
**Information technology — Coded
representation of immersive media —
Part 12:
MPEG immersive video**

*Technologies de l'information — Représentation codée de média
immersifs —*

Partie 12: Vidéo immersive MPEG



Reference number
ISO/IEC 23090-12:2023(E)

Documents
AVA Document Sharing Hub
© ISO/IEC 2023



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2023

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
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword.....	v
Introduction.....	vi
1 Scope.....	1
2 Normative reference.....	1
3 Terms and definitions.....	1
4 Abbreviated terms.....	3
5 Conventions.....	3
6 Overall V3C characteristics, decoding operations, and post-decoding processes.....	4
7 Bitstream format, partitioning, and scanning processes.....	4
7.1 General.....	4
7.2 V3C bitstream formats.....	4
7.3 NAL bitstream formats.....	4
7.4 Partitioning of atlas frames into tiles.....	4
7.5 Tile partition scanning processes.....	4
7.6 Mapping of views to V3C components.....	4
7.7 Sources and outputs.....	5
8 Syntax and semantics.....	6
8.1 Method of specifying syntax in tabular form.....	6
8.2 Specification of syntax functions and descriptors.....	6
8.3 Syntax in tabular form.....	6
8.3.1 General syntax.....	6
8.3.2 V3C unit syntax.....	6
8.3.3 Byte alignment syntax.....	6
8.3.4 V3C parameter set syntax.....	6
8.3.5 NAL unit syntax.....	6
8.3.6 Raw byte sequence payloads, trailing bits, and byte alignment syntax.....	7
8.3.7 Atlas tile data unit syntax.....	7
8.3.8 Supplemental enhancement information message syntax.....	7
8.3.9 V3C MIV extension syntax in tabular form.....	7
8.4 Semantics.....	12
8.4.1 General semantics.....	12
8.4.2 V3C MIV extension semantics.....	12
8.4.3 Order of V3C units and association to coded information.....	19
9 Decoding process.....	20
9.1 General decoding process.....	20
9.2 Atlas data decoding process.....	20
9.2.1 General atlas data decoding process.....	20
9.2.2 Decoding process for a coded atlas frame.....	20
9.2.3 Atlas NAL unit decoding process.....	20
9.2.4 Atlas tile header decoding process.....	20
9.2.5 Decoding process for patch data units.....	20
9.2.6 Decoding process of the block to patch map.....	21
9.2.7 Conversion of tile level patch information to atlas level patch information.....	21
9.3 Occupancy video decoding process.....	22
9.4 Geometry video decoding process.....	22
9.5 Attribute video decoding process.....	22
9.6 Packed video decoding process.....	22
9.7 Common atlas data decoding process.....	22
9.7.1 General common atlas data decoding process.....	22
9.7.2 Decoding process for a coded common atlas frame.....	23
9.7.3 Common atlas NAL unit decoding process.....	23

9.7.4	Common atlas frame order count derivation process.....	23
9.7.5	Common atlas frame MIV extension decoding process.....	23
9.8	Sub-bitstream extraction process.....	28
9.8.1	General.....	28
9.8.2	V3C unit extraction.....	28
9.8.3	NAL unit extraction process.....	28
9.8.4	Group extraction process.....	28
10	Pre-reconstruction process.....	28
11	Reconstruction process.....	28
12	Post-reconstruction process.....	28
13	Adaptation process.....	28
14	Parsing process.....	28
Annex A	(normative) Profiles, tiers, and levels.....	29
Annex B	(informative) Post-decoding conversion to nominal video formats.....	32
Annex C	(informative) V3C sample stream format.....	34
Annex D	(normative) NAL sample stream format.....	35
Annex E	(normative) Atlas hypothetical reference decoder.....	36
Annex F	(normative) Supplemental enhancement information.....	37
Annex G	(informative) Volumetric usability information.....	53
Annex H	(Informative) Overview of the rendering processes.....	54
Bibliography		71

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.

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 or www.iec.ch/members_experts/refdocs).

ISO and IEC draw attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO and IEC take no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO and IEC had received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents and <https://patents.iec.ch>. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

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. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by Joint 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 and IEC websites.

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 and www.iec.ch/national-committees.

Introduction

This document was developed to support compression of immersive video content, in which a real or virtual 3D scene is captured by multiple real or virtual cameras. The use of this document enables storage and distribution of immersive video content over existing and future networks, for playback with 6 degrees of freedom of view position and orientation.

Information technology — Coded representation of immersive media —

Part 12: MPEG immersive video

1 Scope

This document specifies the syntax, semantics and decoding processes for MPEG immersive video (MIV), as an extension of ISO/IEC 23090-5. It provides support for playback of a three-dimensional (3D) scene within a limited range of viewing positions and orientations, with 6 Degrees of Freedom (6DoF).

2 Normative reference

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 23090-5¹⁾, *Information technology — Coded Representation of Immersive Media — Part 5: Visual Volumetric Video-based Coding (V3C) and Video-based Point Cloud Compression (V-PCC)*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 23090-5 and the following apply.

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

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

3.1

atlas sample

position on the rectangular frame associated with an *atlas*

3.2

coded MIV sequence

coded V3C sequence conforming to the constraints specified in this document

3.3

decoding process

process specified in this document that reads a *bitstream* and derives *patch data* and related information that can be used to render a *viewport* (3.22)

3.4

decoding order

order in which syntax elements are processed by the *decoding process* (3.3)

1) Under preparation. Stage at time of publication: ISO/IEC FDIS 23090-5:2023.

3.5
field of view
FOV

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

3.6
MIV access unit

V3C composition unit that is a set of all *sub-bitstream access units* (3.13) that share the same *decoding order* (3.4) count

3.7
MIV coded sub-bitstream sequence

sub-bitstream IRAP access unit (3.14) followed by zero or more *sub-bitstream access units* (3.13)

Note 1 to entry: A *MIV coded sub-bitstream sequence* is a *coded sub-bitstream sequence* conforming to the constraints specified in this document.

3.8
MIV IRAP access unit

MIV access unit (3.6) for which all *sub-bitstream access units* (3.13) are *sub-bitstream IRAP access units* (3.14)

Note 1 to entry: A *MIV IRAP access unit* is a *V3C IRAP composition unit* conforming to the constraints specified in this document.

3.9
multi-plane image
MPI

set of pairs of texture and transparency *attribute frames*, each associated with an implicit constant geometry frame

3.10
renderer

embodiment of a process to create a *viewport* (3.22) from a *volumetric frame* corresponding to a *viewing orientation* (3.19) and *viewing position* (3.20)

3.11
source

one or more video sequences, each containing *geometry* or an *attribute* such as texture and transparency information before encoding

3.12
source view

source (3.11) video material before encoding that corresponds to the format of a *view* (3.15), which may have been acquired by capture of a 3D scene by a real or virtual camera

3.13
sub-bitstream access unit

partition of a *sub-bitstream* that has a certain *decoding order* (3.4) count

Note 1 to entry: A *sub-bitstream access unit* is a *sub-bitstream composition unit*.

