ISO/IEC JTC 1/SC 29/WG 03 N1198

**ISO/IEC JTC 1/SC 29/WG 03  
MPEG Systems   
Convenorship: KATS (Korea, Republic of)**

**Document type:** Output Document

**Title:** Technologies under consideration on carriage of V3C data

**Status:** Approved

**Date of document:** 2024-06-05

**Source:** ISO/IEC JTC 1/SC 29/WG 03

**No. of pages:** 119 (with cover page)

**Email of Convenor:** young.L @ samsung . com

**Committee URL:** <https://isotc.iso.org/livelink/livelink/open/jtc1sc29wg3>

**INTERNATIONAL ORGANIZATION FOR STANDARDIZATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

**ISO/IEC JTC 1/SC 29/WG 03 MPEG SYSTEMS**

**ISO/IEC JTC 1/SC 29/WG 03 N1198**

**April 2024, Rennes, FR**

|  |  |
| --- | --- |
| **Title** | **Technologies under consideration on carriage of V3C data** |
| **Source** | **WG 03, MPEG Systems** |
| **Status** | **Approved** |
| **Serial Number** | **23478** |

# Introduction

This document contains technologies under consideration for carriage of point cloud data. The following contributions are covered:

* DASH signalling for V-PCC (m47436, m49110, m49109, m48117, m54337)
* Signaling of V-PCC and 3D Video Tracks and their Spatial Relationship in MPEG-DASH (m50982)
* DASH-based Signaling of Recommended Viewport Information (m50654, m54310)
* Dynamic Signaling of User-Selected and Recommended Viewports for PC Data (m50655, m54310)
* Encapsulation and Signalling of V-PCC data in MMT (m50769, m52792)
* Timed Metadata Signaling and Storage for Point Cloud Data (m50661, m52443, m54308)
* Carriage of multiple atlases (m54292)
* Carriage of V-PCC data with Multiple Atlases (m53503)
* VPCC spatial region descriptor (m53504)
* Carriage of non-timed V3C data (m54293)
* Partial access of V3C data (m54396)
* Support for partial access utilizing spatial scalability (m54354)
* Visibility cone information indication in system layer (m54355)
* Comments on signaling of static and dynamic regions for V3CD (m56828)
* On Defining Common Metadata for MPEG-I Immersive Media (m57821)
* On harmonization of metadata for MPEG-I systems (m57815)
* Shared common atlas and atlas data track (m59320)
* Timed and non-timed V3C components in ISOBMFF (m63055)
* Supporting to decode with single decoder instance for V-PCC content (m65113)
* Supplemental media in volumetric video in ISOBMFF (m64929)
* Support of 2D snapshots (m66538)

# DASH signalling for V-PCC (m47436, m49110, m49109, m54337)

## Signaling V-PCC Components in DASH MPD

(Ed. Note (MPEG 128): The following sections are related to having multiple representations in the same AdaptationSet and Tile Grouping, which was agreed in MPEG 128 to be left out of the CD text. Hence, they are left in the TuC)

The main **AdaptationSet** shall contain either a single initialization segment at the adaptation set level or multiple initialization segments at the representation level (one for each **Representation**). Initialization segments should contain V-PCC sequence parameter sets, as defined in [15], needed to initialize the V-PCC decoder.

In the case of a single initialization segment, V-PCC sequence parameter sets for all **Representation**s shall be contained in the initialization segment. When more than one **Representation** is signaled in the main **AdaptationSet**, the initialization segment of each **Representation** shall contain the V-PCC sequence parameter sets for that particular **Representation**.

## Content adaptation and Indication of valid grouping of V-PCC DASH Representations (m48117)

The most trivial way to adapt V-PCC content to varying bandwidth conditions is either by selecting different bitrate representations of components or by streaming a subset of available V-PCC components (i.e. attributes) to save bandwidth.

However, in addition to the above, it is possible to have adaptation based on:

* resolution of the video coded attributes (e.g. full resolution of texture, half resolution of texture, etc.)
* Utilized video codec for the video coded attributes

Naïve creation of adaptation sets using different codecs, quality parameters, and resolutions may lead to undefined player behavior or may require massive replication of VPCC patch-level data, thus increasing content creation/consumption complexity prior to and during a streaming session.

Above-mentioned adaptation options are orthogonal to bitrate adaptation; hence they may need to be defined as new adaptation sets (i.e. having different bitrate representations). Utilization of different *preselection* listings could enable initial grouping of different codec and resolution options.

In current smartphones, it is possible to have different number of video codec instances per codec type. The table below shows examples of maximum allowed decoder instances per codec and per resolution on different mobile devices that we tested:

Table 10.3: Maximum allowed video decoder instances, resolutions and codec types

|  |  |  |  |
| --- | --- | --- | --- |
|  | Decoder Configuration 1 | Decoder Configuration 2 | Decoder Configuration 3 |
| Device 1 | 4x 4K HEVC | 2x 4K HEVC + 2x4K AVC | 2x4K HEVC + 5xHD HEVC + 4K AVC |
| Device 2 | 4x 4K HEVC | 3x 4K HEVC + 5xHD HEVC |  |
| Device 3 | 7x4K HEVC |  |  |

Tests also showed that one can interchange HEVC with AVC decoders in the above table.

The tests indicate that there are different options for the mobile devices for selecting the coded media versions of V-PCC components for managing the desired media processing capacity: based on number of instances, resolution and codec type.

Obviously, a single V-PCC track and SPS would not be able to provide such interchangeable codec options. As an example, *codec\_ids* are part of SPS data which is stored in the V-PCC track. On the other hand, it is not acceptable to push the complexity to the content creation side and expect the content creator to generate different V-PCC tracks containing different combinations of codecs and resolutions.

It should be possible to signal an update on certain parameters of SPSs so that options like codec switching is doable without a massive effort of new V-PCC track generation. It is encouraged that MPEG Systems experts find a solution to this issue together with MPEG 3DG experts.

Another aspect of adaptation would be to stop streaming of some V-PCC attributes when there is not enough bandwidth. For example, during a DASH streaming session, normal and reflectance attributes’ bitstreams may be preferred not to be streamed anymore if the bandwidth drops below a certain threshold. A similar case occurs if some attributes have a shorter duration than the overall presentation. Such a use case was brought to the participants’ attention in PCC systems AHG meeting on 13-14.6.2019 at Palo Alto, USA [31]. For such cases, the V-PCC decoder needs to know that a particular attribute’s bitstream data will no longer be available. Otherwise, there would be a rendering interrupt situation where V-PCC decoder may wait for some attribute bitstream data which will be no longer available.

It should be possible to signal an update on the number of attributes listed in the SPS so that V-PCC decoder would be aware of the number of streamed attributes. A contribution [32] is adopted during MPEG 127 by 3DG group .

### Indication of valid grouping of representations

During the PCC systems AHG meeting on 13-14.6.2019 at Palo Alto, USA [5], it was made clear by V-PCC experts that not all combinations of DASH Representations may provide the desired visual experience to the viewer. For example, not all occupancy and geometry combinations make visual sense. Hence, “pre-defined combinations” need to be signaled.

It is proposed that **ContentComponent** element’s **@tag** attribute is utilized at Representation level to indicate the list of such pre-defined combinations. A comma separated list of *integer identifiers* is sufficient to signal pre-defined representation combinations.

Representations of different Adaptation Sets which have the same identfier in **ContentComponent@tag** attribute would indicate a pre-defined combination and can be consumed together. An identifier of value “0” can indicate that representation is consumable in any combination. The selection process for such pre-defined combinations is out of scope of the specification.

An example can be as follows:

Contents of DASH MPD of a V-PCC encoded content with one texture attribute:

* Adaptation Set 1: contains V-PCC track and a single representation.
  + Representation 1**: ContentComponent@tag = “0”**
* Adaptation Set 2: contains Occupancy component track and 3 Representations
  + Representation 1: {HEVC, @10 mbps. **ContentComponent@tag = “1”}**
  + Representation 2: {HEVC, @20 mbps. **ContentComponent@tag = “2”}**
  + Representation 3: {HEVC, @30 mbps. **ContentComponent@tag = “2,3”}**
* Adaptation Set 3: contains Geometry component track and 3 Representations
  + Representation 1: {HEVC, @10 mbps. **ContentComponent@tag = “1”}**
  + Representation 2: {HEVC, @20 mbps. **ContentComponent@tag = “2”}**
  + Representation 3: {HEVC, @40 mbps. **ContentComponent@tag = “3”}**
* Adaptation Set 4: contains Texture Attribute component track and 4 Representations
  + Representation 1: {HEVC, @10 mbps. **ContentComponent@tag = “0”}**
  + Representation 2: {HEVC, @20 mbps. **ContentComponent@tag = “1,2”}**
  + Representation 3: {HEVC, @40 mbps. **ContentComponent@tag = “2,3”}**
  + Representation 3: {HEVC, @60 mbps. **ContentComponent@tag = “3”}**

Representations which have identifier value “0” in their comma separated list of identifiers in **ContentComponent@tag** can be consumed with any Representation.

Representations which have identifier value X in their comma separated list of identifiers in **ContentComponent@tag** can be consumed with any other Representation which have the same value X in their comma separated list of identifiers in **ContentComponent@tag.**

## Mapping Atlas Tile Adaptation Sets to Static Spatial Regions (m54337)

To associate Adaptation Sets for a V3C atlas tile, including the Adaptation Set for the V3C atlas tile track and associated Adaptation Sets for the video-coded components, a **V3CRegionId** descriptor shall be used. This descriptor is a **SupplementalProperty** element with a @schemeIdUri attribute equal to "urn:mpeg:mpegI:v3c:2020:vpsr". A single **V3CRegionId** descriptor shall be present at the adaptation set level or the representation level in the V3C atlas tile Adaptation Set or at the preselection level for the Preselection representing the atlas tile.

The @value attribute of the **V3CRegionId** shall be equal to the value of @id for one of the spatial regions listed in the **V3C3DRegions** descriptor in the Main V3C Adaptation Set.

If the **V3CRegionId** descriptor is present in an Adaptation Set, this indicates that the Adaptation Set is part of a set of Adaptation Sets associated with the spatial region whose id is given by the descriptor. If the **V3CRegionId** descriptor is present in a Preselection, this indicates that the Adaptation Sets that are part of this Preselection are associated with the spatial region whose id is given by the descriptor.

## Dynamic Volumetric Metadata (m54337)

Dynamic volumetric timed-metadata tracks carrying dynamic spatial region information and scene object information, as specified in clause 8.5 of ISO/IEC 23090-10, shall be represented in the DASH MPD file by a separate Adaptation Set with a single Representation with the following constraints:

* @codecs attribute shall have the value 'dyvm'.
* @associationId attribute may contain an ID of a V3C Main Adaptation Set signalled in the MPD.

The sole Representation of dynamic volumetric metadata Adaptation Set is therefore associated (linked) with the Representations in V3C Main Adaptation Set using an @associationId attribute, defined in ISO/IEC 23009-1 ‎[5], with a @associationType attribute value that includes the 4CC 'dyvm'

# Signaling of V-PCC and 3D Video Tracks and their Spatial Relationship in MPEG-DASH (m50982)

## General

This contribution is to propose a 3D spatial relationship description in DASH (3rd Edition) for signaling of V-PCC tracks, when they are used to carry point cloud data, and their spatial relationship, especially when they tracks are encapsulations of multiple regions in a V-PCC bit-stream. The proposed description can be applied to immersive video tracks in general.

## Background: Encapsulation and Signaling of Multiple Regions in V-PCC Tracks

When a 3D source, like a point-cloud, is sub-divided into multiple regions, like sub-point-clouds (or V-PCC tiles), for the purposes of partial delivery and access, the regions can be encapsulated at either the V-PCC bit-stream level or the patch data group level.

If a V-PCC bit-stream is carried as a single track, then multiple regions of the bit-stream can be signaled in the following two cases:

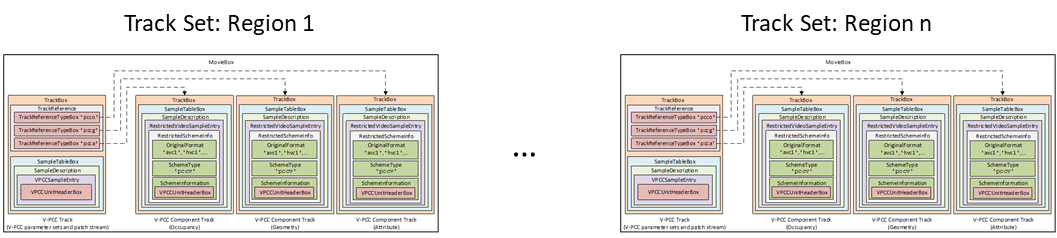
* No additional single track carriage of the multiple regions: multiple timed metadata tracks of type '6dtr' or '6dsr' for the regions are linked to the V-PCC track, to signal that there are multiple regions in the (source) bit-stream as specified in the metadata tracks. All the timed metadata tracks shall have a same source\_id value to signal their regions coming from a same source.
* Additional single track carriage of the multiple regions: each region track shall contain a track grouping box of type '6dtr' or '6dsr' for signaling that the track carries the content of a region of a source indicated by the metadata in the track grouping box. All the track grouping boxes shall have a same source\_id value to signal their regions coming from a same source.

If a V-PCC bit-stream is carried using the multiple track approach within a separate V-PCC track and a number of component tracks, as depicted in the following file format container structure:

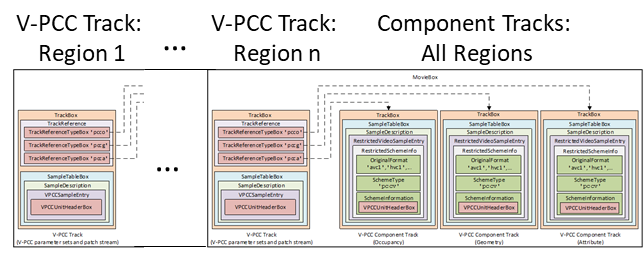


then multiple regions of the source V-PCC streams can be signaled at the V-PCC bit-stream or patch data group levels, in order to encapsulate V-PCC content of the regions, respectively.

At the bit-stream level, there are multiple V-PCC bit-streams. The content of the multiple regions are encapsulated in multiple sets of ISOBMFF volumetric and components tracks as shown below, so that each set of the track represents one of the multiple regions, which corresponds to one of the multiple V-PCC bit-streams.



At the patch group level, there are multiple V-PCC tracks together with a common set of component tracks. The content of the multiple regions are encapsulated in the patch data groups of the multiple V-PCC track, together with the common set of the components tracks as shown below, so that each V-PCC track in conjunction with the common component tracks represents one of the multiple regions.



Correspondingly, in the both cases, the spatial relationship of the (V-PCC) regions and their source can be signed using the following track grouping and timed metadata mechanisms:

* Track grouping box: each V-PCC track carries a TrackGroupTypeBox of type '6dtr', SpatialRelationship3DTileRegionsBox. This covers the both cases of encapsulating V-PCC regions, as above. All V-PCC tracks with track grouping boxes containing a same source\_id represent regions of a same source, when coupled with their corresponding component tracks.

Timed metadata track: each V-PCC track is referenced by a timed metadata track of sample entry type '6dtr'. This covers the both cases of encapsulating V-PCC regions, as above. All V-PCC tracks referenced by timed metadata tracks with a same source\_id represent regions of a same source, when coupled with their corresponding component tracks.

## 3D Spatial Relationship Description in DASH

When 3D video content, such as V-PCC content, is encapsulated in ISOBMFF in the manner specified in the CD of 23090-10 Carriage of PC Data (N18606), its regions can be signalled in the two manners as suggested in the previous section.

In either case, volumetric V-PCC tracks and component tracks will each have their own corresponding DASH representations. A volumetric representation in DASH for a volumetric track will be a dependent representation, with attribute @dependencyId to list the id’s of all the complementary component representations for its component tracks.

When a volumetric track represents a region, when together with its component tracks, its corresponding volumetric representation represents a region for streaming, together with the complimentary representaitons for the component tracks.

If the region metadata of the regions are carried in timed metadata tracks, then the timed metadata representations for the timed metadata tracks will be associated with their volumetric representations, via attribute @associationId to list the id’s of all the volumetric representations of the tracks that the timed metadata track references to.

If the viewport metadata of a viewport are carried in a timed metadata track, then the timed metadata representation for the timed viewport metadata track will be associated with its volumetric representations, via attribute @associationId to list the id’s of all the volumetric representations of the tracks that the timed metadata track references to.

If the region metadata of the regions are carried in track grouping boxes of volumetric tracks, then it is proposed to use the following 3D extension of the (2D) SRD scheme to specify spatial relationships among the 3D regions (objects).

### 3D Spatial Relationship Description (SRD) scheme

The SRD scheme allows Media Presentation Description authors to express spatial relationships between 3D Spatial Objects. A Spatial Object is represented by either an Adaptation Set or a Sub-Representation. As an example, a spatial relationship may express that a 3D video represents a 3D spatial part of another full-size 3D video (e.g. a 3D region of interest, or a 3D tile).

The **SupplementalProperty** and/or **EssentialProperty** descriptors with @schemeIdUri equal to "urn:mpeg:dash:3dsrd:20xx" and "urn:mpeg:dash:3dsrd:dynamic:20xx" may be used to provide spatial relationship information associated to the containing Spatial Object. SRD information shall be contained exclusively in these two MPD elements (**AdaptationSet** and **SubRepresentation**).

To preserve the compatibility with legacy clients, MPD shall use **SupplementalProperty** and **EssentialProperty** in such a way that at least one Representation can be interpreted by legacy clients after discarding the element containing **EssentialProperty.**

NOTE Sub-Representation level SRDs can be used to represent Spatial Objects in one Representation such as HEVC tiling streams. In that case, SRD descriptors can be present at Adaptation Set as well as Sub-Representationlevels.

### 3D SRD @value syntax

#### General

The @value of the **SupplementalProperty** or **EssentialProperty** elements using the 3D SRD scheme is a comma-separated list of values for 3D SRD parameters.

When @value is not present, the 3D SRD does not express any spatial relationship information at all and can be ignored.

#### Common parameters

The source\_id parameter provides a unique identifier, within the Period, for the source of the content. It implicitly defines a coordinate system associated to this source. This coordinate system uses an arbitrary origin (0; 0; 0); the x-axis is oriented from left to right, the y-axis from top to bottom and the z-axis from near to far. All SRD sharing the same source\_id value have the same origin and axes orientations. Spatial relationships for Spatial Objects using SRD with different source\_id values are undefined.

For a given source\_id value, a reference space is defined, corresponding to the rectangular region encompassing the entire source content, whose near-top-left corner is at the origin of the coordinate system. The total\_width, total\_height and total\_depth values in a SRD provide the size of this reference space expressed in arbitrary units.

NOTE – There may be no Spatial Object in the MPD that covers the entire source of the content, e.g. when the entire source content is represented by two separate videos.

MPD authors can express, using the spatial\_set\_id parameter, that some Spatial Objects, within a given source\_id, have a particular spatial relationship. For instance, an MPD author may group all Adaptation Sets corresponding to tiles at a same resolution level. This way, the spatial\_set\_id parameter may be used by the DASH client to quickly select spatially related Spatial Objects. When two or more groups of full-frame video which consists of one or more Spatial Objects with the same total\_width, total\_height and total\_depth value, different values of spatial\_set\_id can be used to distinguish the groups of full-frame video.

NOTE – ISO/IEC 23009-3 gives concrete examples showing how to use the spatial\_set\_id.

#### Specific parameters for static spatial description

For expressing static description within the scope of the Period, the following Scheme Identifier is used "urn:mpeg:dash:3dsrd:20xx".

The centre\_x, centre\_y and centre\_z parameters (respectively rotation\_yaw, rotation\_pitch and rotation\_roll, and range\_width, range\_height and range\_depth) express 3D positions (respectively 3D rotations and sizes) of the associated Spatial Object in the 3D coordinate system associated to the source. The values of the object\_x, object\_y, object\_z, object\_width, total\_height, and total\_depth parameters are relative to the values of the total\_width, total\_height, and total\_depth parameters, as defined above. Positions (object\_x, object\_y, object\_z)) and sizes (object\_width, object\_height, object\_depth) of SRD sharing the same source\_id value may be compared after taking into account the size of the reference space, i.e. after the object\_x and object\_width values are divided by the total\_width value, the object\_y and object\_height values divided by the total\_height value and the object\_z and object\_depth values divided by the total\_depth value of their respective descriptors.

NOTE – Different total\_width, total\_height and total\_depth values could be used in different descriptors to provide positions and sizes information in different units.

**Table 15.1 — EssentialProperty**@value **and/or SupplementalProperty**@value **attributes for the static SRD scheme**

|  |  |  |
| --- | --- | --- |
| **EssentialProperty**@value **or SupplementalProperty**@value **parameter** | **Use** | **Description** |
| source\_id | M | non-negative integer in decimal representation providing the identifier for the Source of the content |
| object\_x | M | non-negative integer in decimal representation expressing the horizontal position of the near-top-left corner of the Spatial Object in arbitrary units |
| object\_y | M | non-negative integer in decimal representation expressing the vertical position of the near-top-left corner of the Spatial Object in arbitrary units |
| object\_z | M | non-negative integer in decimal representation expressing the depth position of the near-top-left corner of the Spatial Object in arbitrary units |
| object\_width | M | non-negative integer in decimal representation expressing the width of the Spatial Object in arbitrary units |
| object\_height | M | non-negative integer in decimal representation expressing the height of the Spatial Object in arbitrary units |
| object\_depth | M | non-negative integer in decimal representation expressing the depth of the Spatial Object in arbitrary units |
| object\_yaw | O | Optional, expressing the yaw rotation of the Spatial Object in the referenced coordinate system.  See its Semantics of rotation\_yaw in the metadata structure.  When the value object\_yaw is present, the values object\_pitch and object\_roll shall be present. |
| object\_pitch | O | Optional, expressing the pitch rotation of the Spatial Object in the referenced coordinate system.  See its Semantics of rotation\_pitch in the metadata structure.  When the value object\_pitch is present, the values object\_yaw and object\_roll shall be present. |
| object\_roll | O | Optional, expressing the roll rotation of the Spatial Object in the referenced coordinate system.  See its Semantics of rotation\_roll in the metadata structure.  When the value object\_roll is present, the values object\_yaw and object\_pitch shall be present. |
| total\_x | O | non-negative integer in decimal representation expressing the horizontal position of the near-top-left corner of the Spatial Source in arbitrary units  When the value total\_x is present, the values total\_y and total\_z shall be present. |
| total\_y | O | non-negative integer in decimal representation expressing the vertical position of the near-top-left corner of the Spatial Source in arbitrary units  When the value total\_y is present, the values total\_x and total\_z shall be present. |
| total\_z | O | non-negative integer in decimal representation expressing the depth position of the near-top-left corner of the Spatial Source in arbitrary units  When the value total\_z is present, the values total\_x and total\_y shall be present. |
| total\_width | O | optional non-negative integer in decimal representation expressing the width of the reference source space in arbitrary units.  At each Period and for a given source\_id value, the following rules apply:   * There shall be at least one descriptor providing a value for the total\_width parameter. * If two or more descriptors provide different total\_width values, all other descriptors shall explicitly provide the value of total\_width. * If the total\_width value is provided in only one descriptor, all other descriptors are assumed to use that total\_width value. * The value of total\_width shall be such that, for each descriptor using this value of total\_width, the sum of object\_x and object\_width is smaller or equal to total\_width.   When the value total\_width is present, the values total\_height and total\_depth shall be present. |
| total\_height | O | optional non-negative integer in decimal representation expressing the height of the reference source space in arbitrary units.  At each Period and for a given source\_id value, the following rules apply:   * There shall be at least one descriptor providing a value for the total\_height parameter. * If two or more descriptors provide different total\_height values, all other descriptors shall explicitly provide the value of total\_height. * If the total\_height value is provided in only one descriptor, all other descriptors are assumed to use that total\_height value. * The value of total\_height shall be such that, for each descriptor using this value of total\_height, the sum of object\_y and object\_height is smaller or equal to total\_height.   When the value total\_height is present, the values total\_width and total\_depth shall be present. |
| total\_depth | O | optional non-negative integer in decimal representation expressing the depth of the reference source space in arbitrary units.  At each Period and for a given source\_id value, the following rules apply:   * There shall be at least one descriptor providing a value for the total\_depth parameter. * If two or more descriptors provide different total\_ depth values, all other descriptors shall explicitly provide the value of total\_depth. * If the total\_depth value is provided in only one descriptor, all other descriptors are assumed to use that total\_depth value. * The value of total\_depth shall be such that, for each descriptor using this value of total\_depth, the sum of object\_z and object\_depth is smaller or equal to total\_depth.   When the value total\_depth is present, the values total\_width and total\_height shall be present. |
| total\_yaw | O | Optional, expressing the yaw rotation of the Spatial Source in the referenced coordinate system.  See its Semantics of rotation\_yaw in the metadata structure  When the value total\_yaw is present, the values total\_pitch and total\_yaw shall be present. |
| total\_pitch | O | Optional, expressing the pitch rotation of the Spatial Source in the referenced coordinate system.  See its Semantics of rotation\_pitch in the metadata structure  When the value total\_pitch is present, the values total\_yaw and total\_roll shall be present. |
| total\_roll | O | Optional, expressing the roll rotation of the Spatial Source in the referenced coordinate system.  See its Semantics of rotation\_roll in the metadata structure  When the value total\_roll is present, the values total\_yaw and total\_pitch shall be present. |
| spatial\_set\_id | O | optional non-negative integer in decimal representation providing an identifier for a group of Spatial Object.  When not present, the Spatial Object associated to this descriptor does not belong to any spatial set and no spatial set information is given.  When the value of spatial\_set\_id is present, the value of total\_width and total\_height shall be present. |
| **Legend:**  M=Mandatory, O=Optional | | |

#### Specific parameters for dynamic spatial description

For expressing dynamic description within the scope of the Period, the following Scheme Identifier is used "urn:mpeg:dash:3dsrd:dynamic:20xx".

