8.24 polynomial coefficient values that describes the curve function from the normalized radius (r) of luma sample x_p in the circular image to the angle (θ), in radians, using the following polynomial equation:

$$\theta = \sum_{j=0}^{N_i-1} p_j * r^j$$

where p_j and N_i are represented by `polynomial_coef_k_distortion` and `num_polynomial_coefs_distortion`, respectively.

6.3 FisheyeVideoSupplementalInfoStruct syntax structure

6.3.1 Syntax

```
aligned(8) class FisheyeVideoSupplementalInfoStruct() {
    unsigned int(8) num_circular_images;
    bit(20) reserved = 0;
    unsigned int(1) flip_info_flag;
    unsigned int(1) camera_intrinsic_flag;
    unsigned int(1) local_fov_flag;
    unsigned int(1) deadzone_flag;

    for (i=0; i<num_circular_images; i++) {
        if (flip_info_flag == 1) {
            bit(30) reserved = 0;
            unsigned int(2) image_flip;
        }
        if (camera_intrinsic_flag == 1) {
            unsigned int(32) image_scale_axis_angle;
            unsigned int(32) image_scale_x;
            unsigned int(32) image_scale_y;
            bit(16) reserved = 0;
            unsigned int(16) num_polynomial_coefs_lsc;
            for (j=0; j<num_polynomial_coefs_lsc; j++) {
                signed int (32) polynomial_coef_k_lsc_r;
                signed int (32) polynomial_coef_k_lsc_g;
                signed int (32) polynomial_coef_k_lsc_b;
            }
        }
        if (local_fov_flag == 1) {
            unsigned int (16) num_angle_for_displaying_fov;
            bit(16) reserved = 0;
        }
    }
}
```

```

    for (j=0; j<num_angle_for_displaying_fov; j++) {
        unsigned int(32) displayed_fov;
        unsigned int(32) overlapped_fov;
    }
    bit(16) reserved = 0;
    unsigned int (16) num_local_fov_region;
    for (j=0; j<num_local_fov_region; j++) {
        unsigned int(32) start_radius;
        unsigned int(32) end_radius;
        signed int(32) start_angle;
        signed int(32) end_angle;
        unsigned int(32) radius_delta;
        signed int(32) angle_delta;
        for (rad=start_radius; rad<=end_radius; rad+=radius_delta)
            for (ang=start_angle; ang<= end_angle; ang+=angle_delta)
                unsigned int(32) local_fov_weight;
    }
}
}
if (deadzone_flag == 1) {
    bit(24) reserved = 0;
    unsigned int(8) num_deadzones;
    for (j=0; j<num_deadzones; j++) {
        unsigned int(16) deadzone_left_horizontal_offset;
        unsigned int(16) deadzone_top_vertical_offset;
        unsigned int(16) deadzone_width;
        unsigned int(16) deadzone_height;
    }
}
}

```

6.3.2 Semantics

`num_circular_images` specifies the number of circular images in each coded picture this structure applies to. Typically, the value is equal to 2, but other non-zero values are also possible.

`flip_info_flag` equal to 1 specifies that image flipping region information is present. `flip_info_flag` equal to 0 specifies that image flipping information is not present.

`camera_intrinsic_flag` equal to 1 specifies that image scale and lens shading compensation camera intrinsic parameters are present. `camera_intrinsic_flag` equal to 0 specifies that image scale and lens shading compensation camera intrinsic parameters are not present.

`local_fov_flag` equal to 1 specifies that the local field of view parameters are present. `local_fov_flag` equal to 0 specifies that the local field of view parameters are not present.

`deadzone_flag` equal to 1 specifies that deadzone related parameters are present. `deadzone_flag` equal to 0 specifies that deadzone parameters are not present.

`image_flip` specifies whether and how the image has been flipped and thus a reverse flipping operation needs to be applied. The value 0 indicates that the image has not been flipped. The value 1 indicates that the image has been vertically flipped. The value 2 indicates that the image has been horizontally flipped. The value 3 indicates that the image has been both vertically and horizontally flipped.

`image_scale_axis_angle`, `image_scale_x`, and `image_scale_y` are three fixed-point 16.16 values that specify whether and how the circular image has been scaled along an axis. They are used to take into account the natural error in the camera-mirror setting. The axis is defined by a single angle as indicated by the value of `image_scale_axis_angle`, in degrees. An angle of 0 degrees means a horizontal vector is perfectly horizontal and a vertical vector is perfectly vertical. The values of `image_scale_x` and `image_scale_y` indicate the scaling. They may also be called as affine parameters that satisfy the following equation:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} c & d \\ e & 1 \end{bmatrix} \begin{bmatrix} u_N \\ v_N \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix}$$

The equation relates the actual pixel coordinates (u, v) to the ideal image coordinates (u_N, v_N) . c_x and c_y represent `image_centre_x`, `image_centre_y` respectively. c , d , and e represent `image_scale_x`, `image_scale_axis_angle`, and `image_scale_y` respectively.

`num_polynomial_coefs_lsc` specifies the number of polynomial coefficients of the lens shading compensation (LSC) parameters for the circular image.

`polynomial_coef_k_lsc_r`, `polynomial_coef_k_lsc_g`, and `polynomial_coef_k_lsc_b` are 8.24 fixed-point values that specify the LSC parameters to compensate the shading artifact of the fisheye lens that reduces the color in the radial direction. The compensating weight (w) to be multiplied to the original color is approximated as a curve function using the following polynomial equation:

$$w = \sum_{i=1}^N p_{i-1} \cdot r^{i-1}$$

where r is the normalized radius as specified in subclause 6.2.2.