3.14
sub-bitstream IRAP access unit

sub-bitstream access unit (3.13) that forms an independent random-access point for the *sub-bitstream*

Note 1 to entry: A *sub-bitstream IRAP access unit* is a *sub-bitstream IRAP composition unit*.

3.15
view

2D rectangular arrays of *view samples* (3.18) consisting of *attribute frames* and corresponding *geometry frame* representing the projection of a *volumetric frame* onto a surface using *view parameters* (3.16)

3.16**view parameters**

parameters of the projection used to generate a *view* (3.15) from a *volumetric frame*, including intrinsic and extrinsic parameters

3.17**view parameters list**

listing of one or more *view parameters* (3.16)

3.18**view sample**

position on the rectangular frame associated with a *view* (3.15)

3.19**viewing orientation**

unit quaternion representing the orientation of a user who is consuming the visual content

3.20**viewing position**

triple of x, y, z characterizing the position in the Cartesian coordinates of a user who is consuming the visual content

3.21**viewing space**

domain constraints for an intended *viewport* (3.22) rendering

Note 1 to entry: The domain is defined in the 3D global space and related to the *viewing orientation* (3.19); it defines a scale between 0 and 1 for every point in space for a given direction of the *viewport* (3.22), to be used by the application.

3.22**viewport**

view (3.15) suitable for display and viewing

4 Abbreviated terms

For the purposes of this document, the abbreviated terms given in ISO/IEC 23090-5 and the following apply.

CSG	constructive solid geometry
ERP	equirectangular projection
HMD	head-mounted display
MIV	MPEG immersive video
OMAF	Omnidirectional media format

5 Conventions

The specifications in ISO/IEC 23090-5:—, Clause 5 and its subclauses apply with the following addition to subclause 5.8:

$\text{Cos}(x)$ the trigonometric cosine function operating on an argument x in units of radians

$\text{Dot}(x, y)$ dot product function, known also as scalar product function, operating on two vectors x and y

$\text{Norm}(x) = \text{Sqrt}(\text{Norm2}(x, x))$

$\text{Norm2}(x) = \text{Abs}(\text{Dot}(x, x))$

$\text{Sin}(x)$ the trigonometric sine function operating on an argument x in units of radians

π the ratio of a circle's circumference to its diameter

6 Overall V3C characteristics, decoding operations, and post-decoding processes

The specifications in ISO/IEC 23090-5:—, Clause 6 apply.

7 Bitstream format, partitioning, and scanning processes

7.1 General

The specifications in ISO/IEC 23090-5:—, subclause 7.1 apply.

7.2 V3C bitstream formats

The specifications in ISO/IEC 23090-5:—, subclause 7.2 apply.

7.3 NAL bitstream formats

The specifications in ISO/IEC 23090-5:—, subclause 7.3 apply.

7.4 Partitioning of atlas frames into tiles

The specifications in ISO/IEC 23090-5:—, subclause 7.4 apply.

7.5 Tile partition scanning processes

The specifications in ISO/IEC 23090-5:—, subclause 7.5 apply.

7.6 Mapping of views to V3C components

This subclause describes the concept of views and its mapping to patches in V3C components.

A view represents a field of view of a volumetric frame for a particular view position and orientation. Each view, at a given time instance, may be represented by one 2D frame providing geometry information plus one 2D frame per attribute, providing attribute information, and occupancy information that may either be embedded within geometry or represented explicitly as a 2D frame. Each view may use the equirectangular, perspective, or orthographic projection format. The atlas components of a view use the same projection format.

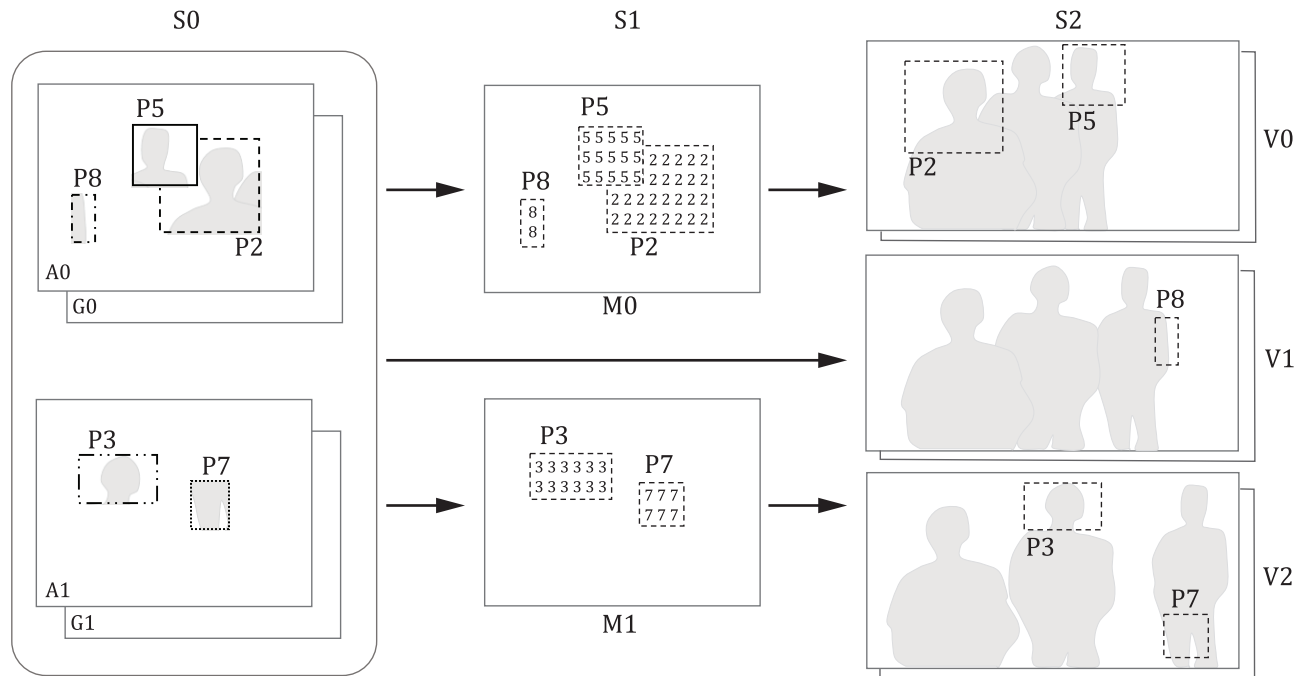
The volumetric frame and all views each have an associated reference frame. Cartesian coordinates of 3D points can therefore be expressed according to the reference frame of the scene, as represented by the volumetric frame, or according to the reference frame of any view. The camera extrinsic parameters (position and orientation) of the views specify the relations between their reference frames, enabling switching of the 3D coordinate system to represent a 3D point from one reference frame attached to a given view to another reference frame attached to another view.