In the case the Spatial Object moves within the reference space, the coordinates of the Spatial Object are time dependent and thus cannot be expressed as static values in a SRD. As a result, the SRD does not provide directly the coordinates and size as in the static case but instead specifies the @id attribute of the metadata Representation that provides the coordinates and size of the Spatial Object. This @id attribute value is signalled in the coordinate\_id parameter.

Examples of such scenarios include director's view, object tracking view, person tracking view in video conference applications, etc. For instance, the MPD author may offer two Spatial Objects, providing a wide angle view and a close-up view of the same scene of a sport event. The close-up view follows the action of the most popular athlete. But to ensure a satisfying Quality of Experience for the end-user, it is essential to describe the position of the close-up view with respect to the wide angle view at any point in time of the media content. This way, the end-user application can seamlessly switch from one video to another providing a smooth zooming in and out transition for the end-user.

**Table 15.2 — EssentialProperty**@value **and/or SupplementalProperty**@value **attributes for the dynamic SRD scheme**

|  |  |  |
| --- | --- | --- |
| **EssentialProperty**@value **or SupplementalProperty**@value **parameter** | **Use** | **Description** |
| source\_id | M | non-negative integer in decimal representation providing the identifier for the source of the content |
| coordinate\_id | M | specifies the @id attribute of the Representation that provides the 3D coordinates of the Spatial Object as timed metadata track according to ISO/IEC 23001-10. |
| spatial\_set\_id | O | optional non-negative integer in decimal representation providing an identifier for a group of Spatial Object.  When not present, the Spatial Object associated to this descriptor does not belong to any spatial set and no spatial set information is given. |
| **Legend:**  M=Mandatory, O=Optional | | |

The coordinates and size of a moving Spatial Object shall be provided by a Representation offering a 3D Cartesian coordinate track '6dtr' as defined in ISO/IEC 23001-10 TuC [33]. The @associationId attribute of this metadata Representation shall contain the value of the attribute @id of the Representation containing the moving Spatial Object. In addition, the @associationType attribute of this metadata Representation shall be set to 'cdsc'.

The following mapping between the '6dtr' sample parameters and the SRD parameters shall apply in order to determine the coordinates and size of a Spatial Object whose coordinates and size is provided as '6dtr' metadata Representation.

|  |  |  |
| --- | --- | --- |
| **Region and Source** | **3D Cartesian Coordinates Sample  (ISO/IEC 23001-10 TuC [33])** | **SRD parameters** |
| Region for Object | near\_top\_left\_x | object\_x |
| near\_top\_left\_y | object\_y |
| near\_top\_left\_z | object\_z |
| range\_width | object\_width |
| range\_height | object\_height |
| range\_depth | object\_depth |
| rotation\_yaw | object\_yaw |
| rotation\_pitch | object\_pitch |
| rotation\_roll | object\_roll |
| Source for Source (Total) | near\_top\_left\_x | total\_x |
| near\_top\_left\_y | total\_y |
| near\_top\_left\_z | total\_z |
| range\_width | total\_width |
| range\_height | total\_height |
| range\_depth | total\_depth |
| rotation\_yaw | total\_yaw |
| rotation\_pitch | total\_pitch |
| rotation\_roll | total\_roll |

# DASH-based Signaling of Recommended Viewport Information (m50654, m54310)

## Option 1 (m50654)

### General

Volumetric content distribution is gaining traction to deliver 6DoF immersive media experiences. Adaptive streaming based content distribution technologies such as MPEG DASH need to support point cloud content. Recommended viewport indication during streaming of volumetric content is desirable in order to deliver good quality of user experience. For each recommended viewport, the content provider basically optimally produced and encoded the point cloud content to be viewed from that particular viewport. The described viewport could for instance indicate the presence of a physical or virtual camera angle from which the content is produced. Moreover, depending on the viewport, for instance the content encoding may prioritize certain point cloud objects and/or encode certain point cloud regions or objects with higher quality compared to other regions or objects in the point cloud.

It should be understood that the MPD-based signaling proposed in this contribution is complementary to the recommended viewport signaling documented in TuC on Carriage of Point Cloud Data [33] that relies on the use of the timed metadata track to convey recommended viewport information. While the latter is useful to communicate recommended viewports that are dynamically varying in nature, the former (proposed) MPD framework is aimed at conveying a static set of recommended viewports (e.g., “VIP Tribune View”, “Marathon Tribune View”, etc. in a soccer stadium) for which the content provider has produced dedicated streams of point cloud content optimized for the corresponding viewports across which the DASH client can switch during the point cloud streaming based on user preferences.

### Proposed DASH Signaling of Recommended Viewports

A **SupplementalProperty** element with a @schemeIdUri attribute equal to "urn:mpeg:mpegI:pcc:2019:rv" may be defined for the recommended viewport with a descriptor in order to signal the recommended viewports of the point cloud content. For each recommended viewport, the content provider basically optimally produced and encoded the point cloud content to be viewed from that particular viewport. The described viewport could for instance indicate the presence of a physical or virtual camera angle from which the content is produced. Moreover, depending on the viewport, for instance the content encoding may prioritize certain point cloud objects and/or encode certain point cloud regions or objects with higher quality compared to other regions or objects in the point cloud.

For live presentations (with dynamic manifests or MPDs), potential changes in recommended viewports may be signaled via regular MPD updates. For other DASH settings, event-based MPD update mechanisms may also be used to signal potential updates on recommended viewports.

At most one recommended viewport indication descriptor may be present at adaptation set level. A recommended viewport indication descriptor is not expected to be present at MPD representation level, but it could be so.

Depending on user’s viewing devices, a viewport can be considered as a 3D spherical region or 2D planar region in the 3D space with 6 degrees of freedom (6 DoF), and hence 2D viewport or 3D viewport, respectively.

The Point Cloud recommended 3D viewport indication descriptor indicates that each Representation is produced for the 3D viewport with the sphere region as specified by syntax elements center\_azimuth, center\_elevation, center\_tilt, azimuth\_range, and elevation\_range to indicate the spherical coordinate system (to cover rotational movements of the viewport), plus syntax elements center\_x, center\_y and center\_z to indicate the x-y-z coordinates of the center point of the sphere that contains the viewport (to cover translational movements of the viewport).

The Point Cloud recommended 2D viewport indication descriptor indicates that each Representation is produced for the 2D viewport with the rectangular region as specified by syntax elements center\_azimuth, center\_elevation, center\_tilt, width\_range, and height\_range to indicate the rectangular coordinate system (to cover planar movements of the viewport), plus syntax elements center\_x, center\_y and center\_z to indicate the x-y-z coordinates of the center point of the plane that contains the viewport (to cover translational movements of the viewport).

Moreover, it may be possible to indicate recommended viewports via specific contextual information (e.g., the position of the ball, position of a star player, etc.) along with (or instead of) the coordinate-based description of the content coverage. One way to signal this information would be to define a string value associated with each adaptation set to carry the relevant contextual information.

At the beginning of the DASH media presentation, all of the recommended viewports for the point cloud content will be signaled to the DASH client as part of the MPD. Depending of the viewing preference of the user, the DASH client would determine which viewport is desired by the user, and fetch the DASH representations from the adaptation set corresponding to that viewport. During the presentation, the user may decide to switch the viewport (e.g., rather than view the game from the Seat A in the stadium, switch on to Seat B), and then the DASH client would switch to the adaptation set corresponding to the new viewport and fetch the corresponding DASH representations.

The MPD-based indication of recommended viewports for point cloud content includes elements and attributes as specified in Table 16.1.

**Table 16.1 – Semantics of elements and attributes of RV descriptor**

|  |  |  |  |
| --- | --- | --- | --- |
| **Elements and attributes for RV descriptor** | **Use** | **Data type** | **Description** |
| **rv** | 0..1 | pcc:RVType | Container element whose attributes and elements specify point cloud recommended viewport information. |
| **rv**@view\_idc\_presence\_flag | O | xs:boolean | Value 0 specifies that **rv.ViewportInfo**@view\_idc is not signalled. Value 1 specifies that **rv.ViewportInfo**@view\_idc is signalled and indicates the association of viewport positions with particular (left, right, or both) views or monoscopic content. When not present, **rv**@view\_idc\_presence\_flag is inferred to be equal to 0. |
| **rv**@default\_view\_idc | CM | Int | Value 0 indicates that all the viewport positions are monoscopic. Value 1 indicates that all the viewport positions are on the left view of a stereoscopic content. Value 2 indicates that all the viewport positions are on the right view of a stereoscopic content. Value 3 indicates that all the viewport positions are on both the left and right views. |
| **rv.ViewportInfo** | 1..255 | pcc:ViewportInfoType | Element whose attribute **rv.ViewportInfo**@view\_idc, when present, provides information about view(s) to which viewport specified by attributes **rv.ViewportInfo**@centre\_azimuth, **rv.ViewportInfo**@centre\_elevation, **rv.ViewportInfo**@centre\_tilt, **rv.ViewportInfo**@azimuth\_range, **rv.ViewportInfo**@elevation\_range, **rv.ViewportInfo**@width\_range, **rv.ViewportInfo**@height\_range, **rv.ViewportInfo**@center\_x, **rv.ViewportInfo**@center\_y, **rv.ViewportInfo**@center\_z, **rv.ViewportInfo**@context applies. |
| **rv.ViewportInfo**@viewport\_id | O | Int | Identifier corresponding to this particular recommended viewport |
| **rv.ViewportInfo**@view\_idc | O | Int | Value 1 indicates that the viewport position is on the left view of a stereoscopic content, value 2 indicates the viewport position is on the right view of a stereoscopic content, and value 3 indicates that the viewport position is on both the left and right views. Value 0 is reserved. |
| **rv.ViewportInfo**@center\_azimuth | O | Range is [-180\* 2−16, 180\* 2−16] | Specifies the azimuth of the centre point of the viewport position in units of 2−16 degrees relative to the global coordinate axes. |
| **rv.ViewportInfo**@center\_elevation | O | Range is [-90\* 2−16, 90\* 2−16] | Specifies the elevation of the centre point of the viewport position in units of 2−16 degrees relative to the global coordinate axes. |
| **rv.ViewportInfo**@center\_tilt | O | Range is [-180\* 2−16, 180\* 2−16] | Specifies the tilt angle of the viewport position, in units of 2−16 degrees, relative to the global coordinate axes. |
| **rv.ViewportInfo**@azimuth\_range | O | Int | Specifies the azimuth range of the viewport position through the centre point of the sphere region in units of 2−16 degrees. Only relevant for 3D viewports |
| **rv.ViewportInfo**@elevation\_range | O | Int | Specifies the elevation range of the viewport position through the centre point of the sphere region in units of 2−16 degrees. Only relevant for 3D viewports |
| **rv.ViewportInfo**@width\_range | O | Int | Specifies the width range of the rectangular region through its center point. Only relevant for 2D viewports |
| **rv.ViewportInfo**@height\_range | O | Int | Specifies the height range of the rectangular region through its center point. Only relevant for 2D viewports |
| **rv.ViewportInfo**@center\_x | O | Int | Integer in decimal representation expressing the x-coordinate of the center point of the sphere or plane containing the viewport in arbitrary units |
| **rv.ViewportInfo**@center\_y | O | Int | Integer in decimal representation expressing the y-coordinate of the center point of the sphere or plane containing the viewport in arbitrary units |
| **rv.ViewportInfo**@center\_z | O | Int | Integer in decimal representation expressing the z-coordinate of the center point of the sphere or plane containing the viewport in arbitrary units |
| **rv.ViewportInfo**@context | O | String | String describing the contextual information associated with the viewport, e.g., “VIP Tribune View”, “Marathon Tribune View”, etc. Context information may or may not be signalled in conjunction with the viewport coordinate information |

## Option 2 (m54310)

### General

Volumetric content distribution is gaining traction to deliver 6DoF immersive media experiences. Adaptive streaming based content distribution technologies such as MPEG DASH need to support V3C content. Recommended viewport indication during streaming of V3C content is desirable in order to deliver good quality of user experience. For each recommended viewport, the content provider basically optimally produced and encoded the V3C content to be viewed from that particular viewport. The described viewport could for instance indicate the presence of a physical or virtual camera angle from which the content is produced. Moreover, depending on the viewport, for instance the content encoding may prioritize certain point cloud objects and/or encode certain point cloud regions or objects with higher quality compared to other regions or objects in the point cloud.

During MPEG 130, a new timed metadata track was defined to signal camera information in V3C carriage format in ISO/IEC 23090-10 [1]-[2]. Using this information, the receiver may render the V3C content based on the signalled real or virtual camera positions and orientations.

In this contribution, we propose the following:

1. MPD-based signaling of recommended viewports
2. SAND-based signaling of user-selective viewports

The MPD-based signaling mechanism is complementary to the recommended viewport signaling adopted in ISO/IEC 23090-10 that relies on the use of the timed metadata track to convey recommended viewport information. While the latter is useful to communicate recommended viewports that are dynamically varying in nature, the former (proposed) MPD framework is aimed at conveying a static set of recommended viewports (e.g., “VIP Tribune View”, “Marathon Tribune View”, etc. in a soccer stadium) for which the content provider has produced dedicated streams of V3C content optimized for the corresponding viewports across which the DASH client can switch during the streaming based on user preferences.

Furthermore, to improve interactivity during streaming of V3C content, it is desirable to signal user’s viewport in the form of a client feedback message to the network so that this information could be: (i) shared with other users in the form of a recommended viewport, i.e., to realize social VR experiences in the form of “see what I see”, (ii) used for cloud/edge media processing, i.e., for transcoding or pre-rendering the media at an edge server. In this contribution, we provide a framework for client feedback signaling of viewport information in the form of SAND status messages. If the content / service provider chooses to support such SAND based capability to collect dynamic viewport information from users, it may also use SAND PER messages to signal recommended viewport information to users.

### Proposed DASH Signaling of Recommended Viewports

A **SupplementalProperty** element with a @schemeIdUri attribute equal to "urn:mpeg:mpegI:v3c:2020:rv" is defined for the Recommended Viewport (RV) descriptor in order to signal the recommended viewports of the V3C content, such that the content provider recommends V3C content to be rendered based on the set of signaled virtual camera position and orientation parameters. For each RV, the content provider may further have optimally produced and encoded the V3C content to be viewed from that particular viewport. The described viewport could for instance indicate the presence of a physical or virtual camera angle from which the content is produced. Moreover, depending on the viewport, for instance the content encoding may prioritize certain point cloud objects and/or encode certain spatial regions or objects with higher quality compared to other regions or objects in the point cloud.

If present, at most one RV descriptor shall be present in each adaptation set for the single-track DASH mode. In case of the multi-track DASH mode, the RV descriptor, if present, shall only be placed in the main adaptation set. No other RV descriptor shall be present at the MPD representation level or any other level in both single-track and multi-track DASH modes.

The RV descriptor indicates that each Representation in the Adaptation Set (for the multi-track case, this includes the Representations in the main Adaptation set and other related Adaptation Sets for the corresponding V3C components) is recommended to be rendered based on the set of virtual camera position and orientation parameters specified by syntax elements camera\_id, camera\_pos\_x, camera\_pos\_y, camera\_pos\_z, camera\_quat\_x, camera\_quat\_y, and camera\_quat\_z.

Moreover, it may be possible to indicate RVs via specific contextual information (e.g., “VIP Tribune View”, “Marathon Tribune View”, etc.) along with the coordinate-based description of the viewport. This information is signaled by defining a string value context associated with each adaptation set to carry the relevant contextual information. Type of the recommended viewport as listed in Table G.2.3.2.1 may also be signaled via viewport\_type.

For live presentations with dynamic MPDs, potential changes in RVs may be signaled via regular MPD updates. For other DASH settings, event-based MPD update mechanisms may also be used to signal potential updates on RVs.

If the RV is associated with a timed metadata Representation carrying a timed metadata track with sample entry type '6cam', the position and orientation of the RV are dynamic. Otherwise, the position and orientation of the RV is static. In the former case, timed-metadata track for viewport signalling shall be carried in a separate Adaptation Set and associated (linked) with the main V3C track using the @associationId attribute, defined in ISO/IEC 23009-1 ‎[DASH], with a @associationType value that includes the 4CC ‘6cam’ for the corresponding Adaptation Set or Representation.

At the beginning of the DASH media presentation, all of the RVs for the V3C content adaptation sets will be signaled to the DASH client as part of the MPD. Depending of the viewing preference of the user, the DASH client would determine which viewport is desired by the user, and fetch the DASH representations from the adaptation set corresponding to that RV. During the presentation, the user may decide to switch the viewport (e.g., rather than view the game from the Seat A in the stadium, switch on to Seat B), and then the DASH client would switch to the adaptation set corresponding to the new RV and fetch the corresponding DASH representations.

The MPD-based indication of recommended viewports for V3C content includes elements and attributes as specified in Table 1.

**Table 1 – Semantics of elements and attributes of RV descriptor**

|  |  |  |  |
| --- | --- | --- | --- |
| **Elements and attributes for RV descriptor** | **Use** | **Data type** | **Description** |
| @camera\_id | O | xs:string | Specifies the identifier of the camera to which the viewport belongs. |
| **ViewportInfo** | 1 | v3c:ViewportInfoType | Container element whose sub-elements and attributes provide information about the viewport. |
| **ViewportInfo**@camera\_pos\_x | M | Int | Indicates the x coordinate of the location of the camera in meters in the global reference coordinate system. The values shall be in units of 2−16 meters.  If position of the RV is dynamic, this attribute specifies the initial viewport X position for this RV. Otherwise, this attribute specifies the static viewport X position. |
| **ViewportInfo**@camera\_pos\_y | M | Int | Indicates the y coordinate of the location of the camera in meters in the global reference coordinate system. The values shall be in units of 2−16 meters.  If position of the RV is dynamic, this attribute specifies the initial viewport Y position for this RV. Otherwise, this attribute specifies the static viewport Y position. |
| **ViewportInfo**@camera\_pos\_z | M | Int | Indicates the z coordinate of the location of the camera in meters in the global reference coordinate system. The values shall be in units of 2−16 meters.  If position of the RV is dynamic, this attribute specifies the initial viewport Z position for this RV. Otherwise, this attribute specifies the static viewport Z position. |
| **ViewportInfo**@camera\_quat\_x | M | Int | Indicates the x component of the orientation of the camera using the quaternion representation. The values shall be a floating-point value in the range of −1 to 1, inclusive. |
| **ViewportInfo**@camera\_quat\_y | M | Int | Indicates the y component of the orientation of the camera using the quaternion representation. The values shall be a floating-point value in the range of −1 to 1, inclusive. |
| **ViewportInfo**@camera\_quat\_z | M | Int | Indicates the z component of the orientation of the camera using the quaternion representation. The values shall be a floating-point value in the range of −1 to 1, inclusive. |
| **ViewPortInfo**@initialViewport | O | xs:Boolean | If equal to true this attribute specifies that this RV is the initial viewport that should be used out of all the RVs in the current Period.  If equal to false this attribute specifies that this RV is not the initial viewport in the current Period.  In a Period at most one RV shall have **ViewPointInfo**@initialViewport equal to true. |
| **ViewportInfo**@context | O | xs:string | String describing the human-readable contextual information associated with the viewport, e.g., “VIP Tribune View”, “Marathon Tribune View”, etc. Context information may or may not be signalled in conjunction with the viewport information. |
| **ViewportInfo**@viewport\_type | O | Int | Type of the recommended viewport as listed in Table G.2.3.2.1. |

The data types for various elements and attributes shall be as defined in the XML schema. An XML schema for this shall be as shown below. The schema shall be represented in an XML schema that has namespace urn:mpeg:mpegI:v3c:2020 and is specified as follows:

<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"

targetNamespace="urn:mpeg:mpegI:v3c:2020"

xmlns:omaf="urn:mpeg:mpegI:v3c:2020"

elementFormDefault="qualified">

<xs:element name="ViewportInfo" type="v3c:ViewportInfoType"/>

<xs:complexType name="ViewportInfoType">

<xs:attribute name="camera\_pos\_x" type="xs:int" use="required" default="0"/>

<xs:attribute name="camera\_pos\_y" type="xs:int" use="required" default="0"/>

<xs:attribute name="camera\_pos\_z" type="xs:int" use="required" default="0"/>

<xs:attribute name="camera\_quat\_x" type="xs:int" use="required" default="0"/>

<xs:attribute name="camera\_quat\_y" type="xs:int" use="required" default="0"/>

<xs:attribute name="camera\_quat\_z" type="xs:int" use="required" default="0"/>

<xs:attribute name="initialViewport" type="xs:boolean" use="optional"/>

<xs:attribute name="context" type="xs:string" use="optional"/>

<xs:attribute name="viewport\_type" type="xs:int" use="optional" default="0"/>

<xs:anyAttribute processContents="skip"/>

</xs:complexType>

</xs:schema>

# Dynamic Signaling of User-Selected and Recommended Viewports for PC Data (m50655, m54310)

## Option 1 (m50655)

### General

Volumetric content distribution is gaining traction to deliver 6DoF immersive media experiences. Adaptive streaming based content distribution technologies such as MPEG DASH need to support point cloud content. Viewport indication during streaming of volumetric content is essential in order to deliver good quality of user experience. In particular, to improve interactivity during streaming of point cloud content, it is desirable to signal user’s viewport in the form of a client feedback message to the network so that this information could be: (i) shared with other users in the form of a recommended viewport, i.e., to realize social VR experiences in the form of “see what I see”, (ii) used for cloud/edge media processing, i.e., for transcoding or pre-rendering the media at an edge server.