The weighing factors for red, green, and blue may be calculated separately when p is represented by `polynomial_coef_k_lsc_r`, `polynomial_coef_k_lsc_g`, and `polynomial_coef_k_lsc_b`, respectively, and r is the corresponding radius from the image centre after normalization by the `full_radius`. N is represented by `num_polynomial_coefs_lsc`. Figure 12 shows an example of an input fisheye video picture and the output after applying lens shading compensation.



Figure 12 – Example of a pair of output images (right) resulting from lens shading compensation from a pair of circular input images (left)

`num_angle_for_displaying_fov` specifies the number of angles that define the displayed and overlapped regions. According to the value of `num_angle_for_displaying_fov`, multiple values of `displayed_fov` and `overlapped_fov` are defined with equal intervals, which start at 12 o'clock and increase clockwise.

`displayed_fov` specifies the field of view of the part of the circular image that is recommended to be used for displaying without blending that involves adjacent circular images.

`overlapped_fov` specifies the field of view of the part of the circular image that may be overlapped on a sphere by adjacent circular images and that is recommended to be used for displaying either as such or by blending with adjacent circular images.

The values of `displayed_fov` and `overlapped_fov` shall be less than or equal to the value of `field_of_view`.

NOTE The value of `field_of_view` is determined by the physical property of each fisheye lens, while the values of `displayed_fov` and `overlapped_fov` are determined by the configuration of multiple fisheye lenses. For example, when the value of `num_circular_images` is equal to 2 and two lenses are symmetrically located, the value of `displayed_fov` and `overlapped_fov` could be set as 180 and 190 respectively, by default. However, the value could be changed depending on the configuration of the lens and the characteristics of the contents. For example, if the stitching quality with the `displayed_fov` values (left camera = 170 and right camera = 190) and the `overlapped_fov` values (left camera = 185 and right camera = 190) is better than the quality with the default values (180 and 190) or if the physical configuration of cameras is asymmetric, then the unequal `displayed_fov` and `overlapped_fov` values could be taken. In addition, when it comes to multiple ($N > 2$) fisheye images, a single `displayed_fov` value cannot specify the exact area of each fisheye image. As shown in Figure 13 and Figure 14, `displayed_fov` (red-coloured) varies according to the direction. In order to manipulate multiple ($N > 2$) fisheye images, `num_angle_for_displaying_fov` is introduced. For example, if this value is equal to 12, then the fisheye image is divided into 12 sectors where each sector angle is 30 degrees.

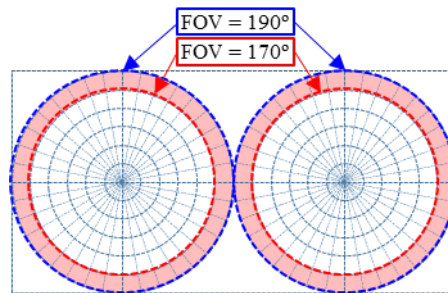
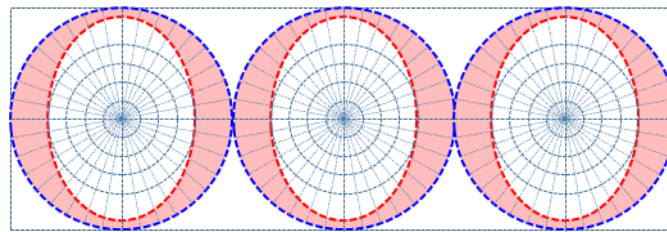


Figure 13 – Displayed FOV for two fisheye images (=170 or 190)

Figure 14 – Displayed FOV and overlapped FOV for multiple ($N > 2$) fisheye images

`num_local_fov_region` specifies the number of local fitting regions that have different fields of view.

`start_radius` (r_s), `end_radius` (r_e), `start_angle` (a_s), and `end_angle` (a_e) specify the region for local fitting/warping to change the actual field of view for displaying locally. `start_radius` and `end_radius` are fixed-point 16.16 values that specify the minimum and maximum radius values. `start_angle` and `end_angle` specify the minimum and maximum angle values that start at 12 o'clock and increase clockwise, in units of 2^{-16} degrees. `start_angle` and `end_angle` shall be in the range of $-180 * 2^{16}$ to $180 * 2^{16} - 1$, inclusive. Figure 15 illustrates an example of local FOV parameters.

`radius_delta` (r_d) is a fixed-point 16.16 value that specifies the delta radius value for representing a different field of view for each radius.

`angle_delta` (a_d) specifies the delta angle value, in units of 2^{-16} degrees, for representing a different field of view for each angle.

`local_fov_weight` (W) is a 8.24 fixed point value which specifies the weighting value for the field of view of the position specified by `start_radius`, `end_radius`, `start_angle`, `end_angle`, the angle index i and the radius index j . The positive value of `local_fov_weight` specifies the expansion of field of view, while the negative value specifies the contraction of field of view. Figure 16 shows an example result of applying local FOV parameters with a set of weighting values.