A coded atlas contains information describing the patches within the atlas. For each patch, a view ID is signalled which identifies which view the patch originated from.

A patch represents a rectangular region of a view, with corresponding regions in all present atlas components: attribute(s), geometry, and occupancy. The size (width and height) of each patch in an atlas is signalled. In this version of the document, the size of a patch is always the same as the corresponding

rectangular region in the view texture attribute component, but scaling may optionally be applied to the geometry component or the occupancy component.

[Figure 1](#) shows an illustrative example, in which two atlases contain five patches, which are mapped to three views, with a texture attribute component and a geometry component.



Key

- A0-A1 decoded attribute frames for atlas 0 and 1
- G0-G1 decoded geometry frames for atlas 0 and 1
- M0-M1 maps for atlas 0 and 1
- P0-P8 patches
- S0 stage 0 where attribute and geometry frames are decoded for each atlas
- S1 stage 1 where block to patch mapping is performed
- S2 stage 2 where patches are mapped to views
- V0-V2 reconstructed views

Figure 1 — Example mapping of 5 patches in 2 atlases to 3 views

7.7 Sources and outputs

The volumetric video source that is represented by the bitstream is a sequence of volumetric frames. Each volumetric frame is represented by one or more view frames, each of which may be represented by a geometry picture, an attribute picture for each attribute, and occupancy information, which may be conveyed in the geometry picture or represented separately.

The outputs of the decoding process are described in [subclause 9.1](#).

The outputs of the non-normative rendering process of [Annex H](#) are a sequence of one or more views. The number of views and the associated view parameters may be selected by the application. For example, a single view may be output corresponding to a viewport suitable for display, or a set of views may be output which correspond to the source view parameters.

8 Syntax and semantics

8.1 Method of specifying syntax in tabular form

The specifications in ISO/IEC 23090-5:—, subclause 8.1 apply.

8.2 Specification of syntax functions and descriptors

The specifications in ISO/IEC 23090-5:—, subclause 8.2 apply.

8.3 Syntax in tabular form

8.3.1 General syntax

The specifications in ISO/IEC 23090-5:—, subclause 8.3 apply with the following addition.

An overview of the V3C bitstream structure with MIV extensions is represented in [Figure 2](#).

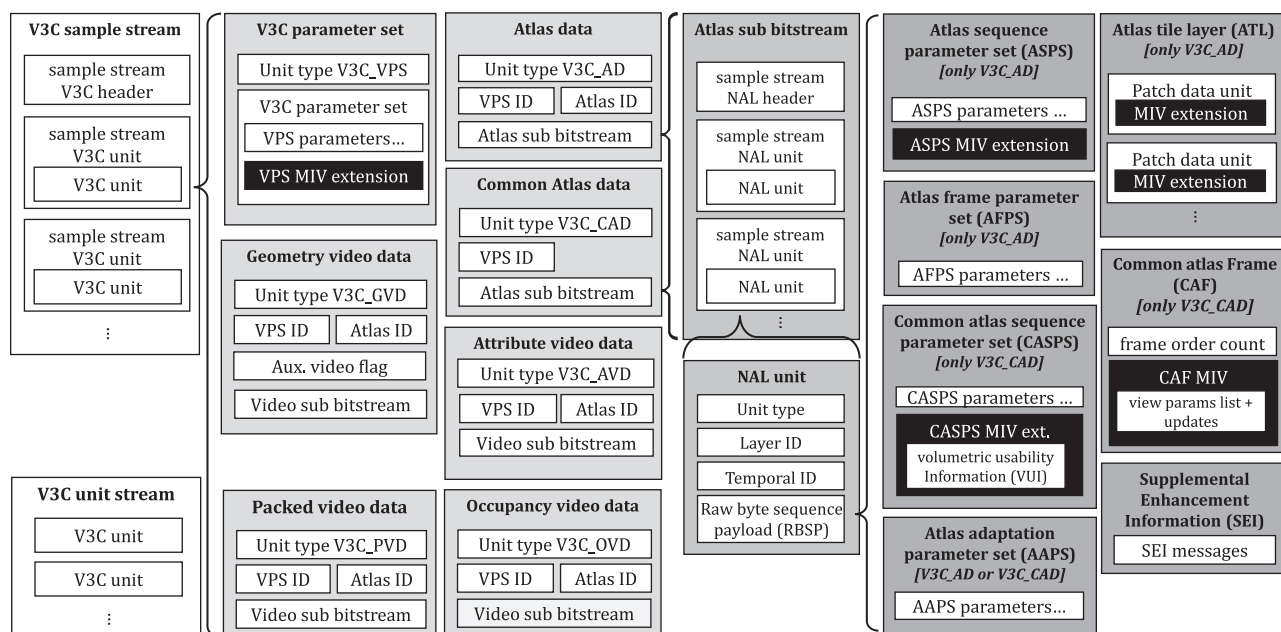


Figure 2 — Overview of V3C bitstream with MIV extensions

8.3.2 V3C unit syntax

The specifications in ISO/IEC 23090-5:—, subclause 8.3.2 apply.

8.3.3 Byte alignment syntax

The specifications in ISO/IEC 23090-5:—, subclause 8.3.3 apply.

8.3.4 V3C parameter set syntax

The specifications in ISO/IEC 23090-5:—, subclause 8.3.4 apply.

8.3.5 NAL unit syntax

The specifications in ISO/IEC 23090-5:—, subclause 8.3.5 apply.

8.3.6 Raw byte sequence payloads, trailing bits, and byte alignment syntax

The specifications in ISO/IEC 23090-5:—, subclause 8.3.6 apply.

8.3.7 Atlas tile data unit syntax

The specifications in ISO/IEC 23090-5:—, subclause 8.3.7 apply.

8.3.8 Supplemental enhancement information message syntax

The specifications in ISO/IEC 23090-5:—, subclause 8.3.8 apply.