In this contribution, we provide a framework for client feedback signaling of viewport information in the form of SAND status messages. If the content / service provider chooses to support such SAND based capability to collect dynamic viewport information from users, it may also use SAND PER messages to signal recommended viewport information to users. Such signaling is also described in this contribution, as an alternative solution to what has been documented in TuC on Carriage of Point Cloud Data [33] that relies on the use of the timed metadata track to convey recommended viewport information.

### Proposed Server-Client Signaling Interface

Server and Network Assisted DASH (SAND) introduces messages between DASH clients and network elements or between various network elements for the purpose to improve efficiency of streaming sessions by providing information about real-time operational characteristics of networks, servers, proxies, caches as well as DASH client's performance and status. SAND constitutes Part 5 of the MPEG DASH specifications, namely ISO/IEC 23009-5 [35]. Within this architecture, the following categories of messages, called SAND messages, are exchanged:

* Parameters Enhancing Reception (PER) messages that are sent from DANEs to DASH clients,
* Status messages that are sent from DASH clients to DANEs.

**Client Feedback Signalling of User-Selected Viewports over the Point Cloud**

A new SAND status message ‘SelectedViewport’ can be specified in order to signal the user-selected viewports of the point cloud content at a specific time.

Depending on user’s viewing devices, a viewport can be considered as a 3D spherical region or 2D planar region in the 3D space with 6 degrees of freedom (6 DoF), and hence 2D viewport or 3D viewport, respectively. Each signaled user-selected viewport may be identified by the viewport\_id.

The Point Cloud user-selected 3D viewport indication signals the viewport with the sphere region as specified by syntax elements center\_azimuth, center\_elevation, center\_tilt, azimuth\_range, and elevation\_range to indicate the spherical coordinate system (to cover rotational movements), plus syntax elements center\_x, center\_y and center\_z to indicate the x-y-z coordinates of the center point of the sphere that contains the viewport (to cover translational movements).

The Point Cloud user-selected 2D viewport indication signals the viewport with the rectangular region as specified by syntax elements center\_azimuth, center\_elevation, center\_tilt, width\_range, and height\_range to indicate the rectangular coordinate system (to cover planar movements), plus syntax elements center\_x, center\_y and center\_z to indicate the x-y-z coordinates of the center point of the plane that contains the viewport (to cover translational movements).

Moreover, it may be possible to indicate in the signaled viewport specific contextual information (e.g., the position of the ball, position of a star player, etc.) along with (or instead of) the coordinate-based description of the content coverage. One way to signal this information would be to define a string value to carry the relevant contextual information. Another option may be to signal an object\_ID value, which refers to the specific point cloud object from which the viewport may be derived. The context and Object ID information may be optionally signalled in conjunction with the viewport coordinate information in order to provide more specifics about the viewport.

* + - 1. **Source and destination**

|  |
| --- |
| Type: : Status  Sender : DASH client  Receiver : DANE |

* + - 1. **Data representation**

**Table 17.1 - SelectedViewport parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parameter** | | **Type** | **Cardinality** | **Description** |
| SelectedViewport | | Object | 1 |  |
|  | Timestamp | date-time | 1 | Wall-clock time corresponding to the signaled viewport values |
|  | viewport\_id | Int | 1 | Identifier associated with the signaled viewport |
|  | center\_azimuth | Int: Range is [-180\* 2−16, 180\* 2−16] | 1 | Specifies the azimuth of the centre point of the viewport position in units of 2−16 degrees relative to the global coordinate axes. When not present, center\_azimuth is inferred to be equal to 0. |
|  | center\_elevation | Int: Range is [-90\* 2−16, 90\* 2−16] | 1 | Specifies the elevation of the centre point of the viewport position in units of 2−16 degrees relative to the global coordinate axes. When not present, center\_elevation is inferred to be equal to 0. |
|  | center\_tilt | Int: Range is [-180\* 2−16, 180\* 2−16] | 1 | Specifies the tilt angle of the viewport position, in units of 2−16 degrees, relative to the global coordinate axes. When not present, center\_tilt is inferred to be equal to 0. |
|  | azimuth\_range | Int | 0,1 | Specifies the azimuth range of the sphere region through the centre point of the sphere region in units of 2−16 degrees. When not present, azimuth\_range is inferred to be equal to 360 \* 216. Only relevant for 3D viewports |
|  | elevation\_range | Int | 0,1 | Specifies the elevation range of the sphere region through the centre point of the sphere region in units of 2−16 degrees. When not present, elevation\_range is inferred to be equal to 180 \* 216. Only relevant for 3D viewports |
|  | width\_range | Int | 0,1 | Specifies the width range of the rectangular region through its center point. Only relevant for 2D viewports |
|  | height\_range | Int | 0,1 | Specifies the height range of the rectangular region through its center point. Only relevant for 2D viewports |
|  | center\_x | Int | 1 | Integer in decimal representation expressing the x-coordinate of the center point of the sphere or plane containing the viewport in arbitrary units |
|  | center\_y | Int | 1 | Integer in decimal representation expressing the y-coordinate of the center point of the sphere or plane containing the viewport in arbitrary units |
|  | center\_z | Int | 1 | Integer in decimal representation expressing the z-coordinate of the center point of the sphere or plane containing the viewport in arbitrary units |
|  | object\_id | Int | 0,1 | Integer expressing the object ID associated with the viewport. Object ID information may or may not be signalled in conjunction with the viewport coordinate information. |
|  | context | String | 0,1 | String describing the contextual information associated with the viewport, e.g., “ball”, “player X”, etc. Context information may or may not be signalled in conjunction with the viewport coordinate information |

HTTP may be used as the transport protocol for carrying the above SAND status message. This does not preclude that other additional transport protocols could also be implemented, e.g., the use of the WebSocket protocol (IETF RFC 6455), as specified in clause 10 of ISO/IEC 23009-5. In particular, the newly defined SAND message could be carried as part of the HTTP header when requesting DASH segments using HTTP GET (as described in clause 8.2.3 of ISO/IEC 23009-5). Alternatively HTTP POST message could be used, and in this case the SAND message may be carried in the body of the HTTP POST message (as described in clause 8.2.2 of ISO/IEC 23009-5). In case of WebSocket, the newly defined SAND message may be transported over a WebSocket connection from the client to the network in a push-based manner.

**Server Signaling of Recommended Viewports over the Point Cloud to the Streaming Client**

A new SAND PER message ‘RecommendedViewport’ can be specified in order to signal the network-recommended viewports of the point cloud content at a specific time. Each recommended viewport may be identified by the viewport\_id and its type may be indicated using viewport\_type as per ISO/IEC 23090-2.

The Point Cloud recommended 3D viewport indication signals the viewport with the sphere region as specified by syntax elements center\_azimuth, center\_elevation, center\_tilt, azimuth\_range, and elevation\_range to indicate the spherical coordinate system (to cover rotational movements), plus syntax elements center\_x, center\_y and center\_z to indicate the x-y-z coordinates of the center point of the sphere that contains the viewport (to cover translational movements).

The Point Cloud recommended 2D viewport indication signals the viewport with the rectangular region as specified by syntax elements center\_azimuth, center\_elevation, center\_tilt, width\_range, and height\_range to indicate the rectangular coordinate system (to cover planar movements), plus syntax elements center\_x, center\_y and center\_z to indicate the x-y-z coordinates of the center point of the plane that contains the viewport (to cover translational movements).

Moreover, it may be possible to indicate in the signaled viewport specific contextual information (e.g., the position of the ball, position of a star player, etc.) along with (or instead of) the coordinate-based description of the content coverage. One way to signal this information would be to define a string value to carry the relevant contextual information. Another option may be to signal an object\_ID value, which refers to the specific point cloud object from which the viewport may be derived. Object ID and context information may be optionally signalled in conjunction with the viewport coordinate information in order to provide more specifics about the viewport.

* + - 1. **Source and destination**

|  |
| --- |
| Type: : PER  Sender : DANE  Receiver : DASH client |

* + - 1. **Data representation**

**Table 17.2 - RecommendedViewport parameters**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parameter** | | **Type** | **Cardinality** | **Description** |
| RecommendedViewport | | Object | 1 |  |
|  | Timestamp | date-time | 1 | Wall-clock time corresponding to the signaled viewport values |
|  | viewport\_id | Int | 1 | Identifier associated with the signaled viewport |
|  | viewport\_type | Int | 1 | Type of the viewport: Value 0 indicates recommended viewport per the director's cut, i.e., a viewport suggested according to the creative intent of the content author or content provider. Value 1 indicates a recommended viewport selected based on measurements of viewing statistics. Value 2 indicates the viewport selected by another user. |
|  | center\_azimuth | Int: Range is [-180\* 2−16, 180\* 2−16] | 1 | Specifies the azimuth of the centre point of the viewport position in units of 2−16 degrees relative to the global coordinate axes. When not present, center\_azimuth is inferred to be equal to 0. |
|  | center\_elevation | Int: Range is [-90\* 2−16, 90\* 2−16] | 1 | Specifies the elevation of the centre point of the viewport position in units of 2−16 degrees relative to the global coordinate axes. When not present, center\_elevation is inferred to be equal to 0. |
|  | center\_tilt | Int: Range is [-180\* 2−16, 180\* 2−16] | 1 | Specifies the tilt angle of the viewport position, in units of 2−16 degrees, relative to the global coordinate axes. When not present, center\_tilt is inferred to be equal to 0. |
|  | azimuth\_range | Int | 1 | Specifies the azimuth range of the sphere region through the centre point of the sphere region in units of 2−16 degrees. When not present, azimuth\_range is inferred to be equal to 360 \* 216. Only relevant for 3D viewports |
|  | elevation\_range | Int | 1 | Specifies the elevation range of the sphere region through the centre point of the sphere region in units of 2−16 degrees. When not present, elevation\_range is inferred to be equal to 180 \* 216. Only relevant for 3D viewports |
|  | width\_range | Int | 0,1 | Specifies the width range of the rectangular region through its center point. Only relevant for 2D viewports |
|  | height\_range | Int | 0,1 | Specifies the height range of the rectangular region through its center point. Only relevant for 2D viewports |
|  | center\_x | Int | 1 | Integer in decimal representation expressing the x-coordinate of the center point of the sphere or plane containing the viewport in arbitrary units |
|  | center\_y | Int | 1 | Integer in decimal representation expressing the y-coordinate of the center point of the sphere or plane containing the viewport in arbitrary units |
|  | center\_z | Int | 1 | Integer in decimal representation expressing the z-coordinate of the center point of the sphere or plane containing the viewport in arbitrary units |
|  | object\_id | Int | 0,1 | Integer expressing the object ID associated with the viewport. Object ID information may or may not be signalled in conjunction with the viewport coordinate information. |
|  | context | String | 0,1 | String describing the contextual information associated with the viewport, e.g., “ball”, “player X”, etc. Context information may or may not be signalled in conjunction with the viewport coordinate information |

HTTP may be used as the transport protocol for carrying the above SAND PER message. This does not preclude that other additional transport protocols could also be implemented, e.g., the use of the WebSocket protocol (IETF RFC 6455), as specified in clause 10 of ISO/IEC 23009-5. In particular, DASH client may explicitly request the newly defined SAND message from the server (DANE) using an HTTP GET message (as described in clause 8.3 of ISO/IEC 23009-5). In case of WebSocket, the newly defined SAND message may be transported over a WebSocket connection from the client to the network in a push-based manner without requiring the client to continuously poll the DANE to request the updated recommended viewport information.

Dynamic sharing of the user-selected viewport may be considered as a new kind of recommended viewport, which may for instance be identified by extending the viewport\_type indicator in ISO/IEC 23090-2 as follows:

**Table 17.3 – Recommended viewport type**

|  |  |
| --- | --- |
| **Value** | **Description** |
| 0 | A recommended viewport per the director's cut, i.e., a viewport suggested according to the creative intent of the content author or content provider |
| 1 | A recommended viewport selected based on measurements of viewing statistics |
| 2 | A recommended viewport based on the selected viewport of another user |
| 3..239 | Reserved |
| 240..255 | Unspecified (for use by applications or external specifications) |

## Option 2 (m54310)

SAND status message ‘SelectedViewport’ is specified to signal the user-selected viewports of the V3C content at a specific time.

The user-selected viewport indication signals the set of virtual camera position and orientation parameters specified by syntax elements camera\_id, camera\_pos\_x, camera\_pos\_y, camera\_pos\_z, camera\_quat\_x, camera\_quat\_y, and camera\_quat\_z. Moreover, context indicates in the signaled viewport specific contextual information (e.g., the position of the ball, position of a star player, etc.) along with the coordinate-based description of the viewport.

#### Source and destination

|  |
| --- |
| Type: : Status  Sender : DASH client  Receiver : DANE |

#### Data representation

Table 1 - SelectedViewport parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parameter** | | **Type** | **Cardinality** | **Description** |
| SelectedViewport | | Object | 1 |  |
|  | Timestamp | date-time | 1 | Wall-clock time corresponding to the signaled viewport values |
|  | camera\_id | String | 1 | Specifies the identifier of the camera to which the viewport belongs. |
|  | camera\_pos\_x | Integer | 1 | Indicates the x coordinate of the location of the camera in meters in the global reference coordinate system. The values shall be in units of 2−16 meters. |
|  | camera\_pos\_y | Integer | 1 | Indicates the y coordinate of the location of the camera in meters in the global reference coordinate system. The values shall be in units of 2−16 meters. |
|  | camera\_pos\_z | Integer | 1 | Indicates the z coordinate of the location of the camera in meters in the global reference coordinate system. The values shall be in units of 2−16 meters. |
|  | camera\_quat\_x | Integer | 1 | Indicates the x component of the orientation of the camera using the quaternion representation. The values shall be a floating-point value in the range of −1 to 1, inclusive. |
|  | camera\_quat\_y | Integer | 1 | Indicates the y component of the orientation of the camera using the quaternion representation. The values shall be a floating-point value in the range of −1 to 1, inclusive. |
|  | camera\_quat\_z | Integer | 1 | Indicates the z component of the orientation of the camera using the quaternion representation. The values shall be a floating-point value in the range of −1 to 1, inclusive. |
|  | context | String | 0,1 | String describing the human-readable contextual information associated with the viewport, e.g., “ball”, “player X”, etc. Context information may or may not be signalled in conjunction with the viewport information. |

HTTP may be used as the transport protocol for carrying the above SAND status message. This does not preclude that other additional transport protocols could also be implemented, e.g., the use of the WebSocket protocol (IETF RFC 6455), as specified in clause 10 of ISO/IEC 23009-5. In particular, the newly defined SAND message could be carried as part of the HTTP header when requesting DASH segments using HTTP GET (as described in clause 8.2.3 of ISO/IEC 23009-5). Alternatively, HTTP POST message could be used, and in this case the SAND message may be carried in the body of the HTTP POST message (as described in clause 8.2.2 of ISO/IEC 23009-5). In case of WebSocket, the newly defined SAND message may be transported over a WebSocket connection from the client to the network in a push-based manner.

# Encapsulation and Signalling of V-PCC data in MMT (m50769, m52792)

## General

V-PCC encoded content can be delivered over networks using MMT. In this contribution, methods for the encapsulation of V-PCC data for MMT streaming and the necessary MMT signalling messages for supporting the delivery of V-PCC data over MMT is proposed.

For V-PCC contents contained inside the ISOBMFF using multi-tracks, each track is proposed to be encapsulated into a separate asset, which is then packetized into MMTP packets in the usual manner. In order for the server and client to be able to identify a group of multiple assets to a certain V-PCC content (and their variations), a V-PCC defined application message is also proposed. It is also proposed to include a section entitled “Encapsulation and signaling in MMT” into the CD of 23090-10, and to include the text, syntax and semantics below into that section.

## Proposal

In order to support the delivery of V-PCC contents using MMT, each track inside the multi-track ISOBMFF V-PCC container will be encapsulated into a separate Asset, the number Assets equaling the number of tracks inside the original multitrack ISOBMFF V-PCC container. The grouping of multiple Assets which correspond to a single V-PCC content is signalled using the defined application message below. Alternative component tracks can also be exposed using the various message types to enable efficient server and client selection decisions (without the need for first parsing the ISOBMFF file inside the MMTP packet).

MMT defines application-specific signalling messages which allow for the delivery of application-specific information. For the purpose of streaming V-PCC encoded data using MMT, a V-PCC specific signalling message is defined as below. The V-PCC specific signalling message shall have an application identifier with urn of value of “urn:mpeg:mmt:app:vpcc:2019”

In the specified V-PCC signalling message, the following set of application message types are defined:

* VPCCAssetGroupMessage: the sending entity sends this message to inform the client about the set of assets available at the server, and a list of those assets which are being streamed to the receiving entity.
* VPCCCSelectionMessageFeedback: the client uses this feedback message to request the set of assets to be streamed by the sending entity to the receiving entity.
* VPCCViewChangeFeedback: the client uses this feedback message to send an indication of the user’s current viewing space to the server.

The VPCCAssetGroupMessage is mandatory when sending V-PCC encoded content via MMT, providing the client with a list of V-PCC data type assets available at the server, as well as informing the client about which of these assets are currently being streamed to the receiving entity. From this list, the client is able to request a unique sub-set of these V-PCC data type assets using the VPCCCSelectionFeedback message.

For the view dependent delivery of V-PCC contents through MMT, the client shall use the VPCCViewChangeFeedback message to send its current viewing space (frustum) information to the server, after which the server can select and delivery the assets corresponding to that viewing space to the client. The VPCCAssetGroupMessage is also updated and sent to the client accordingly.

The list of defined application message types is provided in Table 1.

**Table 18.1.** V-PCC application message types

|  |  |
| --- | --- |
| **Application Message Type** | **Application Message Name** |
| 0x01 | VPCCAssetGroupMessage |
| 0x02 | VPCCCSelectionFeedback |
| 0x03 | VPCCViewChangeFeedback |
| 0x04-0xFF | Reserved |

[Ed(EA) In MPEG 129, m52792 is accepted into the TuC. It proposes to add a flag to the VPCCAssetGroupMessage to inform the client of the state of each one object, indicating if all the essential data is ready for rendering. The specification text which is related to this contribution is included as yellow highlighted text in the reated sections below.]

## MMT Signaling

### VPCCAssetGroupMessage

#### Syntax

**Table 18.2. VPCCAssetGroupMessage**

|  |  |  |  |
| --- | --- | --- | --- |
| **Syntax** | **Values** | **No. of bits** | **Mnemonic** |
| Application (){  ***message\_id***  ***version***  ***length***  application\_identifier()  if (application\_identifier == “urn:mpeg:mmt:app:vpcc:2019”)  {  app\_message\_type  if (app\_message\_type == 0x01) {  ***num\_vpcc\_asset\_groups***  for (i=0; i<N1; i++) {  ***asset\_group\_id***  ***num\_assets***  ***start\_time***  ***pending\_flag***  for (j=0; j<N2; j++) {  ***data\_type***  ***reserved***  ***is\_pcm***  ***is\_single\_layer***  ***state\_flag***  ***sending\_time\_flag***  if(sending\_time\_flag){  ***sending\_time***  }  ***asset\_id***  }  }  }  }  } | **N1**  **N2**  **“1111”** | 16  8  16  8  8  16  16  32  8  4  1  1  1  1  32  var |  |
|  |
| uimsbf |
|  |

#### Semantics

message\_id indicates the identifier of the V-PCC application message.

version indicates the version of the V-PCC application message.

length indicates the length of the V-PCC application message in bytes, counting from the beginning of the next field to the last byte of the message. The value of this field shall not be equal to 0.

application\_identifier – indicates the application identifier as a urn that uniquely identifies the application to consume the contents of this message.

app\_message\_type – defines an application-specific message type provided in Table 1. The length of this field is 8 bits.

num\_vpcc\_asset\_groups – indicates the number of V-PCC asset groups, where each group contains the component assets associated with a V-PCC content.

asset\_group\_id – indicates the identifier of the asset group associated with a V-PCC content.

num\_assets – indicates the number of assets within this asset group which are associated with a V-PCC content.

start\_time – indicates the presentation time of the V-PCC content from which the state of the assets listed in this message are applicable.

pending\_flag – indicates if all the data compoenets are ready for rendering for each group. When set to “1”, it indicates that the data is ready, otherwise the flag is “0”.

data\_type – indicates the data type of point cloud data present in the asset stream.

**Table 18.3.** Values for data\_type

|  |  |
| --- | --- |
| **Value** | **Description** |
| 0x00 | Reserved |
| 0x01 | V-PCC metadata |
| 0x02 | Occupancy data |
| 0x03 | Geometry data |
| 0x04 | Attribute data |
| 0x05-0xFF | Reserved |

is\_pcm – a flag indicating whether the V-PCC component information present in the asset stream is for a missing points patch. When set to “1”, the V-PCC component information is for a missing points patch, otherwise this flag is set to “0”.

is\_single\_layer – a flag indicating whether the V-PCC component information present in the asset stream carries one layer or all the layers of the V-PCC content. When set to “1”, this indicates that only a single layer is present. When set to “0”, this indicates that layer for the component are present.

state\_flag – a flag indicating the delivery state of the asset. When set to “1”, this indicates that the sending entity is actively sending the asset to the receiving entity. When set to “0”, this indicates that the sending entity is not actively sending the asset to the receiving entity.

sending\_time\_flag – a flag indicating the presence of sending\_time for the first MMTP packet containing the first MPU of the asset stream. The default value is “0”.

sending\_time – indicates the sending time for the first MMTP packet containing the first MPU of the asset stream. Using this information, the client can prepare a new packet processing pipeline for the new asset stream.

asset\_id – provides the asset identifier of the asset.

### VPCCCSelectionFeedback

#### Syntax

**Table 18.4 VPCCCSelectionFeedback**

|  |  |  |  |
| --- | --- | --- | --- |
| **Syntax** | **Values** | **No. of bits** | **Mnemonic** |
| Application (){  ***message\_id***  ***version***  ***length***  application\_identifier()  if (application\_identifier == “urn:mpeg:mmt:app:vpcc:2019”)  {  app\_message\_type  if (app\_message\_type == 0x02) {  ***num\_selected\_asset\_groups***  for (i=0; i<N1; i++) {  ***asset\_group\_id***  ***reserved***  ***switching\_mode***  ***num\_asset\_id***  if(switching\_mode = 0x1 || 0x2){  for (j=0; j<N2; j++) {  ***asset\_id***  }  }  }  }  }  } | **N1**  **“1111”**  **N2** | 16  8  16  8  8  16  4  4  16  var |  |
|  |
| uimsbf |
|  |

#### Semantics

message\_id indicates the identifier of the V-PCC application message.

version indicates the version of the V-PCC application message.

length indicates the length of the V-PCC application message in bytes, counting from the beginning of the next field to the last byte of the message. The value of this field shall not be equal to 0.

application\_identifier – indicates the application identifier as a urn that uniquely identifies the application to consume the contents of this message.

app\_message\_type – defines an application-specific message type provided in Table 1. The length of this field is 8 bits.

num\_selected\_asset\_groups – indicates the number of asset groups for which there is an associated state change request by the receiving entity.

asset\_group\_id – indicates the identifier of the asset group associated with a V-PCC content.

switching\_mode – indicates the switching mode used for the selection of assets as requested by the receiving entity.