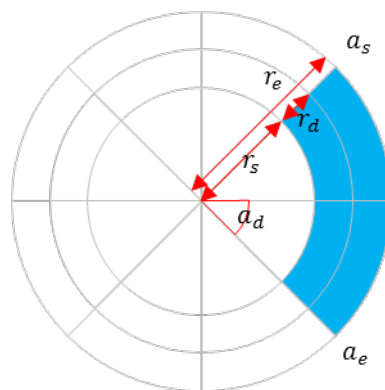


Figure 15 – Illustration of parameters regarding local FOV

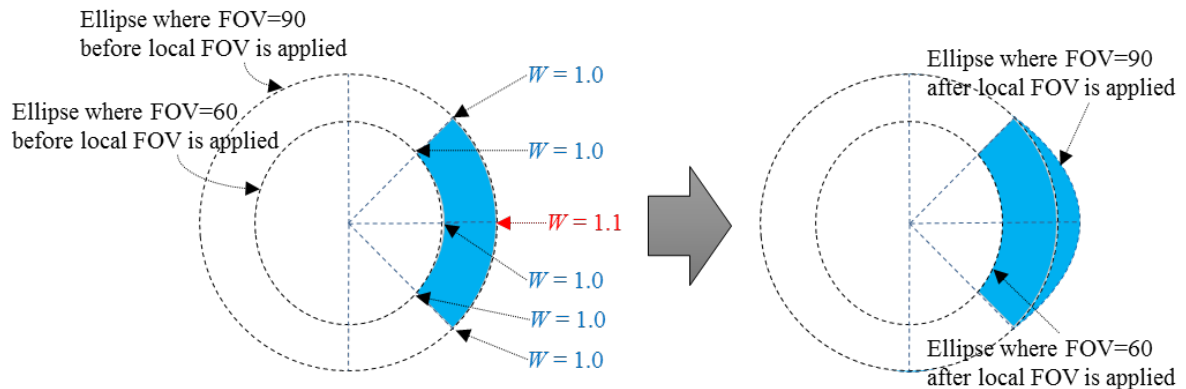


Figure 16 – Example of local FOV with a_s , a_e , and a_d equal to 45°, 135°, and 45°, respectively, and r_s , r_e , and r_d equal to 0.5, 1.0, and 0.5, respectively

`num_deadzones` is an integer that specifies the number of dead zones in each coded picture this structure applies to.

`deadzone_left_horizontal_offset`, `deadzone_top_vertical_offset`, `deadzone_width`, and `deadzone_height` are integer values that specify the position and size of the deadzone rectangular area in which the pixels are not usable. `deadzone_left_horizontal_offset` and `deadzone_top_vertical_offset` specify the horizontal and vertical coordinates, respectively, in luma samples, of the upper left corner of the deadzone in the coded picture. `deadzone_width` and `deadzone_height` specify the width and height, respectively, in luma samples, of the deadzone. To save bits for representing the video, all pixels within a deadzone should be set to the same pixel value, e.g., all black.

7 Omnidirectional media storage and metadata signalling in the ISOBMFF

7.1 Generic extensions to the ISOBMFF

7.1.1 Stereoscopic video track grouping

7.1.1.1 Definition

`TrackGroupBox` with `track_group_type` equal to 'ster' indicates that this track is either the left or right view of a stereo pair suitable for playback on a stereoscopic display.

The tracks that have the same value of `track_group_id` within `StereoVideoGroupBox` form a stereo pair, and there shall be no more than two of such tracks for the same value of `track_group_id`.

NOTE 1 Usually there are two tracks indicated to be a stereo pair with the `StereoVideoGroupBox` having the same value of `track_group_id`. However, only one track could be associated with a stereo pair in specific cases. For example, the file could be edited in a manner that one of the tracks forming a stereo pair gets removed. In another example, only one of the tracks of a stereo pair is selected for transmission, e.g. using DASH.

NOTE 2 When a track contains one or more `StereoVideoGroupBoxes` within the `TrackGroupBox`, the respective Representation in a DASH MPD could use the multiple views scheme of DASH with `@value` including `rID` (when `left_view_flag` is equal to 0) or `lID` (when `left_view_flag` is equal to 1), where `ID` could be equal to the `track_group_id` provided that all content of the MPD originates from the same file or the uniqueness of `track_group_id` values across multiple files is ensured.

7.1.1.2 Syntax

```
aligned(8) class StereoVideoGroupBox extends TrackGroupBox('ster') {
    unsigned int(1) left_view_flag;
    bit(31) reserved = 0;
}
```

7.1.1.3 Semantics

`left_view_flag` equal to 0 indicates the right view of a stereo pair, and `left_view_flag` equal to 1 indicates the left view of a stereo pair. When there are two tracks with the same value of `track_group_id`, the value of `left_view_flag` shall differ.

7.1.2 Indication of `track_group_id` uniqueness

Bit 0 of the `flags` (with bit 0 being the least significant bit) of the `TrackGroupBox` is used to indicate the uniqueness of `track_group_id`. The semantics of the flag is specified as follows:

(`flags & 1`) equal to 1 in a `TrackGroupBox` of a particular `track_group_type` indicates that `track_group_id` in that `TrackGroupBox` is not equal to any `track_ID` value and is not equal to `track_group_id` of any `TrackGroupBox` with a different `track_group_type`. When (`flags & 1`) is equal to 1 in a `TrackGroupBox` with particular values of `track_group_type` and `track_group_id`, (`flags & 1`) shall be equal to 1 in all `TrackGroupBoxes` of the same values of `track_group_type` and `track_group_id`, respectively.