8.3.9 V3C MIV extension syntax in tabular form

8.3.9.1 V3C parameter set MIV extension syntax

vps_miv_extension() {	Descriptor
vme_geometry_scale_enabled_flag	u(1)
vme_embedded_occupancy_enabled_flag	u(1)
if(!vme_embedded_occupancy_enabled_flag)	
vme_occupancy_scale_enabled_flag	u(1)
group_mapping()	
}	

8.3.9.2 Group mapping syntax

group_mapping() {	Descriptor
gm_group_count	u(4)
if(gm_group_count > 0)	
for(k = 0; k <= vps_atlas_count_minus1; k++)	
j = vps_atlas_id[k]	
gm_group_id[j]	u(v)
}	

8.3.9.3 Atlas sequence parameter set MIV extension syntax

asps_miv_extension() {	Descriptor
asme_ancillary_atlas_flag	u(1)
asme_embedded_occupancy_enabled_flag	u(1)
if(asme_embedded_occupancy_enabled_flag)	
asme_depth_occ_threshold_flag	u(1)
asme_geometry_scale_enabled_flag	u(1)
if(asme_geometry_scale_enabled_flag) {	
asme_geometry_scale_factor_x_minus1	ue(v)
asme_geometry_scale_factor_y_minus1	ue(v)
}	
if(!asme_embedded_occupancy_enabled_flag)	
asme_occupancy_scale_enabled_flag	u(1)
if(!asme_embedded_occupancy_enabled_flag && asme_occupancy_scale_enabled_flag) {	

asme_occupancy_scale_factor_x_minus1	ue(v)
asme_occupancy_scale_factor_y_minus1	ue(v)
}	
asme_patch_constant_depth_flag	u(1)
asme_patch_attribute_offset_enabled_flag	u(1)
if(asme_patch_attribute_offset_enabled_flag)	
asme_patch_attribute_offset_bit_depth_minus1	ue(v)
asme_max_entity_id	ue(v)
asme_inpaint_enabled_flag	u(1)
}	

8.3.9.4 Atlas frame parameter set MIV extension syntax

afps_miv_extension() {	Descriptor
if(!afps_lod_mode_enabled_flag) {	
afme_inpaint_lod_enabled_flag	u(1)
if(afme_inpaint_lod_enabled_flag) {	
afme_inpaint_lod_scale_x_minus1	ue(v)
afme_inpaint_lod_scale_y_idc	ue(v)
}	
}	
}	

8.3.9.5 Common atlas sequence parameter set MIV extension syntax

casps_miv_extension() {	Descriptor
casme_depth_low_quality_flag	u(1)
casme_depth_quantization_params_present_flag	u(1)
casme_vui_params_present_flag	u(1)
if(casme_vui_params_present_flag)	
vui_parameters()	
}	

8.3.9.6 Common atlas frame

8.3.9.6.1 MIV extension syntax

caf_miv_extension() {	Descriptor
if(nal_unit_type == NAL_CAF_IDR) {	
miv_view_params_list()	
} else {	
came_update_extrinsics_flag	u(1)
came_update_intrinsics_flag	u(1)
if(casme_depth_quantization_params_present_flag)	
came_update_depth_quantization_flag	u(1)
if(came_update_extrinsics_flag)	

miv_view_params_update_extrinsics()	
if(came_update_intrinsics_flag)	
miv_view_params_update_intrinsics()	
if(came_update_depth_quantization_flag)	
miv_view_params_update_depth_quantization()	
}	
}	

8.3.9.6.2 MIV view parameters list syntax

miv_view_params_list() {	Descriptor
mvp_num_views_minus1	u(16)
mvp_explicit_view_id_flag	u(1)
if(mvp_explicit_view_id_flag)	
for(v = 0; v <= mvp_num_views_minus1; v++)	
mvp_view_id[v]	u(16)
for(v = 0; v <= mvp_num_views_minus1; v++) {	
camera_extrinsics(v)	
mvp_inpaint_flag[v]	u(1)
}	
mvp_intrinsic_params_equal_flag	u(1)
for(v = 0; v <= mvp_intrinsic_params_equal_flag ? 0 : mvp_num_views_minus1; v++)	
camera_intrinsics(v)	
if(casme_depth_quantization_params_present_flag) {	
mvp_depth_quantization_params_equal_flag	u(1)
for(v = 0; v <= mvp_depth_quantization_equal_flag ? 0 : mvp_num_views_minus1; v++)	
depth_quantization(v)	
}	
mvp_pruning_graph_params_present_flag	u(1)
if (mvp_pruning_graph_params_present_flag)	
for(v = 0; v <= mvp_num_views_minus1; v++)	
pruning_parents(v)	
}	

8.3.9.6.3 MIV view parameters update extrinsics syntax

miv_view_params_update_extrinsics() {	Descriptor
mvpue_num_view_updates_minus1	u(16)
for(i = 0; i <= mvpue_num_view_updates_minus1; i++) {	
mvpue_view_idx[i]	u(16)
camera_extrinsics(mvpue_view_idx[i])	
}	
}	

8.3.9.6.4 MIV view parameters update intrinsics syntax

miv_view_params_update_intrinsics() {	Descriptor
mvpui_num_view_updates_minus1	u(16)
for(i = 0; i <= mvpui_num_view_updates_minus1; i++) {	
mvpui_view_idx[i]	u(16)
camera_intrinsics(mvpui_view_idx[i])	
}	
}	

8.3.9.6.5 MIV view parameters update depth quantization syntax

miv_view_params_update_depth_quantization() {	Descriptor
mvpudq_num_view_updates_minus1	u(16)
for(i = 0; i <= mvpudq_num_view_updates_minus1; i++) {	
mvpudq_view_idx[i]	u(16)
depth_quantization(mvpudq_view_idx[i])	
}	
}	

8.3.9.6.6 Camera extrinsics syntax

camera_extrinsics(v) {	Descriptor
ce_view_pos_x[v]	fl(32)
ce_view_pos_y[v]	fl(32)
ce_view_pos_z[v]	fl(32)
ce_view_quat_x[v]	i(32)
ce_view_quat_y[v]	i(32)
ce_view_quat_z[v]	i(32)
}	

8.3.9.6.7 Camera intrinsics syntax

camera_intrinsics(v) {	Descriptor
ci_cam_type[v]	u(8)
ci_projection_plane_width_minus1[v]	u(16)
ci_projection_plane_height_minus1[v]	u(16)
if(ci_cam_type[v] == 0) { /* equirectangular */	
ci_erp_phi_min[v]	fl(32)
ci_erp_phi_max[v]	fl(32)
ci_erp_theta_min[v]	fl(32)
ci_erp_theta_max[v]	fl(32)
} else if(ci_cam_type[v] == 1) { /* perspective */	
ci_perspective_focal_hor[v]	fl(32)
ci_perspective_focal_ver[v]	fl(32)
ci_perspective_principal_point_hor[v]	fl(32)

ci_perspective_principal_point_ver[v]	fl(32)
} else if(ci_cam_type[v] == 2) { /* orthographic */	
ci_ortho_width[v]	fl(32)
ci_ortho_height[v]	fl(32)
}	
}	

8.3.9.6.8 Depth quantization syntax

depth_quantization(v) {	Descriptor
dq_quantization_law[v]	ue(v)
if(dq_quantization_law[v] == 0) {	
dq_norm_disp_low[v]	fl(32)
dq_norm_disp_high[v]	fl(32)
}	
dq_depth_occ_threshold_default[v]	ue(v)
}	

8.3.9.6.9 Pruning parents syntax

pruning_parents(v) {	Descriptor
pp_is_root_flag[v]	u(1)
if(!pp_is_root_flag[v]) {	
pp_num_parents_minus1[v]	u(v)
for(i = 0; i <= pp_num_parents_minus1[v]; i++)	
pp_parent_idx[v][i]	u(v)
}	
}	

8.3.9.7 Patch data unit MIV extension syntax

pdu_miv_extension(tileID, p) {	Descriptor
if(asme_max_entity_id > 0)	
pdu_entity_id[tileID][p]	u(v)
if(asme_depth_occ_threshold_flag)	
pdu_depth_occ_threshold[tileID][p]	u(v)
if(asme_patch_attribute_offset_enabled_flag)	
for(c = 0; c < 3; c++) {	
pdu_attribute_offset[tileID][p][c]	u(v)
if(asme_inpaint_enabled_flag)	
pdu_inpaint_flag[tileID][p]	u(1)
}	

8.4 Semantics

8.4.1 General semantics

The specifications in ISO/IEC 23090-5:—, subclause 8.4 apply with the following addition.