**Table 18.5.** Switching modes and definition of switching\_mode

|  |  |  |
| --- | --- | --- |
| **Value** | **Switching mode** | **Definitinon of switching mode** |
| 0x1 | Refresh | For each asset listed as specified by its asset\_id, its state\_flag will be set to “1”, and the state\_flag for all other non-listed assets of the same data\_type will be set to “0”.  The states for assets of other non listed data types will remain unchanged. |
| 0x2 | Toggle | For each asset listed as specified by its asset\_id, its state\_flag will be changed (to “1”, if originally “0”, to “0” if originally “1”).  The states for all non listed assets will remain unchanged |
| 0x3 | Send all | For the specified asset group, all associated assets within the group have their state\_flag set to “1”. |
| 0x4~0xF | Reserved | Reserved |

num\_group\_id – indicates the number of asset ids which are signalled for the state change according to the switching mode specified.

asset\_id – indicates the identifier for the asset for the state change according to the switching mode specified.

### VPCCViewChangeFeedback

#### Syntax

**Table 18.6. VPCCViewChangeFeedback**

|  |  |  |  |
| --- | --- | --- | --- |
| **Syntax** | **Values** | **No. of bits** | **Mnemonic** |
| Application (){  ***message\_id***  ***version***  ***length***  application\_identifier()  if (application\_identifier == “urn:mpeg:mmt:app:vpcc:2019”)  {  app\_message\_type  if (app\_message\_type == 0x03) {  ***near\_point\_x***  ***near\_point\_y***  ***near\_point\_z***  ***vector\_x***  ***vector\_y***  ***vector\_z***  ***centre\_tilt***  ***near\_azimuth\_range***  ***near\_elevation\_range***  ***far\_azimuth\_range***  ***far\_elevation\_range***  }  }  } |  | 16  8  16  8  32  32  32  32  32  32  32  32  32  32  32 |  |
|  |
| uimsbf |
|  |

#### Semantics

message\_id indicates the identifier of the V-PCC application message.

version indicates the version of the V-PCC application message.

length indicates the length of the V-PCC application message in bytes, counting from the beginning of the next field to the last byte of the message. The value of this field shall not be equal to 0.

application\_identifier – indicates the application identifier as a urn that uniquely identifies the application to consume the contents of this message.

app\_message\_type – defines an application-specific message type provided in Table 1. The length of this field is 8 bits.

near\_point\_x – indicates the x coordinate value of the near point of the client’s viewing frustum. The near point corresponds to the centre point of the near plane of the viewing frustum.

near\_point\_y – indicates the y coordinate value of the near point of the client’s viewing frustum. The near point corresponds to the centre point of the near plane of the viewing frustum.

near\_point\_z – indicates the z coordinate value of the near point of the client’s viewing frustum. The near point corresponds to the centre point of the near plane of the viewing frustum.

vector\_x – indicates the component, in the x coordinate direction, of the vector representing the direction and length of the viewing frustum (from the near plane to the far plane).

vector\_y – indicates the component, in the y coordinate direction, of the vector representing the direction and length of the viewing frustum.

vector\_z – indicates the component, in the z coordinate direction, of the vector representing the direction and length of the viewing frustum.

centre\_tilt – specifies the tilt angle of the viewing frustum in units of 2−16 degrees. centre\_tilt shall be in the range of −180 \* 216 to 180 \* 216 − 1, inclusive.

near\_azimuth\_range, near\_elevation range – specify the azimuth and elevation ranges, respectively, of the near plane region of the viewing frustum in units of 2−16 degrees. near\_azimuth\_range and near\_elevation\_range specify the range through the near (centre) point of the near plane region. near\_azimuth\_range shall be in the range of 0 to 360 \* 216, inclusive. near\_elevation\_range shall be in the range of 0 to 180 \* 216, inclusive.

far\_azimuth\_range, far\_elevation range – specify the azimuth and elevation ranges, respectively, of the far plane region of the viewing frustum in units of 2−16 degrees. far\_azimuth\_range and far\_elevation\_range specify the range through the farr (centre) point of the farr plane region. far\_azimuth\_range shall be in the range of 0 to 360 \* 216, inclusive. far\_elevation\_range shall be in the range of 0 to 180 \* 216, inclusive.

# Timed Metadata Signaling and Storage for Point Cloud Data (m50661, m54308)

(Notes from MPEG 128: Some definitions used in the contribution e.g. region needs to be aligned with other activities.)

## Timed Metadata Signaling and Storage for Point Cloud Data (m50661)

### General

Volumetric content distribution is gaining traction to deliver 6DoF immersive media experiences. Content distribution technologies such as MPEG’s ISO Base Media File Format (ISOBMFF) need to support point cloud content. Quality and priority indication during streaming of volumetric content is desirable in order to improve viewport dependent delivery and thereby optimize bandwidth utilization and quality of user experience. This contribution provides ISOBMFF-based timed metadata track signaling of quality, priority, heatmap information as well as other kinds of timed metadata to help improve distribution of point cloud content.

Quality or priority information signaled in the timed metadata track could indicate the quality or priority ranking value of a particular region within the point cloud, e.g., this could be a bounding box (possibly cubical or spherical), and/or it could refer to a region containing a particular point cloud object. Moreover, this indication of quality or priority could be tied to a particular viewport, in this case, the indication of quality or priority information can be done on a per-viewport basis for a list of pre-defined viewports. It indicates a quality / priority ranking value of a given region (a bounding box, and/or point cloud object) in the point cloud relative to other quality / priority ranking point cloud regions.

For instance, priority information may be signaled in a viewport-dependent manner in order to prioritize across different point cloud bounding regions and/or objects on a per-viewport basis. For each chosen viewport, the related priority of the different regions over the point cloud (bounding box, and/or object) may be signaled along with the descriptors of each bounding box. The priority ranking values may be assigned by the content provider and how they are determined is implementation specific. As the viewport-dependent priority changes, such signaling also supports the ability to indicate the change in the relative priorities of the different point cloud regions or objects, e.g., since the priority indication is signaled on a per viewport basis for a set of pre-defined viewports.

The point cloud region for the quality or priority-ranking is specified by various syntax elements depending on how the bounding box regions are structured. Two example bounding box structures are (i) cubical, (ii) spherical. Below provide the semantics of elements and attributes of the metadata track descriptions with a cubical bounding box only (with the understanding that a similar description can be considered for spherical bounding boxes).

### Proposed ISOBMFF Timed Metadata Signaling and Storage

This clause specifies a generic timed metadata track syntax for indicating point cloud regions. The purpose for the timed metadata track is indicated by the track sample entry type. Each sample specifies a point cloud region. When a point cloud region timed metadata track is linked to one or more media tracks with a 'cdsc' track reference, it describes each media track individually.

Point cloud timed metadata sample shall be a PointCloudTimedMetadataSample, which shall be an array of MetadataInformationBoxes.

The types of information which may be present in the sample shall be indicated in the PointCloudTimedMetadataSampleEntry.

**Sample entry formats**

The track sample entry type 'pcmd' shall be used. The sample entry of this sample type is specified as follows:

class PointCloudTimedMetadataSampleEntry() extends PointCloudRegionSampleEntry(type) (‘pcmd’)

{  
 unsigned int(8) metadata\_type\_indicator;  
}

class PointCloudRegionSampleEntry() extends MetaDataSampleEntry(type) {  
 PointCloudRegionConfigBox(); // mandatory  
 Box[] other\_boxes; // optional  
}

class PointCloudRegionConfigBox extends FullBox('pcrc', 0, 0) {  
   
 bit(7) reserved = 0;  
 unsigned int(1) dynamic\_range\_flag;  
 if (dynamic\_range\_flag == 0) {  
 unsigned int(32) static\_center\_x;

unsigned int(32) static\_center\_y;

unsigned int(32) static\_center\_z;

unsigned int(32) static\_range\_x;

unsigned int(32) static\_range\_y;

unsigned int(32) static\_range\_z;  
 }  
 unsigned int(8) num\_regions;  
}

The semantics are as follows:

metadata\_type\_indicator indicates the types of timed metadata information which may be present in the PointCloudTimedMetadataSample samples that utilize this sample entry.

(metadata\_type\_indicator & 0x01) equal to 1 indicates a relative quality information for the point cloud regions. A value of 0 indicates lowest quality.

(metadata\_type\_indicator & 0x02) equal to 2 indicates a relative priority information for the point cloud regions. A value of 0 indicates lowest priority.

(metadata\_type\_indicator & 0x04) equal to 4 indicates a heatmap signaling for point cloud regions. A value of 0 indicates lowest heatmap value.

Value 0 is not allowed. Bits 3, 4 and 5 in LSB 0 notation are reserved. Bits 6 and 7 in LSB 0 notation are for user-defined metadata types.

dynamic\_range\_flag equal to 0 specifies that the coordinates of the point cloud region remain unchanged in all samples referring to this sample entry. dynamic\_range\_flag equal to 1 specifies that the coordinates are indicated in the sample format.

static\_center\_x specifies the value x-coordinate of the center point (or it could be any other reference point on the cube corner) of the cubical point cloud region.

static\_center\_y specifies the value y-coordinate of the center point (or it could be any other reference point on the cube corner) of the cubical point cloud region.

static\_center\_z specifies the value z-coordinate of the center point (or it could be any other reference point on the cube corner) of the cubical point cloud region.

static\_range\_x specifies the x-coordinate length of the cubical point cloud region.

static\_range\_y specifies the y-coordinate length of the cubical point cloud region.

static\_range\_z specifies the z-coordinate length of the cubical point cloud region.

num\_regions specifies the number of point cloud regions in the samples referring to this sample entry. num\_regions shall be equal to 1. Other values of num\_regions are reserved.

**Sample formats**

Each sample specifies a point cloud region and associated timed metadata.

Point cloud timed metadata sample shall be a PointCloudTimedMetadataSample, which shall be an array of MetadataInformationBoxes.

[Comment: The proposed metadata sample format is an adaptation of the related OMAF timed metadata carriage formats proposed earlier in m49878 [34].]

aligned(8) class PointCloudTimedMetadataSample extends PointCloudRegionSample() {  
 MetadataInformationBox()[];

}

aligned(8) class MetadataInformationBox extends FullBox(‘mibx’, 0, flags) {  
 unsigned int(8) metadata\_type;

unsigned int(8) viewport\_id;

for (i = 0; i < num\_regions; i++)  
 unsigned int(8) value[i];

}

aligned(8) PointCloudRegionSample() {  
 for (i = 0; i < num\_regions; i++)   
 PointCloudRegionStruct(dynamic\_range\_flag, 1);  
}

The semantics are as follows:

metadata\_type specified the type of the point cloud region metadata information. The following value are defined:

1 indicates a relative quality information for the point cloud regions.

2 indicates a relative priority information for the point cloud regions.

4 indicates a heatmap signaling for point cloud regions.

0 is not allowed. Values 8, 16 and 32 are reserved. Values 64 and 128 are for user-defined metadata types.

viewport\_id is the identifier of the viewport for which the metadata is applicable, e.g., priority of different point cloud regions may change depending on the particular viewport. Recommended viewport information may either be signaled at the MPD or timed metadata track in the ISOBMFF.

[Comment: TuC on Carriage of Point Cloud Data [33] already contains a potential solution using timed metadata track and an MPD-based solution is provided in document m50564.]

value specifies the value assigned to a point cloud region which corresponds to the region partition at index [i]. Value can correspond to the quality, priority or heatmap indicator depending on metadata\_type.

interpolate Let the target media samples be the media samples in the referenced media tracks with composition times greater than or equal to the composition time of this sample and less than the composition time of the next sample. Then interpolate equal to 0 specifies that the values of center\_x, center\_y, center\_z, range\_x, range\_y and range\_z in this sample apply to the target media samples. interpolate equal to 1 specifies that the values of center\_x, center\_y, center\_z, range\_x, range\_y and range\_z that apply to the target media samples are linearly interpolated based on composition times, from the values of the corresponding fields in this sample and the previous sample. The value of interpolate for a sync sample, the first sample of the track, and the first sample of a track fragment shall be equal to 0. The semantics of interpolate are specified by the semantics of the structure containing this instance of PointCloudRegionStruct. When interpolate is not present in this instance of PointCloudRegionStruct, it is inferred as specified in the semantics of the syntax structure containing this instance of PointCloudRegionStruct.

center\_x specifies the value x-coordinate of the center point (or it could be any other reference point on the cube corner) of the cubical point cloud region.

center\_y specifies the value y-coordinate of the center point (or it could be any other reference point on the cube corner) of the cubical point cloud region.

center\_z specifies the value z-coordinate of the center point (or it could be any other reference point on the cube corner) of the cubical point cloud region.

range\_x specifies the x-coordinate length of the cubical point cloud region.

range\_y specifies the y-coordinate length of the cubical point cloud region.

range\_z specifies the z-coordinate length of the cubical point cloud region

## Comments in Timed Metadata Signaling and Storage for Point Cloud Data (m54308)

### Introduction

During MPEG 130, a new timed metadata track was defined to signal camera information in V3C carriage format in ISO/IEC 23090-10 [1]-[2]. Using this information, the receiver may render the V3C content based on the signalled real or virtual camera positions and orientations. The syntax and semantics were aligned with those in the MIV specification ISO/IEC 23090-12.

In this contribution, we propose some improvements in syntax and semantics of signaling camera information in V3C carriage format. The purpose is to best align with MIV, which uses the floating point format to signal various parameters for intrinsic and extrinsic camera information.

### Proposed Text Changes to Annex G of ISO/IEC 23090-10

**A.1 Camera Information**

**A.1.1 General**

This section specifies signaling of intrinsic and extrinsic camera information in V3C carriage format. Using this information, the receiver may render the V3C content based on the signalled real or virtual camera positions and orientations.

**A.1.2 Structures**

**A.1.2.1 Syntax**

aligned(8) class ExtCameraInfoStruct() {  
 unsigned int(10) camera\_id;   
 unsigned\_int(1) camera\_pos\_present;   
 unsigned int(1) camera\_ori\_present;

unsigned int (5) pos\_bit\_allocation;

unsigned int (5) ori\_bit\_allocation;

   unsigned int(2) reserved;

if (camera\_pos\_present){   
 signed int(32) camera\_pos\_x;   
 signed int(32) camera\_pos\_y;   
 signed int(32) camera\_pos\_z;   
 }  
 if (camera\_ori\_present){   
 signed int(32) camera\_quat\_x;   
 signed int(32) camera\_quat\_y;   
 signed int(32) camera\_quat\_z;   
 }  
}

aligned(8) class IntCameraInfoStruct() {  
 unsigned int(10) camera\_id;   
 unsigned int(8) camera\_type;   
 unsigned int(1) camera\_proj\_params\_present;   
 unsigned int(1) camera\_depth\_present;

unsigned int(5) fov\_bit\_allocation;

   unsigned int(7) reserved;  
 if (camera\_type==0 && camera\_proj\_params\_present){   
 signed int(32) erp\_horizontal\_fov;   
 signed int(32) erp\_vertical\_fov;   
 }  
 if (camera\_type==1 && camera\_proj\_params\_present){   
 signed int(32) perspective\_horizontal\_fov;   
 signed int(32) perspective\_vertical\_fov;   
 }  
 if (camera\_type==2 && camera\_proj\_params\_present){   
 signed int(32) ortho\_aspect\_ratio;   
 signed int(32) ortho\_horizontal\_size;   
 }  
 if (camera\_depth\_present){   
 unsigned int(32) camera\_near\_depth;   
 unsigned int(32) camera\_far\_depth;   
 }  
}

**A.1.2.2 Semantics**

camera\_id contains an identifying number that is used to identify a given (real or virtual) camera.

camera\_pos\_present equal to 1 indicates that the camera position parameters are present. If camera\_pos\_present is equal to 0, this indicates that the camera position parameters are not present.

camera\_ori\_present equal to 1 indicates that the camera orientation parameters are present. If camera\_ori\_present is equal to 0, this indicates that the camera orientation parameters are not present.

camera\_proj\_params\_present equal to 1 indicates that the camera projection parameters are present. If camera\_proj\_params\_present is equal to 0, this indicates that the camera projection parameters are not present.

camera\_depth\_present equal to 1 indicates that the camera depth parameters are present. If camera\_depth\_present is equal to 0, this indicates that the camera depth parameters are not present.

camera\_pos\_x, camera\_pos\_y and camera\_pos\_z respectively indicate the x, y and z coordinates of the location of the camera in meters in the global reference coordinate system. The values shall be in units of 2−16 meters. If pos\_bit\_allocation is present, then bit allocation shall be according to pos\_bit\_allocation, the values shall be in units of 2−X meters, where X is equal to pos\_bit\_allocation.

camera\_quat\_x, camera\_quat\_y and camera\_quat\_z indicate the x, y, and z components, respectively, of the orientation of the camera using the quaternion representation. The values shall be a floating-point value in the range of −1 to 1, inclusive. These values specify the x, y and z components, namely qX, qY and qZ, for the rotations that are applied to convert the global coordinate axes to the local coordinate axes of the camera using the quaternion representation. The fourth component of the quaternion qW is calculated as follows:

qW = sqrt(1 – ( qX2 + qY2 + qZ2 ) )

The point (w, x, y, z) represents a rotation around the axis directed by the vector (x, y, z) by an angle 2\*cos ^{-1}(w)=2\*sin ^{-1}(sqrt(x^{2}+y^{2}+z^{2})). If ori\_bit\_allocation is present, then bit allocation shall be according to ori\_bit\_allocation, the values shall be in units of 2−X meters, where X is equal to ori\_bit\_allocation.

camera\_type indicates the projection method of the camera. camera\_type equal 0 specifies ERP projection. camera\_type equal to 1 specifies a perspective projection. camera\_type equal to 2 specifies an orthographic projection. camera\_type values in range 3 to 255 are reserved for future use by ISO/IEC.

erp\_horizontal\_fov specifies the longitude range for an ERP projection corresponding to the horizontal size of the viewing frustum associated with the camera, in units of radians. The value shall be in the range 0 to 2π. Bit allocation shall be according to fov\_bit\_allocation.

erp\_vertical\_fov specifies the latitude range for an ERP projection corresponding to the vertical size of the viewing frustum associated with the camera, in units of radians. The value shall be in the range 0 to π. Bit allocation shall be according to fov\_bit\_allocation.

perspective\_horizontal\_fov specifies the horizontal field of view for perspective projection in radians. The value of shall be in the range of 0 and π. Bit allocation shall be according to fov\_bit\_allocation.

perspective\_aspect\_ratio specifies the relative aspect ratio of the viewing frustum associated with the camera for perspective projection (horizontal/vertical). Bit allocation shall be according to fov\_bit\_allocation.

ortho\_aspect\_ratio specifies the relative aspect ratio of the viewing frustum associated with the camera for orthogonal projection (horizontal/vertical). Bit allocation shall be according to fov\_bit\_allocation.

ortho\_horizontal\_size specifies in meters the horizontal size of the orthogonal part of the viewing frustum associated with the camera. Bit allocation shall be according to fov\_bit\_allocation.

camera\_near\_depth and camera\_far\_depth indicate the near and far depths (or distances) based on the near and far planes of the viewing frustum associated with the camera. The value shall be in units of 2−16 meters.

pos\_bit\_allocation indicates the number of bits to be allocated to the decimal place out of the 32 bits in order to achieve the desired level of precision, when expressing camera position information, namely camera\_pos\_x, camera\_pos\_y and camera\_pos\_z.

ori\_bit\_allocation indicates the number of bits to be allocated to the decimal place out of the 32 bits in order to achieve the desired level of precision, when expressing camera orientation information, namely camera\_quat\_x, camera\_quat\_y and camera\_quat\_z.

fov\_bit\_allocation indicates the number of bits to be allocated to the decimal place out of the 32 bits in order to achieve the desired level of precision, when expressing camera field-of-view information, namely erp\_horizontal\_fov, erp\_vertical\_fov, perspective\_horizontal\_fov, perspective\_aspect\_ratio, ortho\_aspect\_ratio and ortho\_horizontal\_size.

# Carriage of multiple atlases (m54292)

## General

In DIS text of ISO/IEC 23090-10 includes the carriage of V3C bitstream containing either single atlas or multiple atlases. Each atlas consists of one or more atlas tiles and the single atlas can be carried in either one V3C track or one or more atlas tile tracks.

For the carriage of single atlas of V3C bitstream, ‘v3c1’ or ‘v3cg’ V3C tracks are used. However, samples in 'v3c1’ or ‘v3cg’ V3C track are differ in according to the presence of atlas tile tracks. When atlas tile tracks present, each sample in 'v3c1’ or ‘v3cg’ V3C track only contains non-ACL NAL units. However, when atlas tile tracks are not present, each sample in this track contains either ACL or non-ACL NAL units. Since samples in same type of V3C track are different according to the presence of other tracks, it requires for file parse to parse same type of track differently. to reduce this ineffectiveness, **we suggest to introduce a new ‘v3sb’ of V3C track carrying non-ACL NAL units when atlas tile tracks present.**

For the carriage of multiple atlas of the V3C bitstream, ‘v3cb' V3C track carrying atlas parameter sets as well as SEI message which are applied to all atlases is introduced. Currently, the V3C unit header box shall present in the sample entry of ‘v3cb' V3C track. However, atlas parameter sets and SEI messages carried in this track are not limited to a certain atlas, **the V-PCC unit header box shall not present in the sample entry of this 'v3cb’ V3C track.**

Like ‘v3c1’ or ‘v3cg’ V3C track, samples in ‘v3a1’, ‘v3ag’ V3C track are different according to the presence of atlas tile tracks. **we also suggest to introduce a new ‘v3ab’ of V3C track carrying non-ACL NAL units associated with certain atlas when atlas tile tracks present.**

Thus, we propose the following aspects for carriage of atlas bitstream

* Introduce new ‘v3sb’ and ‘v3ab’ V3C tracks carrying non-ACL NAL units when atlas tile tracks present.
* Modify the presence of V3C unit header box in sample entry of ‘v3cb' V3C track
* Improve atlas parameter set sample group, sample formats and track references of existing V3C tracks

The below table describes the summary of our proposed approach. (update/addition is marked as yellow-highlighted)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Sample entry type** | | 'v3c1' | 'v3cg' | 'v3sb' | 'v3cb' | 'v3a1' | 'v3ag' | ‘v3ab’ | 'v3t1' |
| **V3C**  **SampleEntry** | VPS | yes | yes | yes | yes | no | no | no | N/A |
| Atlas parameter sets | yes | yes | yes | yes | yes1 | yes1 | yes | N/A |
| V3C unit header box | yes | yes | yes | ~~yes~~  no | yes | yes | yes | N/A |
| **Atlas parameter set sample group** | | may | may | no | no | may1 | may1 | no | N/A |
| **Track references** | | 'v3vo'  'v3vg'  'v3va'  ~~'v3ct'~~ | 'v3vo'  'v3vg'  'v3va'  ~~'v3ct'~~ | 'v3ct' | 'v3cs' | 'v3vo'  'v3vg'  'v3va'  ~~'v3ct'~~ | 'v3vo'  'v3vg'  'v3va'  ~~'v3ct'~~ | 'v3ct' | 'v3vo'  'v3vg'  'v3va' |
| **Sample** | | ACL + non-ACL | ACL + non-ACL | non-ACL | non-ACL | ACL + non-ACL | ACL + non-ACL | non-ACL | ACL |

1 It only includes atlas parameter sets associated with same atlas\_id indicated in V3C unit header box.

Based on the proposed approach, the suggested medication on the DIS text of ISO/IEC 23090-10 in [1] is marked as green-highlighted. in clause 20.2.