7.1.3 Updated semantics of `track_IDs` of the track reference box

The semantics of `track_IDs` of the `TrackReferenceBox` is changed from

`track_IDs` is an array of integers providing the track identifiers of the referenced tracks. The value 0 shall not be present. A given value shall not be duplicated in the array.

to

`track_IDs` is an array of integers providing the track identifiers of the referenced tracks or `track_group_id` values of the referenced track groups. Each value of `track_IDs[i]`, where `i` is a valid index to the `track_IDs[]` array, is an integer that provides a reference from the containing track to the track with `track_ID` equal to `track_IDs[i]` or to the track group with both `track_group_id` equal to `track_IDs[i]` and (`flags & 1`) of `TrackGroupBox` equal to 1. When a `track_group_id` value is referenced, the track reference applies to each track of the referenced track group individually unless stated otherwise in the semantics of particular track reference types. The value 0 shall not be present. A given value shall not be duplicated in the array.

7.1.4 Indication of a track not intended to be presented alone

Bit 4 of the `flags` (with bit 0 being the least significant bit) of the `TrackHeaderBox` is used to indicate whether a track is not intended to be presented alone, e.g., due to that the track represents only a small portion of a videos scene. The semantics of the flag is specified as follows:

`track_not_intended_for_presentation_alone`: Indicates that the track is not intended to be presented alone without other tracks. Flag value is 0x000010. The flag not being set (i.e., `flags & 0x000010` is equal to 0) indicates that the track may or may not be intended to be presented alone without other tracks.

7.1.5 Timed metadata tracks

7.1.5.1 Association with media tracks

When a timed metadata track is linked to one or more media tracks with a '`cdsc`' track reference, it describes each media track individually.

7.1.6 Compatible scheme type box

7.1.6.1 Definition

Box Type: 'csch'
 Container: RestrictedSchemeInfoBox
 Mandatory: No
 Quantity: Zero or more

CompatibleSchemeTypeBox identifies a scheme type that the track conforms to.

```
aligned(8) class CompatibleSchemeTypeBox extends FullBox('csch', 0, flags) {
    // identical syntax to SchemeTypeBox
    unsigned int(32) scheme_type;           // 4CC identifying the scheme
    unsigned int(32) scheme_version;        // scheme version
    if (flags & 0x000001) {
        unsigned int(8) scheme_uri[];      // browser uri
    }
}
```

7.1.6.2 Semantics

The semantics of the syntax elements are identical to the semantics of the syntax elements with the same name in SchemeTypeBox.

7.1.7 Multiple transformations for a single transformed media track

A transformed media track may have undergone several transformations of different types and shall not have undergone more than one transformation of any particular type.

EXAMPLE A transformed track could be both protected and restricted.

The following process applies to conclude the untransformed sample entry type of a transformed media track:

1. Let `schemeInfoContainerBox` be `ProtectionSchemeInfoBox`, `RestrictedSchemeInfoBox`, or `CompleteTrackInfoBox` when the track sample entry type indicates an encrypted, restricted, or incomplete media track, respectively.
2. Let `dataFormat` be equal to the `data_format` value of `OriginalFormatBox` of `schemeInfoContainerBox`.
3. If `dataFormat` indicates a transformed media track, `schemeInfoContainerBox` is updated to be `ProtectionSchemeInfoBox`, `RestrictedSchemeInfoBox`, or `CompleteTrackInfoBox` when `dataFormat` indicates an encrypted, restricted, or incomplete media track, respectively. The process continues from step 2.
4. Otherwise (`dataFormat` does not indicate a transformed media track), the untransformed sample entry type is concluded to be equal to `dataFormat`.

7.1.8 The 'codecs' parameter for a transformed media track

For a transformed media track, the 'codecs' parameter is a comma-separated list of one or more list items. Each list item is formed from a sample entry of the transformed media track by concatenating the track sample entry type (e.g., 'resv'), a scheme type string, zero or more pairs of sample entry types and scheme type strings of transformed media, and an original format string (e.g., 'avc1'), each separated by '.

The following process applies to derive the value for a list item of the `codecs` parameter of a transformed media track:

1. The value of the `codecs` parameter is initialized to be an empty string.
2. Let `schemeInfoContainerBox` be `ProtectionSchemeInfoBox`, `RestrictedSchemeInfoBox`, or `CompleteTrackInfoBox` when the track sample entry type indicates an encrypted, restricted, or incomplete media track, respectively.

3. Let `dataFormat` be equal to the `data_format` value of `OriginalFormatBox` of `schemeInfoContainerBox`. The value of the `codecs` parameter is appended by `dataFormat` followed by a dot ('.') character.
4. When `dataFormat` indicates a transformed media track, the value of the `codecs` parameter is appended by the `scheme_type` four-character code contained in the `SchemeTypeBox` of `schemeInfoContainerBox`.
5. When `schemeInfoContainerBox` contains instances of `CompatibleSchemeTypeBox`, the following applies for each instance of the `CompatibleSchemeTypeBox` of `schemeInfoContainerBox` in any order: the value of the `codecs` parameter is appended by a plus character ('+') followed by the `scheme_type` four-character code contained in the `CompatibleSchemeTypeBox`.
6. The value of the `codecs` parameter is appended by a dot ('.') character.
7. If `dataFormat` indicates a transformed media track, `schemeInfoContainerBox` is updated to be `ProtectionSchemeInfoBox`, `RestrictedSchemeInfoBox`, or `CompleteTrackInfoBox` when `dataFormat` indicates an encrypted, restricted, or incomplete media track, respectively. The process continues from step 3.
8. Otherwise (`dataFormat` does not indicate a transformed media track), the value of the `codecs` parameter is appended by the original format string.