It is a requirement of bitstream conformance that `asps_geometry_3d_bit_depth_minus1` and `asps_geometry_2d_bit_depth_minus1` shall be equal to `gi_geometry_3d_coordinates_bit_depth_minus1` and `gi_geometry_2d_bit_depth_minus1`, respectively.

8.4.2 V3C MIV extension semantics

8.4.2.1 V3C parameter set MIV extension semantics

`vme_geometry_scale_enabled_flag` equal to 1 specifies that the V3C sub-bitstream component corresponding to the geometry components, which are determined through either examining if `vuh_unit_type` is equal to `V3C_GVD` or through external means if the V3C unit header is unavailable, may have a different coded picture width and height than the atlas frame width and height, respectively, specified in the active atlas sequence parameter set RBSP. When `vme_geometry_scale_enabled_flag` is equal to 0, it is a requirement of bitstream conformance that the picture width and picture height of the V3C sub-bitstream component corresponding to the geometry components be equal to the atlas frame width and height, respectively, specified in the active atlas sequence parameter set RBSP. When not present, the value of `vme_geometry_scale_enabled_flag` is inferred to be equal to 0.

`vme_embedded_occupancy_enabled_flag` equal to 1 specifies that the occupancy information is present in the V3C sub-bitstream component corresponding to the geometry component, which is determined through either examining if `vuh_unit_type` is equal to `V3C_GVD` or through external means if the V3C unit header is unavailable. `vme_embedded_occupancy_enabled_flag` equal to 0 specifies that occupancy information is not present or present in the V3C sub-bitstream component corresponding to the occupancy component, which is determined through either examining if `vuh_unit_type` is equal to `V3C_OVD` or `V3C_PVD` or through external means if the V3C unit header is unavailable.

`vme_occupancy_scale_enabled_flag` equal to 1 specifies that the V3C sub-bitstream component corresponding to the occupancy components, which are determined through either examining if `vuh_unit_type` is equal to `V3C_OVD` or `V3C_PVD` or through external means if the V3C unit header is unavailable, may have a different coded picture width and height than the atlas frame width and height, respectively, specified in the active atlas sequence parameter set RBSP. When `vme_occupancy_scale_enabled_flag` is equal to 0, it is a requirement of bitstream conformance that the coded picture width and height of all the V3C sub-bitstream components corresponding to the occupancy components be equal to the atlas frame width and height, respectively, specified in the active atlas sequence parameter set RBSP. When not present, the value of `vme_occupancy_scale_enabled_flag` is inferred to be equal to 0.

8.4.2.2 Group mapping semantics

`gm_group_count` specifies the total number of supported groups in the current bitstream. The value of `gm_group_count` shall be in the range of 0 to 15, inclusive.

`gm_group_id[ k ]` specifies the ID of the group for the atlas with ID equal to `k`. The number of bits used to represent `gm_group_id[ k ]` is equal to $\text{Ceil}(\text{Log2}(\text{gm_group_count}))$.

Atlases indicated by the same group ID should be used together for reconstruction. The constant `MaxAtlasCount` is set equal to 64. The 1D array *AtlasGroupId* of size `MaxAtlasCount` is derived as follows:

```
if( gm_group_count > 0 ) {
    for( k = 0; k <= vps_atlas_count_minus1; k++ ) {
        j = vps_atlas_id[ k ]
        AtlasGroupId[ j ] = gm_group_id[ j ]
    }
}
```

8.4.2.3 Atlas sequence parameter set MIV extension semantics

The variable *aspsAtlasID* is set equal to *vuh_atlas_id* of the ASPS RBSP or determined through external means if the V3C unit header is unavailable.

The 1D arrays *AspsFrameWidth* and *AspsFrameHeight* of size *MaxAtlasCount* are derived as follows:

```
AspsFrameWidth[ aspsAtlasID ] = asps_frame_width
AspsFrameHeight[ aspsAtlasID ] = asps_frame_height
```

asme_ancillary_atlas_flag equal to 1 indicates that the patches of the atlas are not intended to be used for view rendering. **asme_ancillary_atlas_flag** equal to 0 indicates that the patches of the atlas are intended to be used for view rendering. When not present, the value of **asme_ancillary_atlas_flag** is inferred to be equal to 0.

NOTE Patches belonging to atlases with this flag set to 1 may describe the unseen rear sides of objects, and be useful in application-specific collision detection, interactions, or scene relighting.

asme_embedded_occupancy_enabled_flag equal to 1 specifies that the occupancy information is present in the V3C sub-bitstream component corresponding to the geometry component, which is determined through either examining if *vuh_unit_type* is equal to *V3C_GVD* or through external means if the V3C unit header is unavailable. **asme_embedded_occupancy_enabled_flag** equal to 0 specifies that occupancy information is not present or present in the V3C sub-bitstream component corresponding to the occupancy component, which is determined through either examining if *vuh_unit_type* is equal to *V3C_OVD* or *V3C_PVD* or through external means if the V3C unit header is unavailable.

It is a requirement of bitstream conformance that the value of **asme_embedded_occupancy_enabled_flag** shall be equal to the value of **vme_embedded_occupancy_enabled_flag** when the V3C VPS is available.

asme_depth_occ_threshold_flag equal to 1 specifies that the *pdu_depth_occ_threshold* syntax element is present in the *pdu_miv_extension()* syntax structure. **asme_depth_occ_threshold_flag** equal to 0 specifies that the *pdu_depth_occ_threshold* syntax element is not present in the *pdu_miv_extension()* syntax structure. When not present, the value of **asme_depth_occ_threshold_flag** is inferred to be equal to 0.