## Proposed Specification Text

### V3C atlas parameter set sample group

#### General

The use of 'vaps' for the grouping\_type in sample grouping represents the assignment of samples in V3C track to the atlas parameter sets carried in this sample group. When a SampleToGroupBox with grouping\_type equal to 'vaps' is present, an accompanying SampleGroupDescriptionBox with the same grouping type shall be present and contains the ID of the group that the samples belong to.

A V3C track may contain at most one SampleToGroupBox with grouping\_type equal to 'vaps'.

When the'v3c1' or 'v3e1' sample entry is used in V3C track, NAL units of type NAL\_ASPS, NAL\_AAPS, NAL\_AFPS, NAL\_PREFIX\_ESEI, NAL\_PREFIX\_NSEI, NAL\_SUFFIX\_ESEI, and NAL\_SUFFIX\_NSEI (as defined in ISO/IEC 23090-5 [V3C]) shall not be present in the samples of this sample group.

When the 'v3cg' or 'v3eg' sample entry is used in V3C track, NAL units of type NAL\_ASPS, NAL\_AAPS, NAL\_AFPS, NAL\_PREFIX\_ESEI, NAL\_PREFIX\_NSEI, NAL\_SUFFIX\_ESEI, and NAL\_SUFFIX\_NSEI may be present in the samples of this sample group.

When the 'vpcb', 'vpsb' , or 'vpab' sample entry is used, this sample group shall not present.

When the 'vpa1' or 'vpag' sample entry is used, this sample group includes non-ACL NAL units carrying atlas parameter sets as well as SEI messages associated with the same atlas\_id indicated in the sample entry.

#### Syntax

aligned(8) class VPCCAtlasParamSampleGroupDescriptionEntry() extends SampleGroupDescriptionEntry('vaps') {  
 unsigned int(8) numOfSetupUnits;  
 for (i=0; i < numOfSetupUnits; i++) {  
 unsigned int(16) setupUnitLength;  
 nal\_unit(setupUnitLength) setupUnit; // as defined in ISO/IEC 23090-5  
 }

}

#### Semantics

numOfSetupUnits specifies the number of setup units signalled in the sample group description.

setupUnitLength indicates the size, in bytes, of the setupUnit field. The length field includes the size of both the NAL unit header and the NAL unit payload but does not include the length field itself.

setupUnit is a NAL unit of type NAL\_ASPS, NAL\_AFPS, , NAL\_PREFIX\_ESEI, NAL\_PREFIX\_NSEI, NAL\_SUFFIX\_ESEI, or NAL\_SUFFIX\_ESEI carrying data associated with this group of samples.

///////////////////////////////////

### V3C tracks

#### V3C Track Sample Entry

Sample Entry Type: 'v3c1', 'v3cg', ‘v3sb’, 'v3cb', ‘v3ab’, 'v3a1', or 'v3ag'  
Container: SampleDescriptionBox  
Mandatory: A 'v3c1' or 'v3cg' , or 'v3cb' sample entry is mandatory  
Quantity: One or more sample entries may be present

V3C tracks shall use V3CSampleEntry which extends VolumetricVisualSampleEntry with a sample entry type of 'v3c1', 'v3cg', ’v3sb’, v3cb', ’v3ab’, 'v3a1', or 'v3ag'. A V3C track shall not carry ACL NAL units belonging to more than one atlas.

When the V3C bitstream contains a single atlas, a V3C track with sample entry 'v3c1', 'v3cg', or ’v3sb’ shall be used.

When the V3C bitstream contains multiple atlases, each atlas bitstream shall be encapsulated in a separate V3C track. One of those tracks shall use sample entry type 'v3cb', while the other tracks shall use sample entry type 'v3a1', 'v3ag', or ’v3ab’.

A V3C track with sample entry type 'v3cb' shall not include any ACL NAL units.

A V3C track sample entry shall contain a V3CConfigurationBox, as defined in ‎6.2.2. Under 'v3c1', 'v3cg', ‘v3sb’, ‘v3ab’, 'v3a1', or 'v3ag' sample entry, a V3CUnitHeaderBox, as defined in ‎6.2.3 shall be present.

Under the 'v3a1', 'v3ag', and ’v3ab’ sample entry, no V3C parameter set shall be stored in the v3cParameterSets array.

Under the 'v3c1' sample entry, all atlas sub-bitstream parameter sets and SEI messages shall be in the setupUnit array. Under the 'v3cg' and 'v3cb' sample entry, the atlas sub-bitstream parameter sets and SEI messages may be present in the setupUnit array, or in the atlas sub-bitstream.

Under the 'v3a1' sample entry, atlas sub-bitstream parameter sets associated with the same atlas\_id indicated in the V3C unit header box shall be in the setupUnit array. Under the 'v3ag' and ’v3ab’ sample entry, atlas sub-bitstream parameter sets associated with the same atlas\_id indicated in the V3C unit header box may be present in the setupUnit array, or in the atlas sub-bitstream.

An optional BitRateBox may be present in the V3C sample entry to signal the bit rate information of the V3C track.

//////

### 6.4.4. V3C sample format

#### Definition

Each sample in a V3C track or V3C atlas tile track corresponds to a single coded atlas access unit. Samples corresponding to this frame in the various component tracks shall have the same composition time as the V3C track sample. Each V3C sample shall only contain one V3C unit payload of type V3C\_AD as defined in ISO/IEC 23090-5 [V3C], which may include one or more atlas NAL units.

When 'v3cb' sample entry is used, each sample in the V3C track corresponds to one or more non-ACL NAL units.

When 'v3c1', 'v3cg', 'v3a1' or 'v3ag' sample entry is used, each sample in the V3C track corresponds to one or more atlas NAL units

~~and one or more atlas tile tracks are present,~~ When 'v3sb' or ‘v3ab’ sample entry is used, each sample in the V3C track corresponds to one or more non-ACL NAL units associated with same atlas\_id indicated in V3C unit header box in sample entry.

//////

### 6.4.6. Track references

#### Referencing V3C tracks

To link a V3C track with sample entry 'v3cb' to V3C tracks with sample entries 'v3a1', 'v3ag', or 'v3ab', the track reference tool of ISO/IEC 14496-12 [14496-12] shall be used. The 4CCs of these track reference types shall be 'v3cs'.

#### Referencing V3C atlas tile tracks

To link a V3C track with sample entries ~~'v3c1', 'v3cg', 'v3a1' or 'v3ag'~~ 'v3sb' or 'v3ab'to V3C atlas tile tracks with sample entry 'v3t1', the track reference tool of ISO/IEC 14496-12 [14496-12] shall be used. The 4CCs of these track reference types shall be 'v3ct'.

A V3C track with sample entries 'v3c1', 'v3cg', 'v3a1' or 'v3ag' shall not use the 4CCs of track reference type ‘v3ct’.

#### Referencing V3C component tracks

To link a V3C track with sample entries 'v3c1', 'v3cg', ’v3a1’, or ’v3ag’, or a V3C atlas tile track with sample entry 'v3t1' to component video tracks, the track reference tool of ISO/IEC 14496-12 [14496-12] shall be used. One or more TrackReferenceTypeBoxes shall be added to a TrackReferenceBox within the TrackBox of the V3C track or V3C atlas tile track, one for each component. The TrackReferenceTypeBox shall contain an array of track\_IDs designating the video tracks which the V3C track references. The reference\_type of a TrackReferenceTypeBox identifies the type of the component (i.e., occupancy map, geometry, or attribute, or occupancy map). The 4CCs of these track reference types shall be:

* 'v3vo': the referenced track(s) contain the video-coded occupancy map V3C component
* 'v3vg': the referenced track(s) contain the video-coded geometry V3C component
* 'v3va': the referenced track(s) contain the video-coded attribute V3C component

The type of the V3C component carried by the referenced restricted video track, and signalled in the RestrictedSchemeInfoBox of the track, shall match the reference type of the track reference from the V3C track or V3C atlas tile track.

~~Note: When 'v3ct' track reference is present, 'v3va', 'v3vo', 'v3vg' references shall not be used.~~

# Carriage of V-PCC data with Multiple Atlases (m53503)

This contribution proposes system signalling for the encapsulation of V-PCC data with multiple atlases to support efficient access, delivery and rendering of volumetric visual media compressed as MPEG Immersive Video defined in ISO/IEC 23090-12 in a 6DoF environment.

## Background

### Group-based encoder

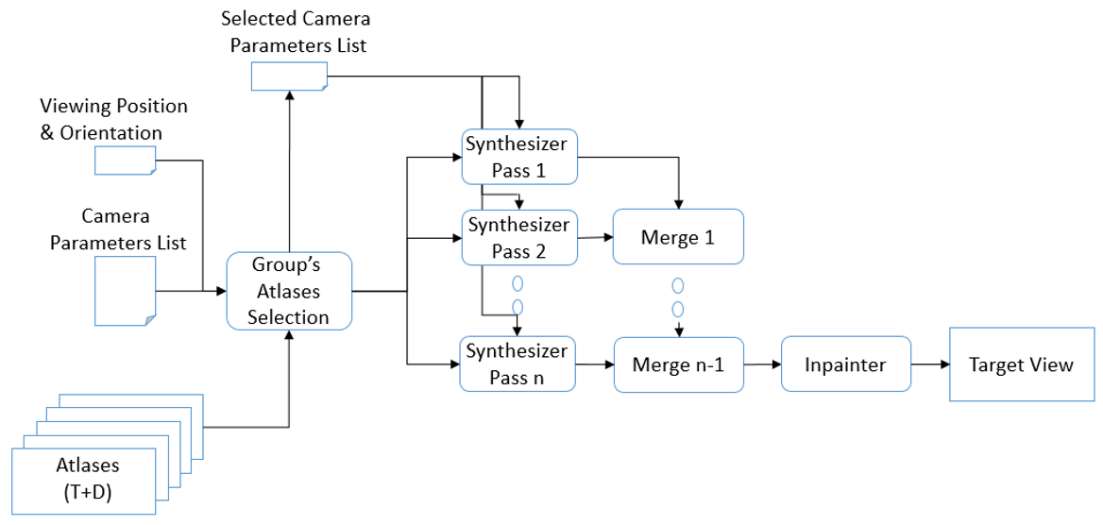
The group-based encoder (Figure 21.1) is the top-level encoder of TMIV. It splits the views into multiple groups, and encodes each of the groups independently using multiple single-group encoders. The resulting metadata is merged into a single MIV bitstream.



**Fig.21.1 A process flow of group-based encoding for atlases generation**

As shown in Figure 21.1, at group’s encoding stage, each single-group encoder produces metadata with its own indexed atlases or views. A unique group ID is assigned per group and attached to the atlas parameters of the related group. In order to enable the renderer to interpret the metadata properly and map the patches correctly across all views, the merger renumbers atlas and view ID's per patch and merges the pruning graphs.

### Group-based renderer

1. 

**Fig.21.2 Process flow for the group-based renderer**

As shown in Figure 21.2, all atlases generated from the same view group by the atlas constructor shoud be grouped together as an atlas group.For the group-based rendering, the decoder needs to decode patches within one or more atlas groups corresponding to one or more view groups from which one or more views of the volumetric visual data (e.g. MIV content) have been selected for the target view rendering.

### Carriage of V-PCC data with multiple atlases



**Fig.21.3 Multi-track encapsulation of V-PCC bitstream with multiple atlases**

As shown in Figure 21.3, before the decoding of an atlas group, a file parser needs to determine and decapsulate a group of V-PCC tracks corresponding to the atlas group based on a syntax element of a V-PCC parameter track in a file storage of the bitstream; wherein the group of V-PCC tracks and the V-PCC parameter track carry all the atlas data for the atlas group.

In section 3 of this document, we propose the followings.

* **ViewGroupInfoStruct** provides the view group information of MIV content captured and processed at the encoding stage which can be used e.g. to compute the distance between the view position and the target viewport position for the atlas group selection
* **VPCC track grouping** indicates V-PCC tracks corresponding to all atlas sub-streams from the same atlas group by a track group of type 'vptg'.
* **VPCCViewGroupsBox** signals static view groups for a MIV content and their respective associated V-PCC track groups
* **Timed-metadata track** with a sample entry type 'dyvg' associated with the V-PCC parameter track to provide change information of dynamic view groups for the MIV stream

## Proposals

### View group information structure

#### Definition

ViewGroupInfoStruct provides the view group information of MIV content captured and processed at the encoding stage, including at least: the view group identifier, the view group description, the number of views, the view identifier and camera parameters for each view.

CameraParametersStruct provides the real or virtual camera position and orientation information, which can be used to render V-PCC or MIV content as either perspective viewport or omnidirectional view at the desired viewing position and orientation.

#### Syntax

aligned(8) class CameraParametersStruct() {

//To be defined e.g. based on outputs of the CE on Immersive media metadata

}

aligned(8) class ViewGroupInfoStruct(camera\_parameters\_included\_flag) {  
 unsigned int(16) view\_group\_id;

String view\_group\_description;

unsigned int(8) num\_views;

for (i=0; i < num\_views; i++) {  
 unsigned int(16) view\_id;

unsigned int(1) basic\_view\_flag;

if (camera\_parameters\_included\_flag) {

CameraParametersStruct();  
 }

}  
}

#### Semantics

view\_group\_id provides the identifier for the view group.

view\_group\_descritption is null-terminated UTF-8 string that provides a textual description of the view group.

num\_views specifies the number of views in the view group.

view\_id provides the identifier for the given view in the view group.

basic\_view\_flag equal to 1 specifies that the associated view is selected as basic view. basic\_view\_flag equal to 0 specifies that the associated view is not selected as basic view.

camera\_parameters\_included\_flag equal to 1 indicates that the CameraParametersStruct is present. camera\_parameters\_included\_flag equal to 0 indicates that the CameraParametersStruct is not present.

### V-PCC Track Grouping

Since all atlases generated from the same view group by the atlas constructor will be grouped together as an atlas group, for the storage of V-PCC bitstream with multiple atlases, all V-PCC tracks corresponding to all atlases from the same atlas group should be indicated by a track group of type 'vptg'.

#### Definition

TrackGroupTypeBox with track\_group\_type equal to 'vptg' indicates that this V-PCC track belongs to a group of V-PCC tracks that correspond to an atlas group.

V-PCC tracks belonging to the same atlas group have the same value of track\_group\_id for track\_group\_type 'vptg', and the track\_group\_id of tracks from one atlas group differs from the track\_group\_id of tracks from any other atlas group.

#### Syntax

aligned(8) class VPCCTrackGroupBox extends trackGroupTypeBox('vptg'){

}

#### Semantics

V-PCC tracks that have the same value of track\_group\_id within TrackGroupTypeBox with track\_group\_type equal to 'vptg' belong to the same atlas group. The track\_group\_id within TrackGroupTypeBox with track\_group\_type equal to 'vptg' is therefore used as the identifier of the atlas group.

### Static view group information box

#### Definition

Static view groups for a MIV content and their respective associated V-PCC track groups shall be signalled in a VPCCViewGroupsBox.

#### Syntax

Box Types: 'vpvg'  
Container: VPCCParametersSampleEntry ('vpcp')  
Mandatory: No  
Quantity: Zero or one

aligned(8) class VPCCViewGroupsBox extends FullBox('vpvg',0,0) {  
 unsigned int(16) num\_view\_groups;  
 for (i=0; i < num\_view\_groups; i++) {  
 ViewGroupInfoStruct(1);  
 unsigned int(32) vpcc\_track\_group\_id;  
 }

}

#### Semantics

num\_view\_groups indicates the number of view groups for the MIV content.

vpcc\_track\_group\_id identifies the group for the V-PCC tracks which carry all the atlas data for the associated view group of the MIV content.

### Dynamic view group information

If the V-PCC parameter track has an associated timed-metadata track with a sample entry type 'dyvg', source view groups defined for the MIV stream carried by the V-PCC parameter track are considered as dynamic view groups (i.e., the view group information may dynamically change over time).

The associated timed-metadata track shall contain a 'cdsc' track reference to the V-PCC parameter track carrying the atlas stream.

#### Sample entry

aligned(8) class DynamicViewGroupSampleEntry extends MetaDataSampleEntry('dyvg') {  
 VPCCViewGroupsBox();

}

#### Sample format

##### Syntax

aligned(8) DynamicViewGroupSample() {  
 unsigned int(16) num\_view\_groups;  
 for (i=0; i < num\_view\_groups; i++) {

ViewGroupInfoStruct(camera\_parameters\_included\_flag);  
}

}

##### Semantics

num\_view\_groups indicates the number of view groups signalled in the sample. This may not necessarily be equal to the total number of available view groups. Only view groups whose source views are being updated are present in the sample.

ViewGroupInfoStruct() is defined in the section 3.1.2. If camera\_parameters\_included\_flag is set to 0, this implies that the camera parameters of the view group have been previously signalled in a previous instance of a ViewGroupInfoStruct with the same view\_group\_id, either in a previous sample or in the sample entry.

# VPCC spatial region descriptor (m53504)

[Ed(EA): Based on MPEG 130 notes on m53504, vpcc:@regionIds is included to the TuC for further study of the remaining Editor note under Table 3 of DIS spec]

This contriburion proposes the following:

* Updating the VPCC descriptor and VPCC3DRegions descriptor, to add @regionIds attribute to reflect the preselected V-PCC components corresponding to one or more 3D spatial regions of the point cloud.

The **vpcc:**@regionIds of VPCC descriptor intends to play different role from the **spatialRegion**@id of VPCC3DRegions descriptor. The former indicates which spatial regions of the point cloud object to be preselected, while the latter indicates which spatial regions the point cloud object contains.

## Proposed specification text

The proposed specification text is highlighted as green

#### 9.2.3.3 V-PCC descriptor

An **EssentialProperty** element with a @schemeIdUri attribute equal to "urn:mpeg:mpegI:vpcc:2019:vpc" is referred to as a VPCC descriptor. At most one VPCC descriptor may be present at the adaptation set level for the main **AdaptationSet** of the point cloud.

Table 1 Attributes for the VPCC descriptor

|  |  |  |  |
| --- | --- | --- | --- |
| **Attributes for VPCC descriptor** | **Use** | **Data type** | **Description** |
| **vpcc:**@pcId | CM | xs:string | An id for the point cloud. This attribute shall be present if multiple versions of the same point cloud are signalled in separate AdapatationSets. |
| **vpcc:**@regionIds | CM | xs:stringVectorType | A list of space separated identifiers for the spatial region of the point cloud. This attribute shall be present if the V-PCC content correspond to one or more 3D spatial regions of the point cloud. |
| Legend:  For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory.  For elements: <minOccurs>..<maxOccurs> (N=unbounded)  Elements are **bold**; attributes are non-bold and preceded with an @. | | | |

# Carriage of non-timed V3C data (m54293)

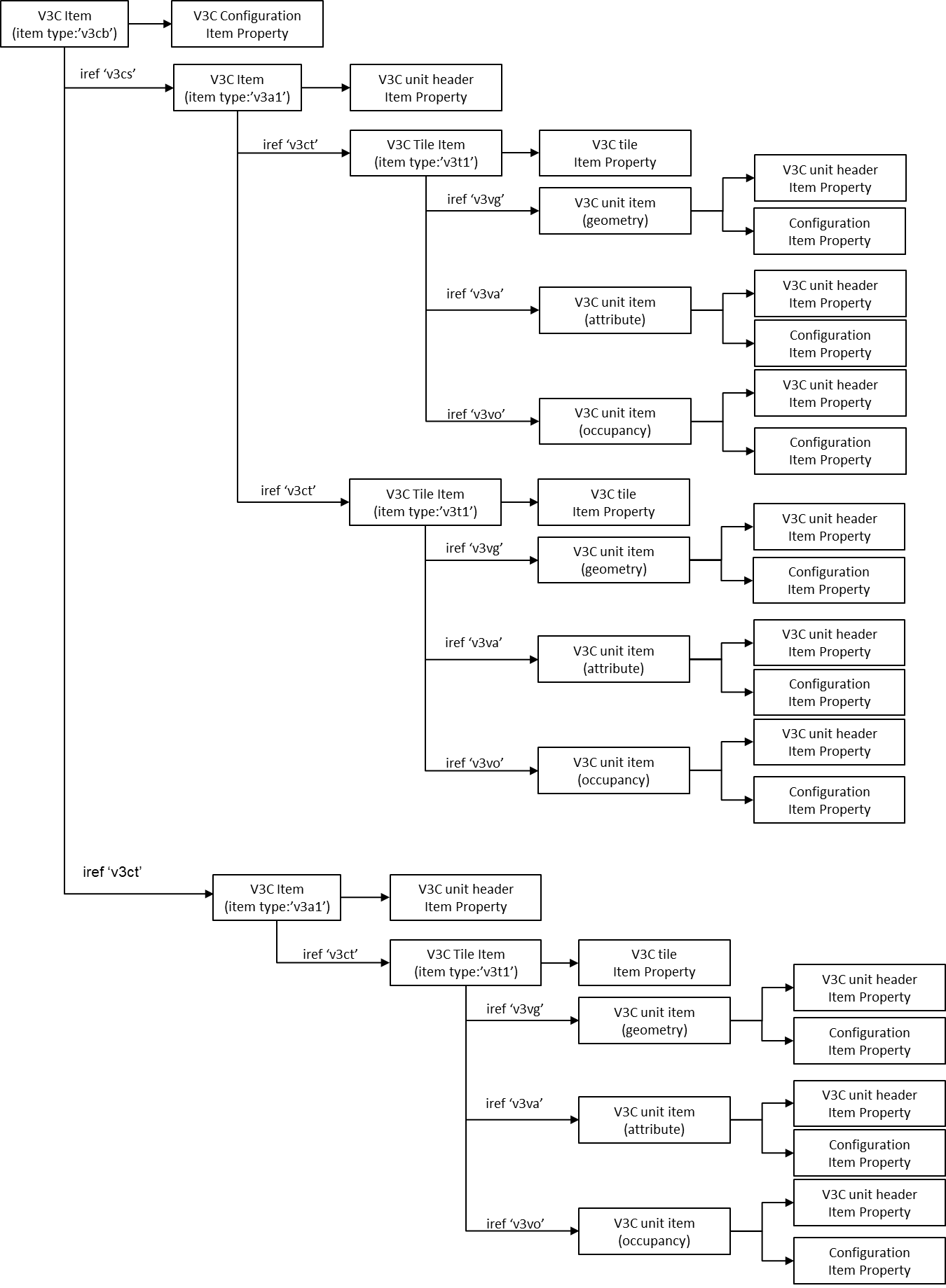
## General

In DIS text of ISO/IEC 23090-10, it only considers the carriage of non-timed V3C data with single atlas even though non-timed V3C data containing multiple atlases. Thus, we need to consider the carriage of non-timed V3C data with multiple atlases.