The original format string starts with the untransformed sample entry type indicating a coding format and follows the specifications of the 'codecs' string for that coding format.

In the process above, steps 4 and 5 describe the derivation of a scheme type string.

For example, the value of the 'codecs' parameter might be 'resv.podv+erpv.avc1'.

7.1.9 Track type box

7.1.9.1 Definition

Box Type:	'ttyp'
Container:	TrackBox
Mandatory:	No
Quantity:	Zero or one

The payload of `TrackTypeBox` has the same syntax as the payload of `FileTypeBox`. The content of an instance of `TrackTypeBox` shall be such that it would apply as the content of `FileTypeBox`, if all other tracks of the file were removed and only the track containing this box remained in the file.

NOTE `TrackTypeBox` could be used in specifying media profiles or track-specific brands.

7.1.9.2 Syntax

```
aligned(8) class TrackTypeBox extends FullBox('ttyp', 0, 0) {
    unsigned int(32)  major_brand;
    unsigned int(32)  minor_version;
    unsigned int(32)  compatible_brands[];    // to end of the box
}
```

NOTE The payload syntax of `TrackTypeBox` is identical to that of `FileTypeBox`.

7.1.9.3 Semantics

The semantics of the syntax elements are identical to the semantics of the syntax elements with the same name in `FileTypeBox`, if all other tracks of the file were removed and only the track containing this box remained in the file.

7.1.10 Clarifications on the stereo video box

This document uses `StereoVideoBox` as specified in ISO/IEC 14496-12 with the following additional specifications:

- The definition of `StereoVideoBox` is changed from

Box Type:	'stvi'
Container:	SchemeInformationBox
Mandatory:	Yes (when the SchemeType is 'stvi')
Quantity:	One

to

Box Type:	'stvi'
Container:	SchemeInformationBox
Mandatory:	Yes (when the SchemeType is 'stvi'), no (otherwise)
Quantity:	Zero or one

- When `stereo_scheme` is equal to 4, **PackedContentInterpretationType** specified in ISO/IEC 23091-2 is inferred to be equal to 1.
- When the value of `stereo_indication_type` indicates the temporal interleaving frame packing arrangement and the display system in use presents two views simultaneously, readers should implicitly set the composition time for constituent picture 0 to coincide with the composition time for constituent picture 1.
- When the value of `stereo_indication_type` indicates the temporal interleaving frame packing arrangement, each sync sample and each SAP sample with `SAP_type` in the range of 1 to 3, inclusive, indicated by the stream access point sample group represent constituent picture 0 followed, in composition time order, by samples representing constituent pictures 1 and 0 in an alternating manner up to but excluding the next sync sample or SAP sample with `SAP_type` in the range of 1 to 3, inclusive, indicated by the stream access point sample group.

7.2 Generic extensions to ISO/IEC 14496-15

7.2.1 Alternative extraction source track grouping

Members of the track group with `track_group_type` equal to 'alte' are alternatives to be used as a source for 'scal' or 'sabt' track reference. The value of `(flags & 1)` shall be equal to 1 in a `TrackGroupTypeBox` of type 'alte' to indicate the uniqueness of `track_group_id` as specified in subclause 7.1.2.

A 'scal' or 'sabt' track reference may refer to a `track_group_id` value of an 'alte' track group. As implied by the general semantics of a track reference to a `track_group_id` as specified in subclause 7.1.3, any single track of an 'alte' track group is a valid source for extraction as specified in ISO/IEC 14496-15:2017, Clauses A.3 and A.7 or for bitstream reconstruction from tile tracks as specified in ISO/IEC 14496-15:2017, subclause 10.5.4.

7.2.2 Tile base track association with coverage information box and timed metadata data track

When a timed metadata track is linked to a tile base track with a 'cdsc' track reference, it describes the HEVC video bitstream carried by the tile base track and all the associated tile tracks.

7.3 OMAF-specific extensions to the ISOBMFF

7.3.1 Sync samples in timed metadata tracks

For timed metadata tracks specified in this document, a sample in a timed metadata track is defined as a sync sample if and only if at least one of media samples in the referenced media tracks having the same decoding time is a sync sample.

7.4 OMAF-specific extensions to ISO/IEC 14496-15

7.4.1 Coverage information box in a tile base track

When `CoverageInformationBox` is included in `ProjectedOmniVideoBox` of a tile base track, it provides information on the area on the sphere covered by the content that is represented by all the tile tracks associated with the tile base track.

7.5 Structures and semantics that are common for video tracks and image items

7.5.1 Semantics of sample locations within a decoded picture

7.5.1.1 Relation of decoded pictures to global coordinate axes (informative)

Figure 17 illustrates the conversions from a spherical picture to a packed picture that could be used in content authoring and the corresponding conversions from a packed picture to a spherical picture to be rendered that could be used in an OMAF player. The example in this clause is described for a packed picture that appears in a projected omnidirectional video track. Similar description could be derived for an image item.