The 1D array *AsmeDepthOccThresholdFlag* of size *MaxAtlasCount* is derived as follows:

```
AsmeDepthOccThresholdFlag[ aspsAtlasID ] = asme_depth_occ_threshold_flag
```

asme_geometry_scale_enabled_flag equal to 1 specifies that the V3C sub-bitstream component corresponding to the geometry components, which are determined through either examining if *vuh_unit_type* is equal to *V3C_GVD* or through external means if the V3C unit header is unavailable, may have a different coded picture width and height than the atlas frame width and height, respectively, specified in the active atlas sequence parameter set RBSP. When **asme_geometry_scale_enabled_flag** is equal to 0, it is a requirement of bitstream conformance that the picture width and picture height of the V3C sub-bitstream component corresponding to the geometry components be equal to the atlas frame width and height, respectively, specified in the active atlas sequence parameter set RBSP. When not present, the value of **asme_geometry_scale_enabled_flag** is inferred to be equal to 0.

It is a requirement of bitstream conformance that the value of **asme_geometry_scale_enabled_flag** shall be equal to the value of **vme_geometry_scale_enabled_flag** when the V3C VPS is available.

asme_geometry_scale_factor_x_minus1 plus 1 specifies a scale factor of the frame width of the geometry video data of the atlas in relation to the nominal atlas width. When not present, the value of **asme_geometry_scale_factor_x_minus1** is inferred to be equal to 0.

The 1D arrays *AsmeGeometryScaleFactorX* and *AsmeGeometryFrameWidth* of size *MaxAtlasCount* are derived as follows:

```
AsmeGeometryScaleFactorX[ aspsAtlasID ] = asme_geometry_scale_factor_x_minus1 + 1
AsmeGeometryFrameWidth[ aspsAtlasID ] =
```

$$\text{AspsFrameWidth}[\text{ aspsAtlasID }] / \text{AsmeGeometryScaleFactorX}[\text{ aspsAtlasID }]$$

It is a requirement of bitstream conformance that $\text{AspsFrameWidth}[\text{ aspsAtlasID }] \% \text{AsmeGeometryScaleFactorX}[\text{ aspsAtlasID }]$ shall be equal to 0.

asme_geometry_scale_factor_y_minus1 plus 1 specifies a scale factor of the frame height of the geometry video data of the atlas in relation to the nominal atlas height. When not present, the value of **asme_geometry_scale_factor_y_minus1** is inferred to be equal to 0.

The 1D arrays *AsmeGeometryScaleFactorY* and *AsmeGeometryFrameHeight* of size MaxAtlasCount are derived as follows:

$$\text{AsmeGeometryScaleFactorY}[\text{ aspsAtlasID }] = \text{asme_geometry_scale_factor_y_minus1} + 1$$

$$\begin{aligned} \text{AsmeGeometryFrameHeight}[\text{ aspsAtlasID }] = \\ \text{AspsFrameHeight}[\text{ aspsAtlasID }] / \text{AsmeGeometryScaleFactorY}[\text{ aspsAtlasID }] \end{aligned}$$

It is a requirement of bitstream conformance that $\text{AspsFrameHeight}[\text{ aspsAtlasID }] \% \text{AsmeGeometryScaleFactorY}[\text{ aspsAtlasID }]$ shall be equal to 0.

asme_occupancy_scale_enabled_flag equal to 1 specifies that the V3C sub-bitstream component corresponding to the occupancy components, which are determined through either examining if **vuh_unit_type** is equal to V3C_OVD or V3C_PVD or through external means if the V3C unit header is unavailable, may have a different coded picture width and height than the atlas frame width and height, respectively, specified in the active atlas sequence parameter set RBSP. When **asme_occupancy_scale_enabled_flag** is equal to 0, it is a requirement of bitstream conformance that the coded picture width and height of all the V3C sub-bitstream components corresponding to the occupancy components be equal to the atlas frame width and height, respectively, specified in the active atlas sequence parameter set RBSP. When not present, the value of **asme_occupancy_scale_enabled_flag** is inferred to be equal to 0.

asme_occupancy_scale_factor_x_minus1 plus 1 specifies a scale factor of the frame width of the occupancy video data of the atlas in relation to the nominal atlas width. When not present, the value of **asme_occupancy_scale_factor_x_minus1** is inferred to be equal to 0.

The 1D arrays *AsmeOccupancyScaleFactorX* and *AsmeOccupancyFrameWidth* of size MaxAtlasCount are derived as follows:

$$\text{AsmeOccupancyScaleFactorX}[\text{ aspsAtlasID }] = \text{asme_occupancy_scale_factor_x_minus1} + 1$$

$$\begin{aligned} \text{AsmeOccupancyFrameWidth}[\text{ aspsAtlasID }] = \\ \text{AspsFrameWidth}[\text{ aspsAtlasID }] / \text{AsmeOccupancyScaleFactorX}[\text{ aspsAtlasID }] \end{aligned}$$

asme_occupancy_scale_factor_y_minus1 plus 1 specifies a scale factor of the frame height of the occupancy video data of the atlas in relation to the nominal atlas height. When not present, the value of **asme_occupancy_scale_factor_y_minus1** is inferred to be equal to 0.

The 1D arrays *AsmeOccupancyScaleFactorY* and *AsmeOccupancyFrameHeight* of size MaxAtlasCount are derived as follows:

$$\text{AsmeOccupancyScaleFactorY}[\text{ aspsAtlasID }] = \text{asme_occupancy_scale_factor_y_minus1} + 1$$

$$\begin{aligned} \text{AsmeOccupancyFrameHeight}[\text{ aspsAtlasID }] = \\ \text{AspsFrameHeight}[\text{ aspsAtlasID }] / \text{AsmeOccupancyScaleFactorY}[\text{ aspsAtlasID }] \end{aligned}$$

asme_patch_constant_depth_flag equal to 1 indicates that the recommended depth for a patch be derived based on patch data unit parameters instead of the geometry video data. **asme_patch_constant_depth_flag** equal to 0 specifies that when **vps_geometry_video_present_flag**[*aspsAtlasID*] is equal to 0, the depth is determined by external means.

The 1D array *AsmePatchConstantDepthFlag* of size MaxAtlasCount is derived as follows:

$$\text{AsmePatchConstantDepthFlag}[\text{ aspsAtlasID }] = \text{asme_patch_constant_depth_flag}$$

asme_patch_attribute_offset_enabled_flag equal to 1 indicates that the **asme_patch_attribute_offset_bit_depth_minus1** syntax element is present in the syntax structure. **asme_patch_attribute_offset_enabled_flag** equal to 0 indicates that the **asme_patch_attribute_offset_bit_depth_minus1** syntax element is not present in the syntax structure.

asme_patch_attribute_offset_bit_depth_minus1 plus 1 specifies the number of bits used to represent the **pdu_attribute_offset[tileID][p][c]** syntax element. **asme_patch_attribute_offset_bit_depth_minus1** shall be in the range of 0 to **ai_attribute_2d_bit_depth_minus1**, inclusive.

asme_max_entity_id specifies the maximum value that can be indicated for the patch entity ID syntax element, **pdu_entity_id[i][j]**, for a patch with index *j* in a tile with tile ID equal to *i*. When not present, the value of **asme_max_entity_id** is inferred to be equal to 0.

asme_inpaint_enabled_flag equal to 1 specifies that the **pdu_inpaint_flag[tileID][p]** syntax elements are present in the **pdu_miv_extension()** syntax structure. **asme_inpaint_enabled_flag** equal to 0 specifies that the **pdu_inpaint_flag[tileID][p]** syntax elements are not present in the **pdu_miv_extension()** syntax structure. When not present, the value of **asme_inpaint_enabled_flag** is inferred to be equal to 0.