To achieve this, we propose following aspects:

* We add new item types and item references of V3C items for either carriage of either single atlas or multiple atlases
* We specify association with item properties according to either single atlas or multiple atlases.
* We introduce a new V3C tile item to store one or more atlas tiles and to define a new item property related to this V3C tile item.

Based on the proposed approach, an overview of the structure for encapsulating non-timed V3C data containing multiple atlases is illustrated in below Figure.



In clause 25.2, the proposed addition for the support of multiple atlases in carriage of non-timed V3C data is described in yellow-highlighted.

## Proposed Specification Text

**7 Carriage of non-timed Visual Volumetric Video-based Coding Data**

**7.1 V3C Items**

A V3C item is an item which represents an independently decodable V3C access unit. V3C items store one or more ACL or non-ACL NAL units.

A V3C item shall be stored as an item of type 'v3cb', 'v3c1', or 'v3a1'.

An item of type ‘v3c1’ shall be associated with one V3CConfigurationProperty and one V3CUnitHeaderProperty. . If V3C tile items are present, an item of type ‘v3a1’ stores one or more non-ACL NAL units associated with same atlas\_id indicated in the V3CUnitHeaderProperty .

An item of type ‘v3cb' shall be associated with one V3CConfigurationProperty and store one or more non-ACL NAL units.

An item of type ‘v3a1’ shall be associated with one V3CUnitHeaderProperty. If V3C tile items are present, an item of type ‘v3a1’ stores one or more non-ACL NAL units associated with same atlas\_id indicated in the V3CUnitHeaderProperty .

In order to indicate the relationship between a ‘v3cb' V3C item to ‘v3a1' V3C items, a new item reference type with 4CC codes 'v3cs’ is used.

In order to indicate the relationship between a V3C item to V3C tile items, a new item reference type with 4CC codes 'v3ct’ is used. This item reference is defined “from” a V3C item “to” the related V3C tile items.

If PrimaryItemBox exists, item\_id in this box shall be set to indicate a V3C item.

**7.2 V3C tile Items**

A V3C tile item shall be stored as an item of type 'v3t1', and contain ACL NAL units, which belong to the same atlas, at least one tile.

Each V3C tile item shall be associated with one V3CTileConfigurationProperty. The V3CTileConfigurationProperty shall indicate the tile address of tiles present in the V3C tile item.

**7.3 V3C Unit Item**

A V3C unit item is an item which represents a V3C unit data. V3C unit items store V3C unit payload(s) of occupancy, geometry, and attribute video data units. A V3C unit item shall store only one V3C access unit related data.

An item type 4CC code for a V3C unit item shall be set depending on the codec used to encode corresponding video data units. A V3C unit item shall be associated with corresponding V3C unit header item property and codec specific configuration item property.

V3C unit items shall be marked as hidden items, since it is not meaningful to display independently.

In order to indicate the relationship between a V3C item and V3C units or between a V3C tile item and V3C units, three new item reference types with 4CC codes 'v3vo', 'v3vg' and 'v3va' are defined. Item reference is defined “from” either a V3C item or a V3C tile item “to” the related V3C unit items. The 4CC codes of item reference types shall be:

* 'v3vo': the referenced V3C unit item(s) contain the occupancy video data units.
* 'v3vg': the referenced V3C unit item(s) contain the geometry video data units.
* 'v3va': the referenced V3C unit item(s) contain the attribute video data units.

**7.4 V3C-related item properties**

Two new descriptive item properties are defined to carry the V3C parameter set information and V3C unit header information, respectively:

**7.4.1 V3C configuration item property**

**7.4.1.1 Definition**

Box Types: 'v3cp'  
Property type: Descriptive item property  
Container: ItemPropertyContainerBox  
Mandatory (per item): Yes, for a V3C item of type 'v3ci'  
Quantity (per item): One or more for a V3C item of type 'v3ci'

V3C parameter sets are stored as descriptive item properties and shall be associated with the V3C items. In this version of the specification, only one V3C parameter set is allowed to be stored in V3C configuration item property.

essential shall be set to 1 for a 'v3cp' item property.

**7.4.1.2 Syntax**

aligned(8) class v3c\_unit\_payload\_struct () {  
 unsigned int(16) v3c\_unit\_payload\_size;  
 v3c\_unit\_payload();  
}

aligned(8) class V3CConfigurationProperty extends ItemProperty('v3cp') { v3c\_unit\_payload\_struct()[];  
}

**7.4.1.3 Semantics**

v3c\_unit\_payload\_size specifies the size in bytes of the v3c\_unit\_payload().

v3c\_unit\_payload() is a V3C unit of type V3C\_VPS as defined in ISO/IEC 23090-5 [V3C].

**7.4.2 V3C unit header item property**

**7.4.2.1 Definition**

Box Types: 'vunt'  
Property type: Descriptive item property  
Container: ItemPropertyContainerBox  
Mandatory (per item): Yes, for a V3C item of type 'v3ci' and for a V3C unit item  
Quantity (per item): One

V3C unit header is stored as descriptive item properties and shall be associated with the V3C items and the V3C unit items.

essential shall be set to 1 for a 'vunt' item property.

**7.4.2.2 Syntax**

aligned(8) class V3CUnitHeaderProperty() extends ItemFullProperty('vunt', version=0, 0) {  
 v3c\_unit\_header();  
}

**7.4.2.3 Semantics**

v3c\_unit\_header() is as defined in ISO/IEC 23090-5.

**7.4.3 V3C tile configuration item property**

**7.4.3.1 Definition**

Box Types: 'v3tp'  
Property type: Descriptive item property  
Container: ItemPropertyContainerBox  
Mandatory (per item): Yes, for a V3C item of type 'v3t1'  
Quantity (per item): One

V3CTileConfigurationProperty is stored as descriptive item properties and shall be associated with the V3C tile items.

essential shall be set to 1 for a v3tp’ item property.

aligned(8) class V3CTileConfigurationProperty () extends ItemFullProperty(‘v3tp’, version=0, 0) {  
 unsigned int(16) num\_tiles;  
 for(i=0; i < num\_tiles; i++) {  
 unsigned int(16) tile\_id;  
 }

}

**7.4.3.2 Semantics**

**num\_tiles** indicates the number of tiles contained in related V3C tile item.

**tile\_id** indicates the tile address of the tile contained in related V3C tile item.

# Partial access of V3C data (m54396)

## General

### View rendering process

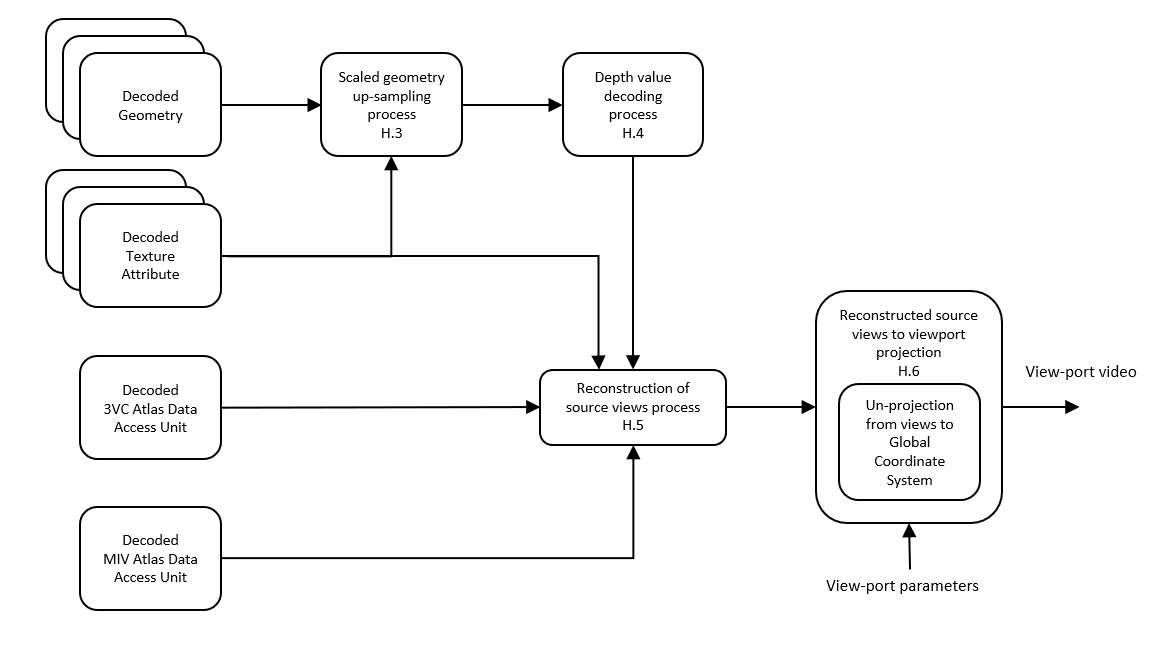


Figure 1: Block diagram of hypothetical view renderer

Figure 1 depicts a hypothetical view rendering process. Inputs to this process are the following:

* A number of atlases(vps\_atlas\_count\_minus1) and for each of the atlases: one decoded geometry picture, one texture attribute picture and one BlockToPatchMap etc
* A number of views(mvp\_num\_views\_minus1)
* The intrinsic and extrinsic parameters per view
* A viewing position (x, y, z), a viewing orientation (quat\_x, quat\_y, quat\_z)
* A viewport picture width and height

The rendering engine will invoke the reconstructed views process for each view and projection pixels of reconstructed view to target viewport. The output is a perspective viewport or omnidirectional view according to a desired viewing pose, enabling motion parallax cues within a limited space.

### Partial access utilizing V3C View information



*Figure 2: Clip of multi-view video(view#1 and view#2)*

Source view indicates source video material before encoding that corresponds to the format of a view representation. As shown in Figure 2, volumetric media coded as V3C data can provide users with six degrees of freedom (6-DoF) immersive media experiences. At any given time, only a part of volumetric media (in the case of MIV content, only one view from multiple views) will be visible depending on the user’s view position, orientation etc. For many applications, the entire volumetric media data does not need to be delivered, decoded and rendered.

From partial access point of view, similar to the mapping between spatial region/object and associated tracks, it’s also beneficial to utilize view information for partial access of V3C data in many streaming scenarios.

In clause 26.2 of this document, we propose the followings.

Partial Access of V3C data

* **ViewInfoStruct** provides the view information of V3C content captured and processed at the encoding stage which can be used e.g. to compute the distance between the view position and the target viewport position for the atlas selection
* **V3C track grouping** indicates V3C tracks corresponding to all atlas sub-streams from the same view by a track group of type 'vptg'.
* **V3CViewsBox** signals static views for a V3C content and their associated V3C tracks
* **Timed-metadata track** with a sample entry type 'dyvw' associated with the V3C track to provide change information of dynamic views for the V3C bitstream

DASH signalling

* **Static views** identified using a **V3CViews** descriptor in the main **AdaptationSet**
* **Dynamic views** as a timed-metadata track shall be carried in a separate **AdaptationSet**

## Proposed Specification Text

**8 Partial Access of Volumetric Visual Data**

**8.6 View information structure**

**8.6.1 Definition**

ViewInfoStruct provides the view information of V3C content captured and processed at the encoding stage, including at least: the view identifier, the view description and view parameters for the view.

ViewParametersStruct defines the projection used to generate a view representation from a 3D scene, including intrinsic and extrinsic parameters.

**8.6.2 Syntax**

aligned(8) class ViewParametersStruct() {

nal\_unit camera\_intrinsics; // as defined in ISO/IEC 23090-12

nal\_unit camera\_extrinsics; // as defined in ISO/IEC 23090-12

}

aligned(8) class ViewInfoStruct(view\_parameters\_included\_flag) {  
 unsigned int(16) view\_id;

String view\_description;

unsigned int(1) basic\_view\_flag;

if (view\_parameters\_included\_flag) {

ViewParametersStruct();  
}

}

**8.6.3 Semantics**

view\_id provides the identifier for the view.

view\_descritption is null-terminated UTF-8 string that provides a textual description of the view.

basic\_view\_flag equal to 1 specifies that the associated view is selected as basic view. basic\_view\_flag equal to 0 specifies that the associated view is not selected as basic view.

view\_parameters\_included\_flag equal to 1 indicates that the ViewParametersStruct is present. view\_parameters\_included\_flag equal to 0 indicates that the ViewParametersStruct is not present.

camera\_intrinsics and camera\_extrinsics contain a NAL unit of type NAL\_AAPS with MIV extension as defined in ISO/IEC 23090-12

**8.7 View Track Grouping**

For the storage of V3C bitstream with multiple views, all V3C tracks corresponding to the same view should be indicated by a track group of type 'vwtg'.

**8.7.1 Definition**

TrackGroupTypeBox with track\_group\_type equal to 'vwtg' indicates that this V3C track belongs to a group of V3C tracks that correspond to a view of V3C content.

V3C tracks belonging to the same view have the same value of track\_group\_id for track\_group\_type 'vwtg', and the track\_group\_id of tracks from one view differs from the track\_group\_id of tracks from any other view.

**8.7.2 Syntax**

aligned(8) class ViewTrackGroupBox extends trackGroupTypeBox('vwtg'){

}

**8.7.3 Semantics**

V3C tracks that have the same value of track\_group\_id within TrackGroupTypeBox with track\_group\_type equal to 'vptg' belong to the same atlas group. The track\_group\_id within TrackGroupTypeBox with track\_group\_type equal to 'vptg' is therefore used as the identifier of the atlas group.

**8.8 Static view information box**

**8.8.1 Definition**

Static views for a V3C content and their respective associated V3C tracks shall be signalled in a V3CViewsBox.

**8.8.2 Syntax**

Box Types: 'v3vw'  
Container: V3CSampleEntry ('v3c1'or 'v3cg')  
Mandatory: No  
Quantity: Zero or one

aligned(8) class V3CViewsBox extends FullBox('v3vw',0,0) {  
 unsigned int(16) num\_views;  
 for (i=0; i < num\_views; i++) {  
 ViewInfoStruct(1);  
 unsigned int(8) num\_v3c\_tracks;

for (j=0; j < num\_v3c\_tracks; j++) {  
 unsigned int(32) v3c\_track\_id;

}  
}

}

**8.8.3 Semantics**

num\_views indicates the number of views for the V3C content.

vpcc\_track\_id identifies the V3C track which carry the atlas data for the associated view.

**8.9 Dynamic view information**

If the V3C track has an associated timed-metadata track with a sample entry type 'dyvw', source views defined for the V3C stream carried by the V3C track are considered as dynamic views (i.e., the view information may dynamically change over time).

The associated timed-metadata track shall contain a 'cdsc' track reference to the V3C track carrying the atlas sub-stream.

1.1.1 8.9.1 Sample entry

aligned(8) class DynamicViewSampleEntry extends MetaDataSampleEntry('dyvw') {  
 V3CViewsBox();

}

**8.9.2 Sample format**

**8.9.2.1 Syntax**

aligned(8) DynamicViewSample() {  
 unsigned int(16) num\_views;  
 for (i=0; i < num\_views; i++) {

ViewInfoStruct(view\_parameters\_included\_flag);  
}

}

**8.9.2.2 Semantics**

num\_views indicates the number of views signalled in the sample. Only views are being updated are present in the sample.

ViewInfoStruct() is defined in the section 8.6.2. If view\_parameters\_included\_flag is set to 0, this implies that the view parameters of the view have been previously signalled in a previous instance of a ViewInfoStruct with the same view\_id, either in a previous sample or in the sample entry.

**9 Encapsulation and signalling in MPEG-DASH**

**9.5.3 Static Views**

To identify the static views in the main **AdaptationSet** for the V3C content and their respective associated V3C tracks, a **V3CViews** descriptor shall be used. A **V3CViews** is a **SupplementalProperty** descriptor with the @schemeIdUri attribute equal to "urn:mpeg:mpegI:v3c:2020:vpvw".

At most one single **V3CViews** descriptor shall be present at the adaptation set level or the representation level in the main **AdaptationSet** or at the preselection level for the V3C content.

The @value attribute of the **V3CViews** descriptor shall not be present. The **V3CViews** descriptor shall include elements and attributes as specified in Table 4.

Table 4 Elements and attributes for the V3CViews descriptor

|  |  |  |  |
| --- | --- | --- | --- |
| **Elements and attributes for V3CViews descriptor** | **Use** | **Data type** | **Description** |
| **view** | 0..N | vpcc:viewType | An element whose elements and attributes specify the view information for the volumetric visual media and its associated V3C tracks. |
| **view**@viewId | M | xs:unsignedByte | This attribute specifies the view identifier |
| **view**@is\_basic\_View | O | xs:boolean | A flag indicating whether the view is selected as basic view. If not present, the default value is false. |
| **view.viewParameters** | O | vpcc:viewParameterType | An element whose attributes specify the projection used to generate a view representation from a 3D scene, including intrinsic and extrinsic parameters |
| **view**@v3cTrackIds | O | xs:StringVectorType | A list of space separated identifiers corresponding to the values of the track\_id for a number of V3C tracks which carry atlas data for the view. |
| Legend:  For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory.  For elements: <minOccurs>..<maxOccurs> (N=unbounded)  Elements are **bold**; attributes are non-bold and preceded with an @. | | | |

**9.5.3 Dynamic Views**

When the views are dynamic, a timed-metadata track for signalling each view information in the presentation timeline shall be carried in a separate **AdaptationSet** with a single representation and associated (linked) with the main V3C track using the @associationId attribute, defined in ISO/IEC 23009-1 [MPEG-DASH], with a @associationType value that includes the 4CC ‘cdcs’ for the corresponding **AdaptationSet** or **Representation**.

# Support for partial access utilizing spatial scalability (m54354)

## General

In the latest version of ISO/IEC 23090-5, V-PCC patches support a feature that enables subsampling a patch across different dimensions before coding its associated information. This concept is referred to as the level of detail (LoD) patch mode. The syntax of the LoD patch mode is as follows (yellow highlighted):

|  |  |
| --- | --- |
| patch\_data\_unit( patchIdx ) { | **Descriptor** |
| **pdu\_2d\_pos\_x**[ patchIdx ] | ue(v) |
| **pdu\_2d\_pos\_y**[ patchIdx ] | ue(v) |
| **pdu\_2d\_size\_x\_minus1**[ patchIdx ] | ue(v) |
| **pdu\_2d\_size\_y\_minus1**[ patchIdx ] | ue(v) |
| **pdu\_3d\_pos\_x**[ patchIdx ] | u(v) |
| **pdu\_3d\_pos\_y**[ patchIdx ] | u(v) |
| **pdu\_3d\_pos\_min\_z**[ patchIdx ] | u(v) |
| if( asps\_normal\_axis\_max\_delta\_value\_enabled\_flag ) |  |
| **pdu\_3d\_pos\_delta\_max\_z**[ patchIdx ] | u(v) |
| **pdu\_projection\_id**[ patchIdx ] | u(v) |
| **pdu\_orientation\_index**[ patchIdx ] | u(v) |
| if( afps\_lod\_mode\_enabled\_flag ) { |  |
| **pdu\_lod\_enabled\_flag**[ patchIndex ] | u(1) |
| if( pdu\_lod\_enabled\_flag[ patchIndex ] > 0 ) { |  |
| **pdu\_lod\_scale\_x\_minus1**[ patchIndex ] | ue(v) |
| **pdu\_lod\_scale\_y\_idc**[ patchIndex ] | ue(v) |
| } |  |
| } | u(v) |
| if( asps\_point\_local\_reconstruction\_enabled\_flag ) |  |
| point\_local\_reconstruction\_data( patchIdx ) | |  |
| if( asps\_miv\_extension\_flag ) | |  |
| pdu\_miv\_extension( patchIdx ) /\* Specified in ISO/IEC 23090-12\*/ | |  |
| } |  |

The above syntax permits different scaling for the horizontal and vertical axis (i.e. pdu\_lod\_scale\_x\_minus1 and pdu\_lod\_scale\_y\_idc), and the bigger scaling factor value is, the sparser point cloud is reconstructed. Figure 1 shows overview of the LoD patch mode process for (a) before encoding and (b) after decoding. In (a) before encoding, distributed sparse points are down scaled into small patch. In (b) after decoding, the small patch is upscaled as according to signalled scaling factors, and then reconstructed as sparse point cloud.

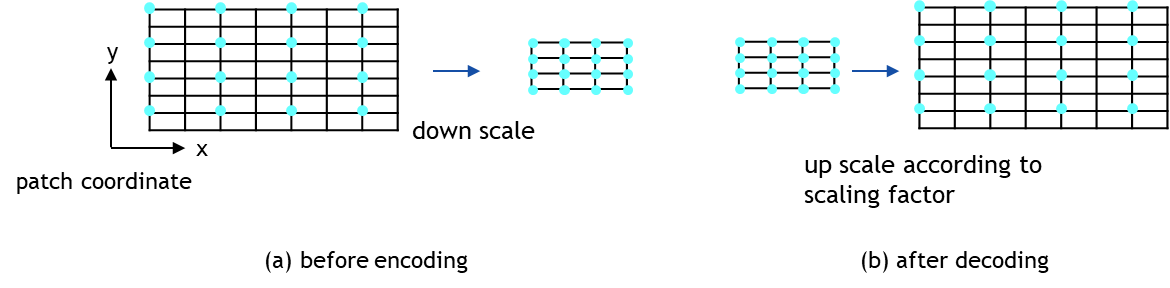


Figure 1: Overview of the LoD patch mode process

As described in ISO/IEC 23090-5, in order to achieve the spatial scalability, patches could be separated into atlas tiles while also using the LoD patch mode. The point cloud could be partitioned in an "LoD" partitioning where we could group together patches that correspond to a particular sampling grid into one atlas tile, while patches that correspond to different sampling grids can be placed to their own appropriate atlas tile. An example is shown in Figure 2. In this case subsampling of the point cloud has been achieved through uniform subsampling of the point cloud across the tangent and bi-tangent axis of each patch.