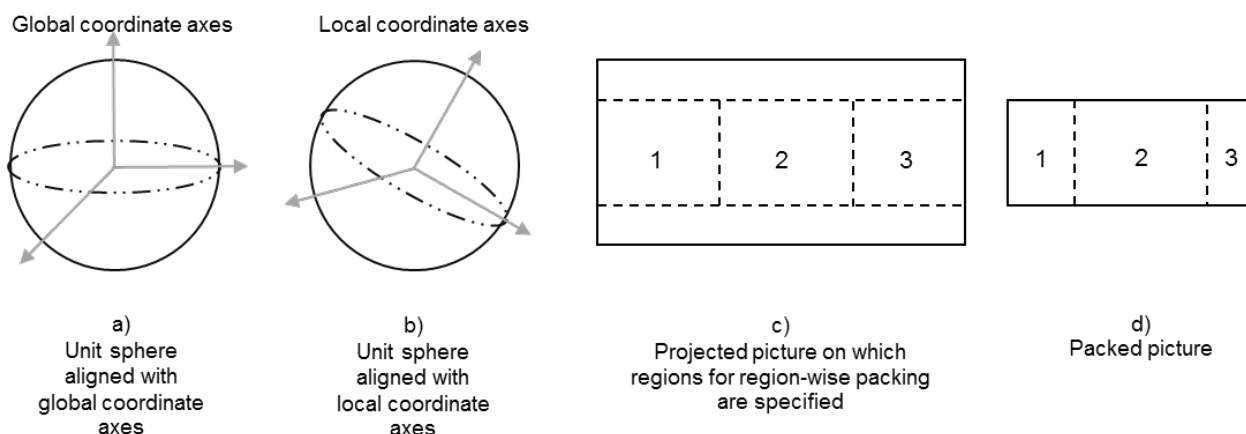


Figure 17 – Example of processing stages to derive a packed picture from a spherical image or vice versa

The content authoring could include the following ordered steps:

- The source images provided as input are stitched to generate a sphere picture on the unit sphere per the global coordinate axes as indicated in Figure 17a.
- The unit sphere is then rotated relative to the global coordinate axes, as indicated in Figure 17b. The amount of rotation to convert from the local coordinate axes to the global coordinate axes is specified by the rotation angles indicated in the `RotationBox`. The local coordinate axes of the unit sphere are the axes of the coordinate system that has been rotated. The absence of `RotationBox` indicates that the local coordinate axes are the same as the global coordinate axes.
- As illustrated in Figure 17c, the spherical picture on the rotated unit sphere is then converted to a two-dimensional projected picture, for example using the equirectangular projection specified in subclause 5.2.1. When spatial packing of stereoscopic content is applied, two spherical pictures for the two views are converted to two constituent pictures, after which frame packing is applied to pack the two constituent pictures to one projected picture.
- Rectangular region-wise packing could be applied to obtain a packed picture from the projected picture. One example of packing is depicted in Figure 17c and Figure 17d. The dashed rectangles in Figure 17c indicate the projected regions on a projected picture, and the respective areas in Figure 17d indicate the corresponding packed regions. In this example, projected regions 1 and 3 are horizontally downsampled, while projected region 2 is kept at its original resolution.

`CoverageInformationBox` could be used to indicate which part of the sphere is covered by the packed picture.

In order to map sample locations of a packed picture (such as that in Figure 17d) to a unit sphere used in rendering (Figure 17a), the OMAF player could perform the following ordered steps:

- A packed picture, such as that in Figure 17d, is obtained as a result of decoding a picture from a video track or an image item.
- If needed, chroma sample arrays of the packed picture are upsampled to the resolution of the luma sample array of the packed picture, and colour space conversion could also be performed.

- If region-wise packing is indicated, the sample locations of the packed picture are converted to sample locations of the respective projected picture, such as that in Figure 17c, as specified in subclause 5.4.2. Otherwise, the projected picture is identical to the packed picture.
- If spatial frame packing of the projected picture is indicated, the sample locations of the projected picture are converted to sample locations of the respective constituent picture of the projected picture, as specified in subclause 7.5.1.3. Otherwise, the constituent picture of the projected picture is identical to the projected picture.
- The sample locations of a constituent picture the projected picture are converted to sphere coordinates that are relative to local coordinate axes, as specified for the omnidirectional projection format being used in subclause 5.2. The resulting sample locations correspond to a sphere picture depicted in Figure 17b.
- If rotation is indicated, the sphere coordinates relative to the local coordinate axes are converted to sphere coordinates relative to the global coordinate axes as specified in subclause 5.3. Otherwise, the global coordinate axes are identical to the local coordinate axes.

The overall process for mapping of luma sample locations within a decoded picture to sphere coordinates relative to the global coordinate axes is normatively specified in subclause 7.5.1.2.

7.5.1.2 Mapping of luma sample locations within a decoded picture to sphere coordinates relative to the global coordinate axes

This clause specifies the semantics of luma sample locations within a decoded picture to sphere coordinates relative to the global coordinate axes. The decoded picture may be of any of the following:

- For video, the decoded picture is the decoding output resulting from a sample of the video track.
- For an image item, the decoded picture is a reconstructed image of the image item.

offsetX is set equal to 0.5 and offsetY is set equal to 0.5.