8.4.2.4 Atlas frame parameter set MIV extension semantics

afme_inpaint_lod_enabled_flag equal to 1 specifies that the inpaint LOD parameters are present. If **afme_inpaint_lod_enabled_flag** is equal to 0, no inpaint LOD parameters are present. When not present, the value of **afme_inpaint_lod_enabled_flag** is inferred to be equal to 0.

afme_inpaint_lod_scale_x_minus1 specifies the inpaint LOD scaling factor to be applied to the local *x* coordinate of a point in any patch with index *p* of the current atlas tile, with tile ID equal to **tileID**, and with **pdu_inpaint_flag[tileID][p]** equal to 1, prior to its addition to the patch coordinate *TilePatch3dOffsetU[tileID][p]*. When not present, the value of **afme_inpaint_lod_scale_x_minus1** is inferred to be equal to 0.

afme_inpaint_lod_scale_y_idc indicates the LOD scaling factor to be applied to the local *y* coordinate of a point in any patch with index *p* of the current atlas tile, with tile ID equal to **tileID**, and with **pdu_inpaint_flag[tileID][p]** equal to 1, prior to its addition to the patch coordinate *TilePatch3dOffsetV[tileID][p]*. When not present, the value of **afme_inpaint_lod_scale_y_idc[tileID][p]** is inferred to be equal to 0.

8.4.2.5 Common atlas sequence parameter set MIV extension semantics

casme_depth_low_quality_flag equal to 1 indicates that the depth fidelity confidence in geometry video sub-bitstreams is low. **casme_depth_low_quality_flag** equal to 0 indicates that the depth fidelity confidence is unknown. When not present, the value of **casme_depth_low_quality_flag** is inferred to be equal to 0.

NOTE Low depth fidelity indicates inconsistency in depth values between views. The encoder may set this flag to 1 for depth estimated on a natural content with an image-based depth estimation method. The encoder may set this flag to 0 for depth derived from a 3D model on a computer generated content.

casme_depth_quantization_params_present_flag equal to 1 indicates that the depth quantization parameters are present in the syntax structure. **casme_depth_quantization_params_present_flag** equal to 0 indicates that the depth quantization parameters are not present in the syntax structure. When not present, the value of **casme_depth_quantization_params_present_flag** is inferred to be equal to 1.

casme_vui_params_present_flag equal to 1 indicates that the **vui_parameters()** syntax structure is present in this syntax structure. **casme_vui_params_present_flag** equal to 0 indicates that the **vui_parameters()** syntax structure is not present in this syntax structure. It is a requirement of bitstream conformance that the value of **casme_vui_params_present_flag** shall be equal to 0 for all non-IRAP access units.

8.4.2.6 Common atlas frame MIV extension semantics

came_update_extrinsics_flag equal to 1 indicates that the `miv_view_params_update_extrinsics( )` syntax structure is present in this syntax structure. **came_update_extrinsics_flag** equal to 0 indicates that the `miv_view_params_update_extrinsics( )` syntax structure is not present in this syntax structure.

came_update_intrinsics_flag equal to 1 indicates that the `miv_view_params_update_intrinsics( )` syntax structure is present in this syntax structure. **came_update_intrinsics_flag** equal to 0 indicates that the `miv_view_params_update_intrinsics( )` syntax structure is not present in this syntax structure.

came_update_depth_quantization_flag equal to 1 indicates that the `miv_view_params_update_depth_quantization( )` syntax structure is present in this syntax structure. **came_update_depth_quantization_flag** equal to 0 indicates that the `miv_view_params_update_depth_quantization( )` syntax structure is not present in this syntax structure. When not present, the value of **came_update_depth_quantization_flag** is inferred to be equal to 0.

8.4.2.6.1 MIV view parameters list semantics

mvp_num_views_minus1 plus 1 specifies the maximum number of views in an MIV view parameters list representing a volumetric frame.

mvp_explicit_view_id_flag equal to 1 specifies that the syntax element `mvp_view_id[ v ]` is present in the `miv_view_params_list( )` syntax structure. **mvp_explicit_view_id_flag** equal to 0 specifies that the syntax element `mvp_view_id[ v ]` is not present in the `miv_view_params_list( )` syntax structure.

mvp_view_id[v] specifies the view ID of the view with index *v*. The value of `mvp_view_id[ v ]` shall be in the range of 0 to 65 535, inclusive. It is a requirement of bitstream conformance to this version of this document that the value of `mvp_view_id[ j ]` shall not be equal to `mvp_view_id[ k ]` for all *j* != *k*. When not present, the value of `mvp_view_id[ v ]` is inferred to be equal to *v*.

mvp_inpaint_flag[v] equal to 1 specifies that the view with index *v* is an inpaint view.

NOTE 1 A renderer can skip inpaint views.

NOTE 2 A renderer can use inpaint views to render viewport regions that would otherwise require inpainting.

mvp_intrinsic_params_equal_flag equal to 1 specifies that the intrinsic parameters are signalled only for the view with index 0 and each view with index in range of 1 to **mvp_num_views_minus1**, inclusive, has the same intrinsic parameters as the view with index 0. **mvp_intrinsic_params_equal_flag** equal to 0 specifies that the intrinsic parameters are explicitly signalled for each view with index in range of 0 to **mvp_num_views_minus1**, inclusive.

NOTE When **mvp_intrinsic_params_equal_flag** == 1, it is still allowed to send intrinsics updates for specific cameras.

mvp_depth_quantization_params_equal_flag equal to 1 specifies that the depth quantization parameters are signalled only for the view with index 0 and each view with index in range of 1 to **mvp_num_views_minus1**, inclusive, has the same depth quantization parameters as the view with index 0. **mvp_depth_quantization_params_equal_flag** equal to 0 specifies that the depth quantization parameters are explicitly signalled for each view with index in range of 0 to **mvp_num_views_minus1**, inclusive.

NOTE When **mvp_depth_quantization_params_equal_flag** == 1, it is still allowed to send depth quantization updates for specific cameras.

mvp_pruning_graph_params_present_flag equal to 1 specifies that the `pruning_parents( )` syntax structure is present in the `miv_view_params_list( )` syntax structure. **mvp_pruning_graph_params_present_flag** equal to 0 specifies that the `pruning_parents( )` syntax structure is not present in the `miv_view_params_list( )` syntax structure.