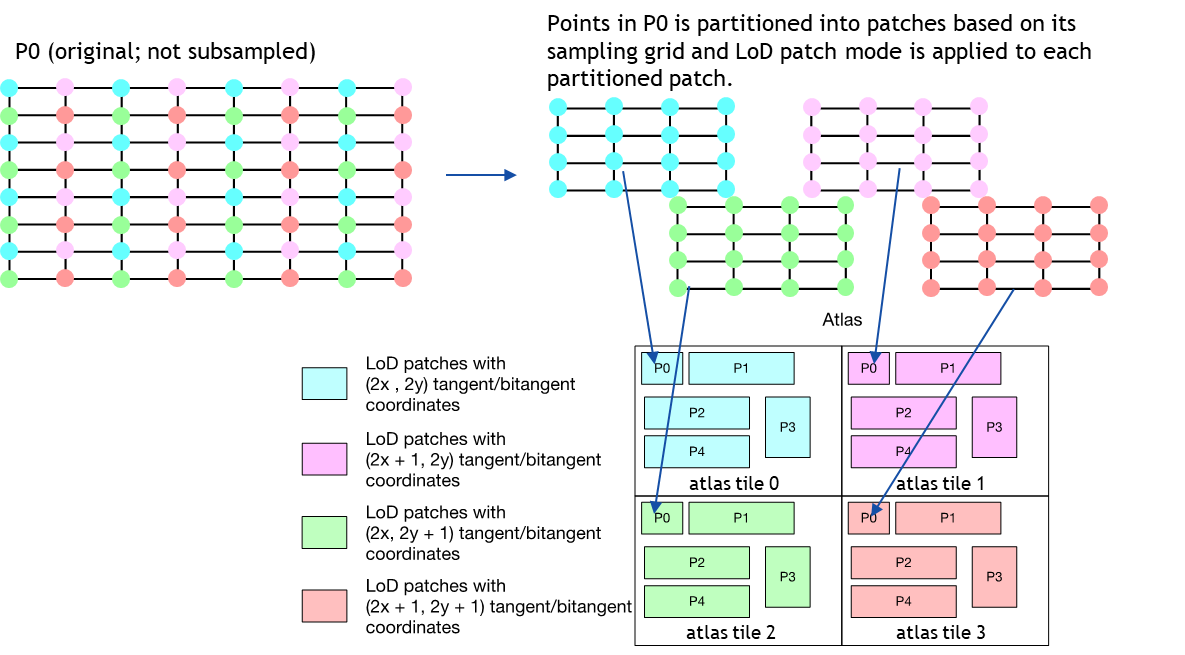


Figure 2: An example of “LoD” partitioning

Using this partitioning, a decoder may decide to only decode atlas tile 0. Since in this example partitioning each tile contains an almost identical set of patches, but in each atlas tile is with quarter of the projected points (because of the use of the LoD patch mode), the decoding and reconstruction process will result in only quarter of the compressed point cloud data being reconstructed. Therefore, the entire point cloud shape will be reconstructed but at a reduced resolution. Decoding and reconstructing additional atlas tiles could be seen as a form of spatial scalability for V-PCC compressed data.

If the client could access these atlas tiles with this partitioning individually, then the client could adaptively acquire and decode point cloud with appropriate LoD considering network bandwidth, client’s capabilities and so on. Moreover, by using this partial access with region-based partial access that has already been supported in Carriage of V3C data DIS, more effective partial access could be achieved e.g. the client could acquire point cloud subset with high LoD if that is placed close to the viewpoint, and with low LoD if that is placed far from the viewpoint, from one entire point cloud.

Based on the consideration, it is proposed to specify metadata that enables the client to identify atlas tiles that “LoD” partitioning is applied, such that the client could partially access to point cloud with appropriate, desired LoD.

In this proposal, following two types of indication are added to V3CSpatialRegionsBox as ISOBMFF extension, and V3C3DRegions descriptor as DASH MPD extension, respectively.

1. The grouping information of spatial regions that form spatial scalability each other.
2. The patch scaling information that enables the client to identify LoD of each spatial region.

### Extension on V3CSpatialRegionsBox

It is proposed to define SpatialScalabilityInfoStruct, which provides the grouping information of spatial regions and the patch scaling information, and to include it in V3CSpatialRegionsBox.

The proposed syntax is as follows, where the updated portion is highlighted in yellow.

aligned(8) class SpatialScalabilityInfoStruct() {

unsigned int(8) group\_id;

unsigned int(16) lod\_scale\_x;

unsigned int(16) lod\_scale\_y;

}

aligned(8) class V3CSpatialRegionsBox extends FullBox('vpsr',0,0) {

unsigned int(16) num\_regions;

for (i=0; i < num\_regions; i++) {

3DSpatialRegionStruct(1);

unsigned int(1) spatial\_scalability\_flag;

bit(7) reserved = 0;

if (spatial\_scalability\_flag)

SpatialScalabilityInforStruct();

unsigned int(8) num\_track\_groups;

for (j=0; j < num\_track\_groups; j++) {

unsigned int(32) track\_group\_id;

unsigned int(16) nal\_group\_id;

}

unsigned int(8) num\_objects;

unsigned int(8) obj\_idx\_length;

for (j=0; j < num\_objects; j++)

unsigned int(obj\_idx\_length \* 8) soi\_object\_idx;

}

}

}

In the above syntax, group\_id provides the grouping information of spatial regions that form spatial scalability each other. Regions with same group\_id forms spatial scalability. For example, if two of such regions are decoded, then reconstructed point cloud becomes higher LoD than when one of such regions is decoded.

The fields lod\_scale\_x and lod\_scale\_y provides the patch scaling information. These fields are set to be equal to the maximum LoD scaling factor value for the local x and y coordinate of a patch (i.e pdu\_lod\_scale\_x\_minus1 and pdu\_lod\_scale\_y\_idc in ISO/IEC 23090-5) indicated among all patches belong to this spatial region, respectively.

For cases that spatial scalability information is dynamically changed, it is also proposed to include SpatialScalabilityInfoStruct in V3CVoumetricMetadataSample.

Figure 3 shows an example of file structure and spatial scalability information indication. In this example, each atlas tile corresponds to quarter of points in original entire point cloud data while with same 3D spatial regions as the original entire point cloud.

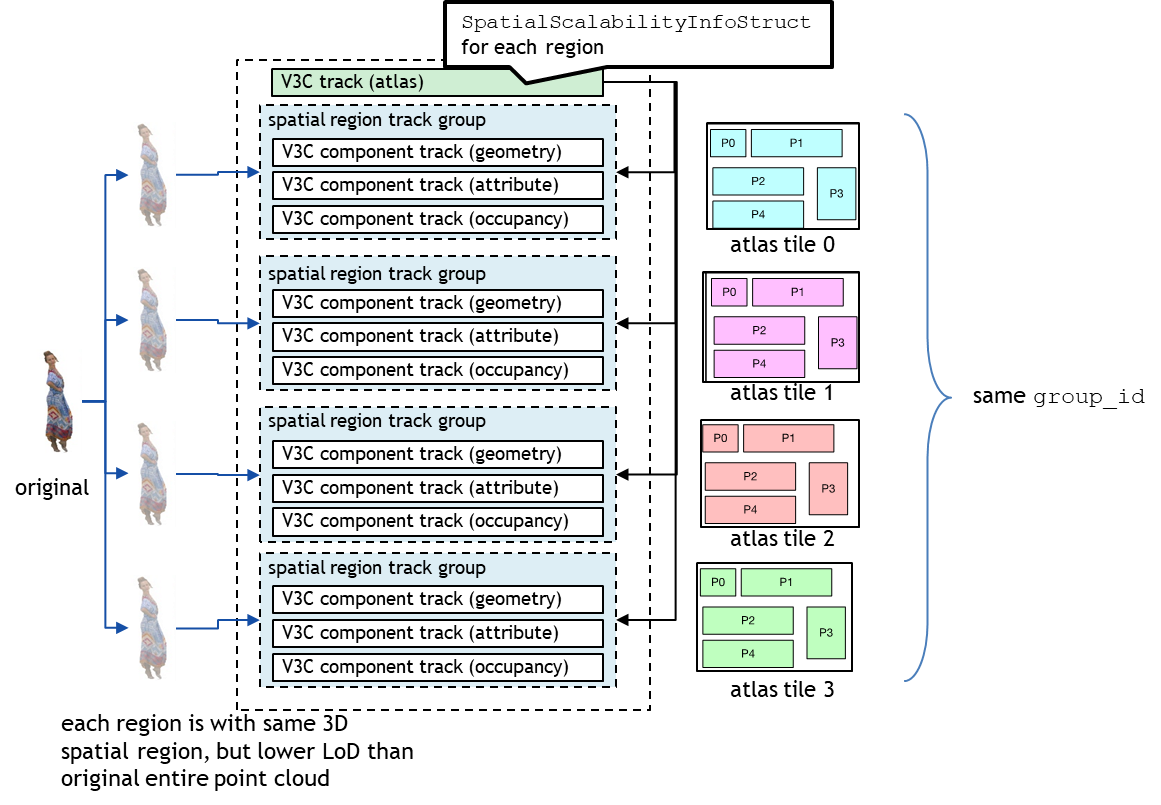


Figure 3: An example of the file structure and spatial scalability information indication

### Extension on V3C3DRegions descriptor

It is also proposed to add **spatialScalabilityInfo** element, which provides the grouping information of spatial regions and the patch scaling information, and include it in V3C3DRegions descriptor.

The proposed syntax and semantics are as follows where the updated portion is highlighted in yellow.

|  |  |  |  |
| --- | --- | --- | --- |
| **Elements and attributes for V3C3DRegions descriptor** | **Use** | **Data type** | **Description** |
| **vpsr** | 0..1 | v3c:spatialRegionMapType | Container element whose attributes and elements specify a mapping between a 3D spatial region and V3C tiles. |
| **vpsr.spatialRegion** | 1..N | v3c:spatialRegionType | An element whose attributes define a 3D spatial region and provide a mapping between the defined region and a number of V3C tiles. |
| **vpsr.spatialRegion**@id | M | xs:unsignedShort | An identifier for the 3D spatial region.  The value of this attribute shall match the value of the 3d\_region\_id field signalled for the corresponding region in the ISOBMFF container. |
| **vspr.spatialRegion**@type | OD | xs:unsignedByte | An attribute whose value indicates the type of the spatial region. Value 0 indicates a cuboid region. The remaining values are reserved.  If not present, the default value is 0. |
| **vpsr.spatialRegion**@x | M | xs:int | The x-coordinate of the reference point for the bounding box defining the spatial region. |
| **vpsr.spatialRegion**@y | M | xs:int | The y-coordinate of the reference point for the bounding box defining the spatial region. |
| **vpsr.spatialRegion**@z | M | xs:int | The z-coordinate of the reference point for the bounding box defining the spatial region. |
| **vspr.spatialRegion.cuboid** | CM | v3c:spatialRegionCuboidType | An element specifying a cuboid extending from the reference point of the spatial region. This element shall be present only when the **spatialRegion**@type attribute is set to 0. |
| **vpsr.spatialRegion.cuboid**@dx | M | xs:int | The length of the bounding box along the x-axis (i.e., width). Negative values indicate a length that extends in the negative direction of the axis. |
| **vpsr.spatialRegion.cuboid**@dy | M | xs:int | The length of the bounding box along the y-axis (i.e., height). Negative values indicate a length that extends in the negative direction of the axis. |
| **vpsr.spatialRegion.cuboid**@dz | M | xs:int | The length of the bounding box along the z-axis (i.e., depth). Negative values indicate a length that extends in the negative direction of the axis. |
| **vpsr.spatialRegion.spatialScalabilityInfo** | O | v3c:spatialScalabilityInfoType | An element whose attributes provide spatial scalability related information that is associated with the 3D spatial region. If present, there shall be patches with LoD parameters in atlases that corresponds to this spatial region. |
| **vpsr.spatialRegion.spatialScalabilityInfo**@groupId | M | xs:unsignedByte | An identifier for the group that forms spatial scalability. The value of this attribute shall match the value of the group\_id field signaled for the corresponding region in the ISOBMFF container. |
| **vpsr.spatialRegion.spatialScalabilityInfo**@lodScaleX | M | xs:unsignedShort | The maximum value of the LoD scaling factor for the local x coordinate of a patch indicated among all patches belong to this spatial region. The value of this attribute shall match the value of the lod\_scale\_x field signaled for the corresponding region in the ISOBMFF container. |
| **vpsr.spatialRegion.spatialScalabilityInfo**@lodScaleY | M | xs:unsignedShort | The maximum value of the LoD scaling factor for the local y coordinate of a patch indicated among all patches belong to this spatial region. The value of this attribute shall match the value of the lod\_scale\_y field signaled for the corresponding region in the ISOBMFF container. |
| **…** |  |  |  |

## Proposed Specification Text

The updated portion is highlighted in yellow.

**8.x Spatial scalability information structure**

**8.x.1 Definition**

SpatialScalabilityInfoStruct provides grouping information of spatial regions that form spatial scalability each other and patch scaling information of each spatial region.

**8.x.2 Syntax**

aligned(8) class SpatialScalabilityInfoStruct() {  
 unsigned int(8) group\_id;  
 unsigned int(16) lod\_scale\_x;  
 unsigned int(16) lod\_scale\_y;  
}

**8.x.3 Semantics**

group\_id is an identifier for the group that forms spatial scalability.

lod\_scale\_x and lod\_scale\_y indicate the maximum values of the LoD scaling factor for the local x and y coordinate of a patch (i.e pdu\_lod\_scale\_x\_minus1 and pdu\_lod\_scale\_y\_idc in ISO/IEC 23090-5) indicated among all patches belong to this spatial region, respectively.

**8.4 Static spatial region information box**

**8.4.1 Definition**

…

**8.4.2 Syntax**

Box Types: 'vpsr'  
Container: V3CSampleEntry ('v3c1' or 'v3cg')  
Mandatory: No  
Quantity: Zero or one

aligned(8) class V3CSpatialRegionsBox extends FullBox('vpsr',0,0) {  
 unsigned int(16) num\_regions;  
 for (i=0; i < num\_regions; i++) {  
 3DSpatialRegionStruct(1);  
 unsigned int(1) spatial\_scalability\_flag;  
 bit(7) reserved = 0;  
 if (spatial\_scalability\_flag)  
 SpatialScalabilityInfoStruct();  
 unsigned int(8) num\_track\_groups;  
 for (j=0; j < num\_track\_groups; j++) {  
 unsigned int(32) track\_group\_id;  
 unsigned int(16) nal\_group\_id;  
 }  
 unsigned int(8) num\_objects;  
 unsigned int(8) obj\_idx\_length;  
 for (j=0; j < num\_objects; j++)  
 unsigned int(obj\_idx\_length \* 8) soi\_object\_idx;  
 }  
 }  
}

**8.4.3 Semantics**

num\_regions indicates the number of 3D spatial regions in the volumetric media.

spatial\_scalability\_flag is a flag that indicates whether the spatial scalability information is signalled. When spatial\_scalability\_flag is set equal to 1, there shall be patches with LoD parameters in atlases that corresponds to this spatial region.

num\_track\_groups indicates the number of track groups associated with a 3D spatial region.

…

**8.5 Dynamic spatial region information**

**8.5.1 General**

…

**8.5.2 Sample entry**

…

**8.5.3 Sample format**

**8.5.3.1 Syntax**

aligned(8) class V3CVoumetricMetadataSample() {  
 bit(1) region\_updates\_flag;  
 bit(1) object\_updates\_flag;  
 bit(6) reserved = 0;  
 if (region\_updates\_flag) {  
 unsigned int(16) num\_regions;  
 for (i=0; i < num\_regions; i++) {  
 3DSpatialRegionStruct(dimensions\_included\_flag);  
 unsigned int(1) spatial\_scalability\_flag;  
 bit(7) reserved = 0;  
 if (spatial\_scalability\_flag)  
 SpatialScalabilityInfoStruct();  
 }  
 }  
 if (object\_updates\_flag) {  
 …  
 }  
}

**8.5.3.2 Semantics**

region\_updates\_flag indicates whether the sample includes updates to 3D spatial regions or not.

object\_updates\_flag indicates whether the sample includes updates to volumetric media scene objects or not.

num\_regions indicates the number of 3D spatial regions signalled in the sample. This may not necessarily be equal to the total number of available regions. Only spatial regions whose position,~~and/or~~ dimensions and/or spatial scalability information are being updated are present in the sample.

3DSpatialRegionStruct() is defined in clause ‎8.1. If dimensions\_included\_flag is set to 0, this implies that the dimensions of the region have been previously signalled in a previous instance of a 3DSpatialRegionStruct with the same 3d\_region\_id, either in a previous sample or in the sample entry.

spatial\_scalability\_flag is a flag that indicates whether the spatial scalability information is signalled. When spatial\_scalability\_flag is set equal to 1, there shall be patches with LoD parameters in atlases that corresponds to this spatial region.

SpatialScalabilityInfoStruct()is defined in clause 8.x.

num\_obj\_updates number of volumetric media scene objects updated in the sample.

…

**9.5 Partial Access**

**9.5.1 Static Spatial Regions**

…

Table 3 Elements and attributes for the V3C3DRegions descriptor

|  |  |  |  |
| --- | --- | --- | --- |
| **Elements and attributes for V3C3DRegions descriptor** | **Use** | **Data type** | **Description** |
| **vpsr** | 0..1 | v3c:spatialRegionMapType | Container element whose attributes and elements specify a mapping between a 3D spatial region and V3C tiles. |
| **vpsr.spatialRegion** | 1..N | v3c:spatialRegionType | An element whose attributes define a 3D spatial region and provide a mapping between the defined region and a number of V3C tiles. |
| **vpsr.spatialRegion**@id | M | xs:unsignedShort | An identifier for the 3D spatial region.  The value of this attribute shall match the value of the 3d\_region\_id field signalled for the corresponding region in the ISOBMFF container. |
| **vspr.spatialRegion**@type | OD | xs:unsignedByte | An attribute whose value indicates the type of the spatial region. Value 0 indicates a cuboid region. The remaining values are reserved.  If not present, the default value is 0. |
| **vpsr.spatialRegion**@x | M | xs:int | The x-coordinate of the reference point for the bounding box defining the spatial region. |
| **vpsr.spatialRegion**@y | M | xs:int | The y-coordinate of the reference point for the bounding box defining the spatial region. |
| **vpsr.spatialRegion**@z | M | xs:int | The z-coordinate of the reference point for the bounding box defining the spatial region. |
| **vspr.spatialRegion.cuboid** | CM | v3c:spatialRegionCuboidType | An element specifying a cuboid extending from the reference point of the spatial region. This element shall be present only when the **spatialRegion**@type attribute is set to 0. |
| **vpsr.spatialRegion.cuboid**@dx | M | xs:int | The length of the bounding box along the x-axis (i.e., width). Negative values indicate a length that extends in the negative direction of the axis. |
| **vpsr.spatialRegion.cuboid**@dy | M | xs:int | The length of the bounding box along the y-axis (i.e., height). Negative values indicate a length that extends in the negative direction of the axis. |
| **vpsr.spatialRegion.cuboid**@dz | M | xs:int | The length of the bounding box along the z-axis (i.e., depth). Negative values indicate a length that extends in the negative direction of the axis. |
| **vpsr.spatialRegion.spatialScalabilityInfo** | O | v3c:spatialScalabilityInfoType | An element whose attributes provide spatial scalability related information that is associated with 3D spatial region. If present, there shall be patches with LoD parameters in atlases that corresponds to this spatial region. |
| **vpsr.spatialRegion.spatialScalabilityInfo**@groupId | M | xs:unsignedByte | An identifier for the group that forms spatial scalability. The value of this attribute shall match the value of the group\_id field signaled for the corresponding region in the ISOBMFF container. |
| **vpsr.spatialRegion.spatialScalabilityInfo**@lodScaleX | M | xs:unsignedShort | The maximum value of the LoD scaling factor for the local x coordinate of a patch indicated among all patches belong to this spatial region. The value of this attribute shall match the value of the lod\_scale\_x field signaled for the corresponding region in the ISOBMFF container. |
| **vpsr.spatialRegion.spatialScalabilityInfo**@lodScaleY | M | xs:unsignedShort | The maximum value of the LoD scaling factor for the local y coordinate of a patch indicated among all patches belong to this spatial region. The value of this attribute shall match the value of the lod\_scale\_y field signaled for the corresponding region in the ISOBMFF container. |
| **vpsr.spatialRegion**@asIds | CM | StringVectorType | A list of space separated identifiers corresponding to the values of the @id attribute for **AdaptationSet**s and/or **Representation**s mapped to this region.  This attribute shall be absent in the case of single-track encapsulation of the V3C content. |
| **vspr.spatialRegion**@trackGroupIds | O | StringVectorType | A list of space separated identifiers corresponding to the values of the track\_group\_id for a number of V3C component track groups.  This attribute shall be absent if V3C components are stored as a single-track. |
| **Legend:**  For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory.  For elements: <minOccurs>..<maxOccurs> (N=unbounded)  Elements are **bold**; attributes are non-bold and preceded with an @. | | | |

The data types for the various elements and attributes of the **V3C3DRegions** descriptor shall be as defined in the following XML schema.

<?xml version="1.0" encoding="UTF-8"?>   
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"   
 targetNamespace="urn:mpeg:mpegI:v3c:2020"   
 xmlns:v3c="urn:mpeg:mpegI:v3c:2020"   
 elementFormDefault="qualified">   
  
 <xs:element name="vpsr" type="v3c:spatialRegionMapType" />   
  
 <xs:complexType name="spatialRegionMapType">   
 <xs:element name="spatialRegion" type="v3c:spatialRegionType" minOccurs="1"/>   
 </xs:complexType>   
  
 <xs:complexType name="spatialRegionType">   
 <xs:attribute name="id" type="xs:unsignedShort" use="required" />   
 <xs:attribute name="type" type="xs:unsignedByte" use="optional" default="0" />   
 <xs:attribute name="x" type="xs:int" use="required" />   
 <xs:attribute name="y" type="xs:int" use="required" />   
 <xs:attribute name="z" type="xs:int" use="required" />   
 <xs:attribute name="asIds" type="StringVectorType" use="required" />   
  
 <xs:element name="cuboid" type="v3c:spatialRegionCuboidType" minOccurs="0" maxOccurs="1"/>   
 <xs:element name="spatialScalabilityInfo" type="v3c:spatialScalabilityInfoType" minOccurs="0" maxOccurs="1"/>   
 </xs:complexType>   
  
 <xs:complexType name="spatialRegionCuboidType">   
 <xs:attribute name="dx" type="xs:int" use="required" />   
 <xs:attribute name="dy" type="xs:int" use="required" />   
 <xs:attribute name="dz" type="xs:int" use="required" />   
 </xs:complexType>   
  
 <xs:complexType name="spatialScalabilityInfoType">   
 <xs:attribute name="groupId" type="xs:int" use="required" />   
 <xs:attribute name="lodScaleX" type="xs:int" use="required" />   
 <xs:attribute name="lodScaleY" type="xs:int" use="required" />   
 </xs:complexType>   
  
</xs:schema>

# Visibility cone information indication in system layer (m54355)

## Background

Visibility cone information specifies the cone shaped volume which is the region providing possible good view for the associated object. Figure 2 shows the overview of visibility cone.



Figure 2: Overview of visibility cone. The black arrow represents the direction vector.

The syntax is as follows. The shape of visibility cone is specified by the direction vector and the angle along the direction vector.

|  |  |
| --- | --- |
| scene\_object\_information( payloadSize ) { |  |
| … |  |
| if ( soi\_num\_object\_updates > 0 ) { |  |
| **…** |  |
| for( i = 0; i  <=  soi\_num\_object\_updates; i++ ) { |  |
| **soi\_object\_idx**[ i ] | u(v) |
| … |  |
| if (!soi\_object\_cancel\_flag[ k ]) { |  |
| … |  |
| if( soi\_visibility\_cones\_present\_flag ) { |  |
| **soi\_visibility\_cones \_update\_flag**[ k ] | u(1) |
| if( soi\_visibility\_cones\_update\_flag[ k ]) { |  |
| **soi\_direction\_x**[ k ] | fl(32) |
| **soi\_direction\_y**[ k ] | fl(32) |
| **soi\_direction\_z**[ k ] | fl(32) |
| **soi\_angle**[ k ] | fl(16) |
| } |  |
| } |  |
| } |  |
| } |  |
| } |  |
| } |  |

When focusing attention to specific part of the point cloud (i.e. viewpoint is brought close to the point cloud), it is possible to use 3D spatial region information to cull point cloud subsets that are not in the viewing frustum. However, subdivision of the point cloud based on 3D spatial region information alone is hardly helpful when viewing the whole point cloud, because all the point cloud subsets fall in the viewing frustum and thus no point cloud subsets are culled.