If RegionWisePackingFlag is equal to 1, the following applies for each packed region n in the range of 0 to NumRegions – 1, inclusive:

- For each sample location (xPackedPicture, yPackedPicture) belonging to the n -th packed region with PackingType[n] equal to 0 (i.e., with rectangular region-wise packing), the following applies:
 - The corresponding sample location (xProjPicture, yProjPicture) of the projected picture is derived as follows:
 - x is set equal to $xPackedPicture - PackedRegLeft[n]$.
 - y is set equal to $yPackedPicture - PackedRegTop[n]$.
 - Subclause 5.4.2 is invoked with x , y , PackedRegWidth[n], PackedRegHeight[n], ProjRegWidth[n], ProjRegHeight[n], TransformType[n], offsetX, and offsetY as inputs, and the output is assigned to sample location (hPos, vPos).
 - xProjPicture is set equal to $ProjRegLeft[n] + hPos$.
 - When SideBySideFlag is equal to 0, and when xProjPicture is greater than or equal to proj_picture_width, xProjPicture is set equal to $xProjPicture - proj_picture_width$.
 - When SideBySideFlag is equal to 1, the following applies:
 - When ProjRegLeft[n] is less than $proj_picture_width / 2$ and xProjPicture is greater than or equal to $proj_picture_width / 2$, xProjPicture is set equal to $xProjPicture - proj_picture_width / 2$.
 - When ProjRegLeft[n] is greater than or equal to $proj_picture_width / 2$ and xProjPicture is greater than or equal to $proj_picture_width$, xProjPicture is set equal to $xProjPicture - proj_picture_width / 2$.
 - yProjPicture is set equal to $ProjRegTop[n] + vPos$.
 - Subclause 7.5.1.3 is invoked with xProjPicture, yProjPicture, ConstituentPicWidth, and ConstituentPicHeight as inputs, and the outputs indicating the sphere coordinates and the constituent frame index (for frame-packed

stereoscopic video) for the luma sample location (xPackedPicture, yPackedPicture) belonging to the n-th packed region in the decoded picture.

Otherwise (RegionWisePackingFlag is equal to 0), the following applies for each sample location (x, y) within the decoded picture:

- xProjPicture is set equal to x + offsetX.
- yProjPicture is set equal to y + offsetY.
- Subclause 7.5.1.3 is invoked with xProjPicture, yProjPicture, ConstituentPicWidth, and ConstituentPicHeight as inputs, and the outputs indicating the sphere coordinates and the constituent frame index (for frame-packed stereoscopic video) for the sample location (x, y) within the decoded picture.

7.5.1.3 Conversion from a sample location in a projected picture to sphere coordinates relative to the global coordinate axes

Inputs to this process are

- the centre point of a sample location (xProjPicture, yProjPicture) within a projected picture, where xProjPicture and yProjPicture are in relative projected picture sample units and may have non-integer real values, and
- pictureWidth and pictureHeight, which are the width and height, respectively, of a monoscopic projected luma picture, in relative projected picture sample units.

NOTE 1 The projected picture for which the sample location (xProjPicture, yProjPicture) is given as input could be a spatially frame-packed picture.

Outputs of this process are:

- sphere coordinates (azimuthGlobal, elevationGlobal), in units of degrees relative to the global coordinate axes, and
- when SpatiallyPackedStereoFlag is equal to 1, the index of the constituent picture (constituentPicture) equal to 0 or 1.

NOTE 2 When the temporal interleaving packing arrangement is in use, the projected picture is associated with the left view or right view as specified in subclause 7.1.10.

The outputs are derived with the following ordered steps:

- If xProjPicture is greater than or equal to pictureWidth or yProjPicture is greater than or equal to pictureHeight, the following applies:
 - constituentPicture is set equal to 1.
 - If xProjPicture is greater than or equal to pictureWidth, xProjPicture is set to xProjPicture – pictureWidth.
 - If yProjPicture is greater than or equal to pictureHeight, yProjPicture is set to yProjPicture – pictureHeight.
- Otherwise, constituentPicture is set equal to 0.
- Depending on the projection format, the following applies:
 - When the projection format is the equirectangular projection, subclause 5.2.2 is invoked with pictureWidth, pictureHeight, xProjPicture, and yProjPicture as inputs, and the output is assigned to azimuthLocal, elevationLocal.
 - When the projection format is the cubemap projection, subclause 5.2.3 is invoked with pictureWidth, pictureHeight, xProjPicture, and yProjPicture as inputs, and the output is assigned to azimuthLocal, elevationLocal.
- If RotationFlag is equal to 1, subclause 5.3 is invoked with azimuthLocal, elevationLocal, $\text{rotation_yaw} \div 2^{16}$, $\text{rotation_pitch} \div 2^{16}$, and $\text{rotation_roll} \div 2^{16}$ as inputs, and the output is assigned to azimuthGlobal and elevationGlobal.
- Otherwise, azimuthGlobal is set equal to azimuthLocal and elevationGlobal is set equal to elevationLocal.

7.5.2 Projection format structure

7.5.2.1 Syntax

```
aligned(8) class ProjectionFormatStruct() {
    bit(3) reserved = 0;
    unsigned int(5) projection_type;
}
```

7.5.2.2 Semantics

`projection_type` indicates the type of the mapping of the projected picture onto the spherical coordinate system as specified in subclause 5.1. The values of `projection_type` and their semantics are specified in Table 8.

7.5.3 Region-wise packing structure

7.5.3.1 Definition

`RegionWisePackingStruct` specifies the mapping between packed regions and the respective projected regions and specifies the location and size of the guard bands, if any.

NOTE Among other information the `RegionWisePackingStruct` also provides the content coverage information in the 2D Cartesian picture domain.

A decoded picture in the semantics of this clause is either one of the following depending on the container for this syntax structure:

- For video, the decoded picture is the decoding output resulting from a sample of the video track.
- For an image item, the decoded picture is a reconstructed image of the image item.