8.4.2.6.2 MIV view parameters update extrinsics semantics

mvpue_num_view_updates_minus1 plus 1 specifies the number of camera_extrinsics(*v*) syntax structures that are present within this syntax structure. The value of mvpue_num_view_updates_minus1 shall be in the range of 0 to mvp_num_views_minus1, inclusive.

mvpue_view_idx[*i*] specifies the view index for which updated camera extrinsic parameters will be signalled. The value of mvpue_view_idx[*i*] shall be in the range of 0 to mvp_num_views_minus1, inclusive. It is a requirement of bitstream conformance to this version of this document that the value of mvpue_view_idx[*j*] shall not be equal to mvpue_view_idx[*k*] for all *j* != *k*.

8.4.2.6.3 MIV view parameters update intrinsics semantics

mvpui_num_view_updates_minus1 plus 1 specifies the number of camera_intrinsics(*v*) syntax structures that are present within this syntax structure. The value of mvpui_num_view_updates_minus1 shall be in the range of 0 to mvp_num_views_minus1.

mvpui_view_idx[*i*] specifies the view index for which updated camera intrinsic parameters will be signalled. The value of mvpui_view_idx[*i*] shall be in the range of 0 to mvp_num_views_minus1, inclusive. It is a requirement of bitstream conformance to this version of this document that the value of mvpui_view_idx[*j*] shall not be equal to mvpui_view_idx[*k*] for all *j* != *k*.

8.4.2.6.4 MIV view parameters update depth quantization semantics

mvpudq_num_view_updates_minus1 plus 1 specifies the number of depth_quantization(*v*) syntax structures that are present within this syntax structure. The value of mvpudq_num_view_updates_minus1 shall be in the range of 0 to mvp_num_views_minus1, inclusive.

mvpudq_view_idx[*i*] specifies the view index for which updated depth quantization parameters will be signalled. The value of mvpudq_view_idx[*i*] shall be in the range of 0 to mvp_num_views_minus1, inclusive. It is a requirement of bitstream conformance to this version of this document that the value of mvpudq_view_idx[*j*] shall not be equal to mvpudq_view_idx[*k*] for all *j* != *k*.

8.4.2.6.5 Camera extrinsics semantics

ce_view_pos_x[*v*] specifies in scene units the x-coordinate of the location of the view with view index equal to *v*.

ce_view_pos_y[*v*] specifies in scene units the y-coordinate of the location of the view with view index equal to *v*.

ce_view_pos_z[*v*] specifies in scene units the z-coordinate of the location of the view with view index equal to *v*.

ce_view_quat_x[*v*] specifies the x component, $qX[v]$, for the rotation of the view with view index equal to *v* using the quaternion representation. The value of ce_view_quat_x[*v*] shall be in the range of -2^{30} to 2^{30} , inclusive. When ce_view_quat_x[*v*] is not present, its value shall be inferred to be equal to 0. The value of $qX[v]$ is computed as follows:

$$qX[v] = ce_view_quat_x[v] \div 2^{30} \quad (1)$$

ce_view_quat_y[*v*] specifies the y component, $qY[v]$, for the rotation of the view with view index equal to *v* using the quaternion representation. The value of ce_view_quat_y[*v*] shall be in the range of

-2^{30} to 2^{30} , inclusive. When $ce_view_quat_y[v]$ is not present, its value shall be inferred to be equal to 0. The value of $qY[v]$ is computed as follows:

$$qY[v] = ce_view_quat_y[v] \div 2^{30} \quad (2)$$

$ce_view_quat_z[v]$ specifies the z component, $qZ[v]$, for the rotation of the view with view index equal to v using the quaternion representation. The value of $ce_view_quat_z[v]$ shall be in the range of -2^{30} to 2^{30} , inclusive. When $ce_view_quat_z[v]$ is not present, its value shall be inferred to be equal to 0. The value of $qZ[v]$ is computed as follows:

$$qZ[v] = ce_view_quat_z[v] \div 2^{30} \quad (3)$$

The fourth component, $qW[v]$, of the quaternion is calculated as follows:

$$qW[v] = \text{Sqrt}(\text{Max}(0, 1 - (qX[v]^2 + qY[v]^2 + qZ[v]^2))) \quad (4)$$

8.4.2.6.6 Camera intrinsics semantics

ci_cam_type[v] specifies the projection method of the view with view index equal to v. **ci_cam_type[v]** equal to 0 specifies ERP projection. **ci_cam_type[v]** equal to 1 specifies a perspective projection. **ci_cam_type[v]** equal to 2 specifies an orthographic projection. **ci_cam_type** values in range 3 to 127 are reserved for future use by ISO/IEC. **ci_cam_type** values in range 128 to 255 are unspecified (available for specification by other standards).

ci_projection_plane_width_minus1[v] plus 1 and **ci_projection_plane_height_minus1[v]** plus 1 specify the horizontal and vertical resolutions of the camera projection planes, respectively, expressed in coded luma samples.

ci_erp_phi_min[v] and **ci_erp_phi_max[v]** specify the longitude range (minimum and maximum values) for an ERP projection, as floating-point in units of degrees. **ci_erp_phi_min[v]** and **ci_erp_phi_max[v]** shall be in the range -180 to 180 .

ci_erp_theta_min[v] and **ci_erp_theta_max[v]** specify the latitude range (minimum and maximum values) for an ERP projection, as floating-point in units of degrees. **ci_erp_theta_min[v]** and **ci_erp_theta_max[v]** shall be in the range -90 to 90 .

ci_perspective_focal_hor[v] and **ci_perspective_focal_ver[v]** are floating-point values that specify in luma sample position units the horizontal and vertical components, respectively, of the focal of a perspective projection.

ci_perspective_principal_point_hor[v] and **ci_perspective_principal_point_ver[v]** are floating-point values that specify in luma sample position units the horizontal and vertical coordinates, respectively, of the principal point of a perspective projection (intersection of optical axis with image plane).

ci_ortho_width[v] and **ci_ortho_height[v]** are positive floating-point values that specify in scene units (e.g. meters) the horizontal and vertical dimensions of the captured part of the volumetric frame.

8.4.2.6.7 Depth quantization semantics

dq_quantization_law[v] specifies the type of depth quantization method of the view with view index equal to v. **dq_quantization_law[v]** equal to 0 specifies a uniform quantization of the inverse of depth values. Even values of **dq_quantization_law[v]** greater than 0 are reserved for future use by ISO/IEC. Odd values of **dq_quantization_law[v]** are outside the scope of this document.

dq_norm_disp_low[v] and **dq_norm_disp_high[v]** specify the normalized disparity of the lowest and highest signalled geometry values, respectively, in scene units⁻¹ (e.g. meters⁻¹) of the view with view index equal to v.