The content of `RegionWisePackingStruct` is informatively summarized below, while the normative semantics follow subsequently in this clause:

- The width and height of the projected picture are explicitly signalled with `proj_picture_width` and `proj_picture_height`, respectively.
- The width and height of the packed picture are explicitly signalled with `packed_picture_width` and `packed_picture_height`, respectively.
- When the projected picture is stereoscopic and has the top-bottom or side-by-side frame packing arrangement, `constituent_picture_matching_flag` equal to 1 specifies that
 - the projected region information, packed region information, and guard band region information in this syntax structure apply individually to each constituent picture,
 - the packed picture and the projected picture have the same stereoscopic frame packing format, and
 - the number of projected regions and packed regions is double of that indicated by the value of `num_regions` in the syntax structure.
- `RegionWisePackingStruct` contains a loop, in which a loop entry corresponds to the respective projected regions and packed regions in both constituent pictures (when `constituent_picture_matching_flag` equal to 1) or to a projected region and the respective packed region (when `constituent_picture_matching_flag` equal to 0), and the loop entry contains the following:
 - a flag indicating the presence of guard bands for the packed region,
 - the packing type (however, only rectangular region-wise packing is specified in this document),
 - the mapping between a projected region and the respective packed region in the rectangular region packing structure `RectRegionPacking(i)`,
 - when guard bands are present, the guard band structure for the packed region `GuardBand(i)`.

The content of the rectangular region packing structure `RectRegionPacking(i)` is informatively summarized below, while the normative semantics follow subsequently in this clause:

- `proj_reg_width[i]`, `proj_reg_height[i]`, `proj_reg_top[i]`, and `proj_reg_left[i]` specify the width, height, top offset, and left offset, respectively, of the *i*-th projected region.
- `transform_type[i]` specifies the rotation and mirroring, if any, that are applied to the *i*-th packed region to remap it to the *i*-th projected region.
- `packed_reg_width[i]`, `packed_reg_height[i]`, `packed_reg_top[i]`, and `packed_reg_left[i]` specify the width, height, the top offset, and the left offset, respectively, of the *i*-th packed region.

The content of the guard band structure `GuardBand(i)` is informatively summarized below, while the normative semantics follow subsequently in subclause 7.5.3:

- `left_gb_width[i]`, `right_gb_width[i]`, `top_gb_height[i]`, or `bottom_gb_height[i]` specify the guard band size on the left side of, the right side of, above, or below, respectively, the *i*-th packed region.
- `gb_not_used_for_pred_flag[i]` indicates if the encoding was constrained in a manner that guards bands are not used as a reference in the inter prediction process.
- `gb_type[i][j]` specifies the type of the guard bands for the *i*-th packed region.

Figure 18 illustrates an example of the position and size of a projected region within a projected picture (on the left side) as well as that of a packed region within a packed picture with guard bands (on the right side). This example applies when the value of `constituent_picture_matching_flag` is equal to 0.

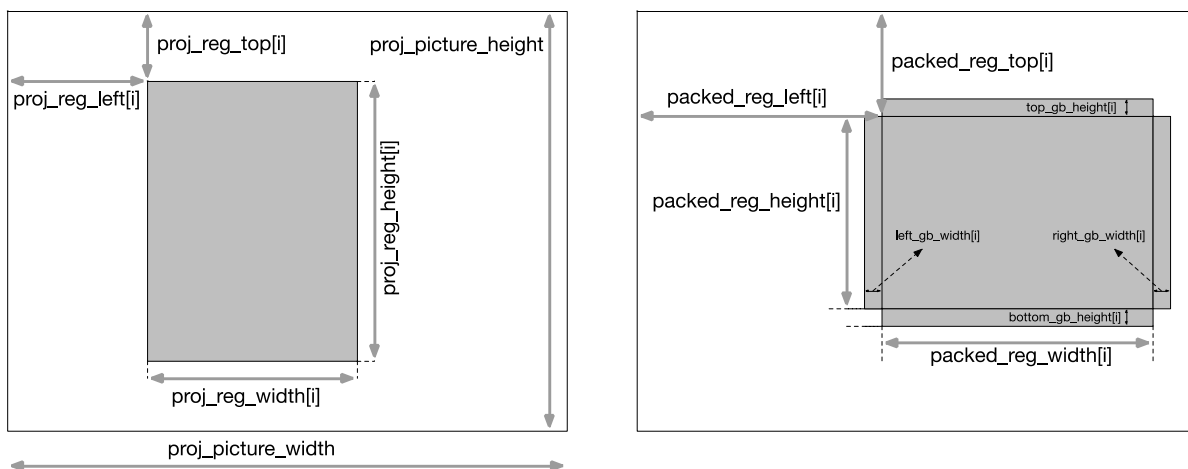


Figure 18 – Projected region and the corresponding packed region with guard bands (informative)

Subclause 7.5.3 is organized as follows:

- The syntax and semantics of the rectangular region packing structure are specified in subclauses 7.5.3.2 and 7.5.3.3, respectively.
- The syntax and semantics of the guard band structure are specified in subclauses 7.5.3.4 and 7.5.3.5, respectively.
- The syntax and semantics of the region-wise packing structure are specified in subclauses 7.5.3.6 and 7.5.3.7, respectively.
- Subclause 7.5.3.8 derives variables from syntax element values of the rectangular region packing, guard band, region-wise packing structures. Subclause 7.5.3.8 also uses the variables to specify constraints for the syntax element values. The variables are also used in other clauses.