In such cases, the visibility cone information is useful for culling of occluded surfaces. Only the surfaces on the front side of the point cloud is typically rendered, therefore this enables to render the same view with a lot less information even when viewing the whole point cloud.

Considering the hierarchical processing model, minimum information needs to be indicated at system-layer so that the file is efficiently structured for partial access. Since the visibility cone information also provides essential information for performing partial access efficiently as described above, this information should be regarded as one of the minimum information that needs to be indicated at system-layer.

Based on the consideration, it is proposed to specify metadata at system-layer that provides visibility cone information for each 3D spatial region. The visibility cone information at system-layer is indicated as coarse information that could be calculated from visibility cone information, which is indicated in object scene information SEI, for objects included in the associated 3D spatial region. Figure 3 shows an overview of the structure based on hierarchical processing model with proposed visibility cone information.

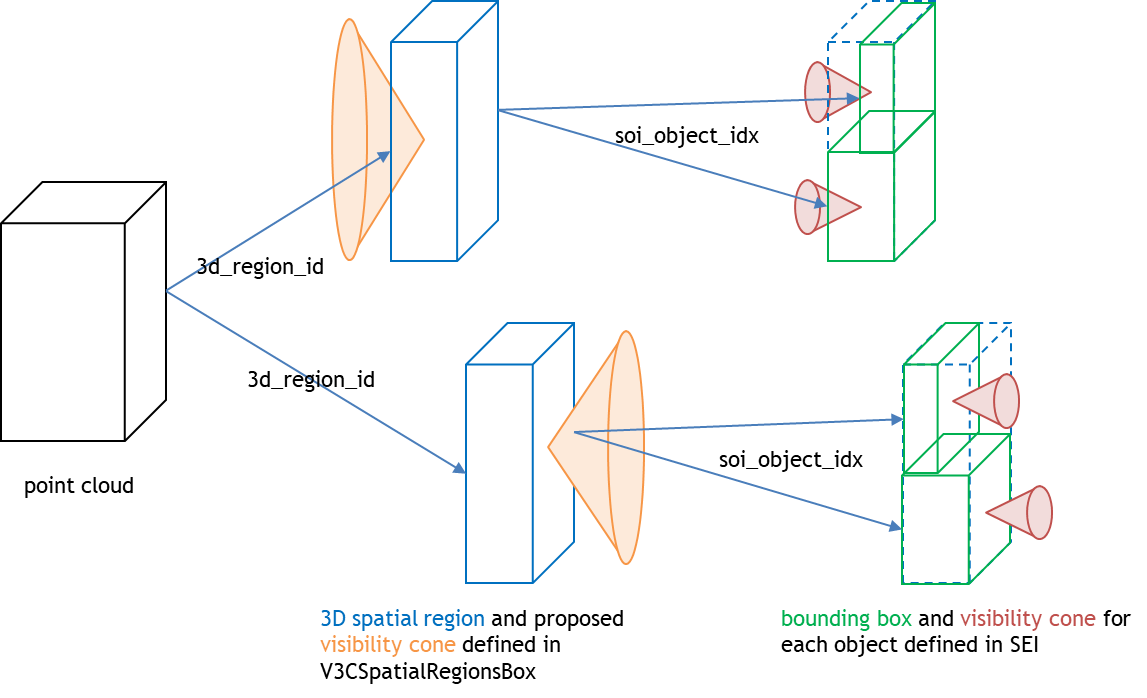


Figure 3: An overview of the structure with visibility cone information

### Extension on V3CSpatialRegionsBox

It is proposed to define VisibilityConeInfoStruct, which provides the visibility cone information of associated spatial region, and include it in V3CSpatialRegionsBox.

The proposed syntax is as follows where the updated portion is highlighted in yellow.

aligned(8) class VisibilityConeInfoStruct() {

unsigned int(32) direction\_x;

unsigned int(32) direction\_y;

unsigned int(32) direction\_z;

unsigned int(16) angle;

}

aligned(8) class V3CSpatialRegionsBox extends FullBox('vpsr',0,0) {

unsigned int(16) num\_regions;

for (i=0; i < num\_regions; i++) {

3DSpatialRegionStruct(1);

unsigned int(1) visibility\_cone\_info\_flag;

bit(7) reserved = 0;

if (visibility\_cone\_info\_flag)

VisilityConeInfoStruct();

unsigned int(8) num\_track\_groups;

for (j=0; j < num\_track\_groups; j++) {

unsigned int(32) track\_group\_id;

unsigned int(16) nal\_group\_id;

}

unsigned int(8) num\_objects;

unsigned int(8) obj\_idx\_length;

for (j=0; j < num\_objects; j++)

unsigned int(obj\_idx\_length \* 8) soi\_object\_idx;

}

}

}

In the above syntax, direction\_x, direction\_y and direction\_z indicate the normalized x-, y- and z-component value of the direction vector for the visibility cone of an associated spatial region. For these fields, the value 0 maps to -1.0 and 232-1 maps to 1.0. angle indicates the angle of the visibility cone along the direction vector in degrees.

For cases that visibility cone information is dynamically changed, it is also proposed to include VisibilityConeInfoStruct in V3CVoumetricMetadataSample.

### Extension on V3C3DRegions descriptor

It is also proposed to add **visibilityConeInfo** element which provides the visibility cone information of associated spatial region in V3C3DRegions descriptor.

The proposed syntax and semantics are as follows where the updated portion is highlighted in yellow.

|  |  |  |  |
| --- | --- | --- | --- |
| **Elements and attributes for V3C3DRegions descriptor** | **Use** | **Data type** | **Description** |
| **vpsr** | 0..1 | v3c:spatialRegionMapType | Container element whose attributes and elements specify a mapping between a 3D spatial region and V3C tiles. |
| **vpsr.spatialRegion** | 1..N | v3c:spatialRegionType | An element whose attributes define a 3D spatial region and provide a mapping between the defined region and a number of V3C tiles. |
| **vpsr.spatialRegion**@id | M | xs:unsignedShort | An identifier for the 3D spatial region.  The value of this attribute shall match the value of the 3d\_region\_id field signalled for the corresponding region in the ISOBMFF container. |
| **vspr.spatialRegion**@type | OD | xs:unsignedByte | An attribute whose value indicates the type of the spatial region. Value 0 indicates a cuboid region. The remaining values are reserved.  If not present, the default value is 0. |
| **vpsr.spatialRegion**@x | M | xs:int | The x-coordinate of the reference point for the bounding box defining the spatial region. |
| **vpsr.spatialRegion**@y | M | xs:int | The y-coordinate of the reference point for the bounding box defining the spatial region. |
| **vpsr.spatialRegion**@z | M | xs:int | The z-coordinate of the reference point for the bounding box defining the spatial region. |
| **vspr.spatialRegion.cuboid** | CM | v3c:spatialRegionCuboidType | An element specifying a cuboid extending from the reference point of the spatial region. This element shall be present only when the **spatialRegion**@type attribute is set to 0. |
| **vpsr.spatialRegion.cuboid**@dx | M | xs:int | The length of the bounding box along the x-axis (i.e., width). Negative values indicate a length that extends in the negative direction of the axis. |
| **vpsr.spatialRegion.cuboid**@dy | M | xs:int | The length of the bounding box along the y-axis (i.e., height). Negative values indicate a length that extends in the negative direction of the axis. |
| **vpsr.spatialRegion.cuboid**@dz | M | xs:int | The length of the bounding box along the z-axis (i.e., depth). Negative values indicate a length that extends in the negative direction of the axis. |
| **vpsr.spatialRegion.visibilityConeInfo** | O | v3c:visibilityConeInfoType | An element whose attributes provide visibility cone information that is associated with 3D spatial region. |
| **vpsr.spatialRegion.visibilityConeInfo**@directionX | M | xs:unsignedInt | The normalized x-component value of the direction vector for the visibility cone of an associated spatial region. The value 0 maps to -1.0 and 232-1 maps to 1.0. The value of this attribute shall match the value of the direction\_x field signaled for the corresponding region in the ISOBMFF container. |
| **vpsr.spatialRegion.visibilityConeInfo**@directionY | M | xs:unsignedInt | The normalized y-component value of the direction vector for the visibility cone of an associated spatial region. The value 0 maps to -1.0 and 232-1 maps to 1.0. The value of this attribute shall match the value of the direction\_y field signaled for the corresponding region in the ISOBMFF container. |
| **vpsr.spatialRegion.visibilityConeInfo**@directionZ | M | xs:unsignedInt | The normalized z-component value of the direction vector for the visibility cone of an associated spatial region. The value 0 maps to -1.0 and 232-1 maps to 1.0. The value of this attribute shall match the value of the direction\_z field signaled for the corresponding region in the ISOBMFF container. |
| **vpsr.spatialRegion.visibilityConeInfo**@angle | M | xs:unsignedShort | The angle value of the visibility cone along the direction vector in degrees. |
| **…** |  |  |  |

## Proposed Specification Text

The updated portion is highlighted in yellow.

**8.x Visibility cone information structure**

**8.x.1 Definition**

VisibilityConeInfoStruct provides visibility cone information of each spatial region.

**8.x.2 Syntax**

aligned(8) class VisibilityConeInfoStruct() {  
 unsigned int(32) direction\_x;  
 unsigned int(32) direction\_y;  
 unsigned int(32) direction\_z;  
 unsigned int(16) angle;  
}

**8.x.3 Semantics**

direction\_x, direction\_y and direction\_z indicate the normalized x-, y- and z-component value of the direction vector for the visibility cone of an associated spatial region. For these fields, the value 0 maps to -1.0 and 232-1 maps to 1.0.

angle indicates the angle of the visibility cone along the direction vector in degrees.

**8.4 Static spatial region information box**

**8.4.1 Definition**

…

**8.4.2 Syntax**

Box Types: 'vpsr'  
Container: V3CSampleEntry ('v3c1' or 'v3cg')  
Mandatory: No  
Quantity: Zero or one

aligned(8) class V3CSpatialRegionsBox extends FullBox('vpsr',0,0) {  
 unsigned int(16) num\_regions;  
 for (i=0; i < num\_regions; i++) {  
 3DSpatialRegionStruct(1);  
 unsigned int(1) visibility\_cone\_info\_flag;  
 bit(7) reserved = 0;  
 if (visibility\_cone\_info\_flag)  
 VisibilityConeInfoStruct();  
 unsigned int(8) num\_track\_groups;  
 for (j=0; j < num\_track\_groups; j++) {  
 unsigned int(32) track\_group\_id;  
 unsigned int(16) nal\_group\_id;  
 }  
 unsigned int(8) num\_objects;  
 unsigned int(8) obj\_idx\_length;  
 for (j=0; j < num\_objects; j++)  
 unsigned int(obj\_idx\_length \* 8) soi\_object\_idx;  
 }  
 }  
}

**8.4.3 Semantics**

num\_regions indicates the number of 3D spatial regions in the volumetric media.

visibility\_cone\_info\_flag is a flag that indicates whether the visibility cone information is signalled.

num\_track\_groups indicates the number of track groups associated with a 3D spatial region.

…

**8.5 Dynamic spatial region information**

**8.5.1 General**

…

**8.5.2 Sample entry**

…

**8.5.3 Sample format**

**8.5.3.1 Syntax**

aligned(8) class V3CVoumetricMetadataSample() {  
 bit(1) region\_updates\_flag;  
 bit(1) object\_updates\_flag;  
 bit(6) reserved = 0;  
 if (region\_updates\_flag) {  
 unsigned int(16) num\_regions;  
 for (i=0; i < num\_regions; i++) {  
 3DSpatialRegionStruct(dimensions\_included\_flag);  
 unsigned int(1) visibility\_cone\_info\_flag;  
 bit(7) reserved = 0;  
 if (visibility\_cone\_info\_flag)  
 VisibilityConeInfoStruct();  
 }  
 }  
 if (object\_updates\_flag) {  
 …  
 }  
}

**8.5.3.2 Semantics**

region\_updates\_flag indicates whether the sample includes updates to 3D spatial regions or not.

object\_updates\_flag indicates whether the sample includes updates to volumetric media scene objects or not.

num\_regions indicates the number of 3D spatial regions signalled in the sample. This may not necessarily be equal to the total number of available regions. Only spatial regions whose position,~~and/or~~ dimensions and/or visibility cone information are being updated are present in the sample.

3DSpatialRegionStruct() is defined in clause ‎8.1. If dimensions\_included\_flag is set to 0, this implies that the dimensions of the region have been previously signalled in a previous instance of a 3DSpatialRegionStruct with the same 3d\_region\_id, either in a previous sample or in the sample entry.

visibility\_cone\_info\_flag is a flag that indicates whether the visibility cone information is signalled.

VisibilityConeInfoStruct()is defined in clause 8.x.

num\_obj\_updates number of volumetric media scene objects updated in the sample.

…

**9.5 Partial Access**

**9.5.1 Static Spatial Regions**

…

Table 3 Elements and attributes for the V3C3DRegions descriptor

|  |  |  |  |
| --- | --- | --- | --- |
| **Elements and attributes for V3C3DRegions descriptor** | **Use** | **Data type** | **Description** |
| **vpsr** | 0..1 | v3c:spatialRegionMapType | Container element whose attributes and elements specify a mapping between a 3D spatial region and V3C tiles. |
| **vpsr.spatialRegion** | 1..N | v3c:spatialRegionType | An element whose attributes define a 3D spatial region and provide a mapping between the defined region and a number of V3C tiles. |
| **vpsr.spatialRegion**@id | M | xs:unsignedShort | An identifier for the 3D spatial region.  The value of this attribute shall match the value of the 3d\_region\_id field signalled for the corresponding region in the ISOBMFF container. |
| **vspr.spatialRegion**@type | OD | xs:unsignedByte | An attribute whose value indicates the type of the spatial region. Value 0 indicates a cuboid region. The remaining values are reserved.  If not present, the default value is 0. |
| **vpsr.spatialRegion**@x | M | xs:int | The x-coordinate of the reference point for the bounding box defining the spatial region. |
| **vpsr.spatialRegion**@y | M | xs:int | The y-coordinate of the reference point for the bounding box defining the spatial region. |
| **vpsr.spatialRegion**@z | M | xs:int | The z-coordinate of the reference point for the bounding box defining the spatial region. |
| **vspr.spatialRegion.cuboid** | CM | v3c:spatialRegionCuboidType | An element specifying a cuboid extending from the reference point of the spatial region. This element shall be present only when the **spatialRegion**@type attribute is set to 0. |
| **vpsr.spatialRegion.cuboid**@dx | M | xs:int | The length of the bounding box along the x-axis (i.e., width). Negative values indicate a length that extends in the negative direction of the axis. |
| **vpsr.spatialRegion.cuboid**@dy | M | xs:int | The length of the bounding box along the y-axis (i.e., height). Negative values indicate a length that extends in the negative direction of the axis. |
| **vpsr.spatialRegion.cuboid**@dz | M | xs:int | The length of the bounding box along the z-axis (i.e., depth). Negative values indicate a length that extends in the negative direction of the axis. |
| **vpsr.spatialRegion.visibilityConeInfo** | O | v3c:visibilityConeInfoType | An element whose attributes provide visibility cone information that is associated with 3D spatial region. |
| **vpsr.spatialRegion.visibilityConeInfo**@directionX | M | xs:unsignedInt | The normalized x-component value of the direction vector for the visibility cone of an associated spatial region. The value 0 maps to -1.0 and 232-1 maps to 1.0. The value of this attribute shall match the value of the direction\_x field signaled for the corresponding region in the ISOBMFF container. |
| **vpsr.spatialRegion.visibilityConeInfo**@directionY | M | xs:unsignedInt | The normalized y-component value of the direction vector for the visibility cone of an associated spatial region. The value 0 maps to -1.0 and 232-1 maps to 1.0. The value of this attribute shall match the value of the direction\_y field signaled for the corresponding region in the ISOBMFF container. |
| **vpsr.spatialRegion.visibilityConeInfo**@directionZ | M | xs:unsignedInt | The normalized z-component value of the direction vector for the visibility cone of an associated spatial region. The value 0 maps to -1.0 and 232-1 maps to 1.0. The value of this attribute shall match the value of the direction\_z field signaled for the corresponding region in the ISOBMFF container. |
| **vpsr.spatialRegion.visibilityConeInfo**@angle | M | xs:unsignedShort | The angle value of the visibility cone along the direction vector in degrees. |
| **vpsr.spatialRegion**@asIds | CM | StringVectorType | A list of space separated identifiers corresponding to the values of the @id attribute for **AdaptationSet**s and/or **Representation**s mapped to this region.  This attribute shall be absent in the case of single-track encapsulation of the V3C content. |
| **vspr.spatialRegion**@trackGroupIds | O | StringVectorType | A list of space separated identifiers corresponding to the values of the track\_group\_id for a number of V3C component track groups.  This attribute shall be absent if V3C components are stored as a single-track. |
| **Legend:**  For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory.  For elements: <minOccurs>..<maxOccurs> (N=unbounded)  Elements are **bold**; attributes are non-bold and preceded with an @. | | | |

The data types for the various elements and attributes of the **V3C3DRegions** descriptor shall be as defined in the following XML schema.

<?xml version="1.0" encoding="UTF-8"?>   
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"   
 targetNamespace="urn:mpeg:mpegI:v3c:2020"   
 xmlns:v3c="urn:mpeg:mpegI:v3c:2020"   
 elementFormDefault="qualified">   
  
 <xs:element name="vpsr" type="v3c:spatialRegionMapType" />   
  
 <xs:complexType name="spatialRegionMapType">   
 <xs:element name="spatialRegion" type="v3c:spatialRegionType" minOccurs="1"/>   
 </xs:complexType>   
  
 <xs:complexType name="spatialRegionType">   
 <xs:attribute name="id" type="xs:unsignedShort" use="required" />   
 <xs:attribute name="type" type="xs:unsignedByte" use="optional" default="0" />   
 <xs:attribute name="x" type="xs:int" use="required" />   
 <xs:attribute name="y" type="xs:int" use="required" />   
 <xs:attribute name="z" type="xs:int" use="required" />   
 <xs:attribute name="asIds" type="StringVectorType" use="required" />   
  
 <xs:element name="cuboid" type="v3c:spatialRegionCuboidType" minOccurs="0" maxOccurs="1"/>   
 <xs:element name="visibilityConeInfo" type="v3c:visibilityConeInfoType" minOccurs="0" maxOccurs="1"/>   
 </xs:complexType>   
  
 <xs:complexType name="spatialRegionCuboidType">   
 <xs:attribute name="dx" type="xs:int" use="required" />   
 <xs:attribute name="dy" type="xs:int" use="required" />   
 <xs:attribute name="dz" type="xs:int" use="required" />   
 </xs:complexType>   
  
 <xs:complexType name="visibilityConeInfoType">   
 <xs:attribute name="directionX" type="xs:unsignedInt" use="required" />   
 <xs:attribute name="directionY" type="xs:unsignedInt" use="required" />   
 <xs:attribute name="directionZ" type="xs:unsignedInt" use="required" />   
 <xs:attribute name="angle" type="xs:unsignedShort" use="required" />   
 </xs:complexType>   
  
</xs:schema>

# Additional Timed Metadata for Dynamic Spatial Regions

## Introduction

For enabling partial access of V3C data, the latest version of ISO/IEC 23090-10 [36] resulting from MPEG#131 contains: (i) static spatial region information box V3CSpatialRegionsBox, and (ii) dynamic spatial region information timed metadata track ‘dyvm’. The latter is for the case when 3D spatial regions defined for the volumetric media stream carried by the V3C track are considered as dynamic regions, i.e., the spatial region information may dynamically change over time. The former is when the spatial region information does not change over time. Since a client will only be viewing only a portion of the volumetric content at any given time (corresponding to the content visible based on user’s viewport), such selective partial access is essential in order to reduce the bandwidth and latency required to stream volumetric content.

The timed metadata track ‘dyvm’ for dynamic spatial information signaling contains two flags for region and object updates, and if true signals the relevant updates. If the flag for region updates is true, the new information is signaled on updated spatial regions and their mapping to tile component track groups, atlas tiles, and atlas tile tracks. If the flag for object updates is true, the new information is signaled on updated objects and corresponding object characteristics such as bounding box, spatial region mapping, priority, dependencies and cancellations.

As another object-related characteristic for dynamic spatial regions, object motion tracking can be used. In particular, a streaming client player can use motion tracking of the object(s) to determine which spatial regions or tiles of the volumetric content needs to be updated per frame in relation to the user’s viewport and fetch content from the server accordingly. For example, depending on the motion vector of an object over a specific spatial region in relation to the user’s viewport, a given spatial region may be fetched in higher or lower quality, resolution or level of detail (LoD), e.g., say if an object over a given spatial region outside of the user’s viewport is predicted to be inside of the user’s viewport based on its motion vector, it can be fetched in higher quality. Also, for the objects inside the client’s viewport, the player may fetch these objects at different quality levels, resolutions, or level of detail (LoD) depending on the relative distance of the object to the client’s viewport, which can again be assessed based on the object’s motion vector.

In this contribution, we propose the signaling of object motion vector information as part of the ‘dyvm’ timed metadata track for dynamic spatial regions.

## Proposed Updates to ‘dyvm’ Timed Metadata Track

## Dynamic spatial region information

### General

If the V3C track has an associated timed-metadata track with a sample entry type 'dyvm', 3D spatial regions defined for the volumetric media stream carried by the V3C track are considered as dynamic regions (i.e., the spatial region information may dynamically change over time). If objects are added or removed in the middle of the bitstream by scene object information SEI messages and a V3CSpatialRegionsBox is present in the sample entry then VolumetricMetadataSample shall be present.

The associated timed-metadata track shall contain a 'cdsc' track reference to the V3C track carrying the atlas stream.

### Sample entry

aligned(8) class DynamicVolumetricMetadataSampleEntry extends MetaDataSampleEntry('dyvm') {  
 V3CSpatialRegionsBox();  
}

### Sample format

### Syntax

aligned(8) class V3CVolumetricMetadataSample() {  
 bit(1) region\_updates\_flag;  
 bit(1) object\_updates\_flag;  
 bit(6) reserved = 0;  
 if (region\_updates\_flag) {  
 unsigned int(16) num\_regions;  
 for (i=0; i < num\_regions; i++) {  
 3DSpatialRegionStruct(dimensions\_included\_flag);  
 bit(1) update\_mapping\_flag;  
 bit(7) reserved = 0;  
 if (update\_mapping\_flag) {

if (all\_tiles\_in\_single\_track\_flag) {

unsigned int(8) num\_track\_groups;  
 for (j=0; j < num\_track\_groups; j++) {  
 unsigned int(32) component\_track\_group\_id;   
 unsigned int(8) num\_tiles;  
 for (k=0; k < num\_tiles; k++) {  
 unsigned int(16) tile\_id;  
 }  
 }  
 } else {  
 unsigned int(8) num\_tile\_tracks;  
 for (int j=0; j < num\_tile\_tracks; j++) {  
 unsigned int(32) tile\_track\_id;  
 }  
 }  
 }  
 }  
 }  
 if (object\_updates\_flag) {  
 unsigned int(8) num\_obj\_updates;  
 for (i=0; i<num\_obj\_updates; i++) {  
 unsigned int(8) obj\_idx\_length;  
 unsigned int(obj\_idx\_length \* 8) soi\_object\_idx;  
 bit(3) reserved = 0;  
 bit(1) obj\_spatial\_region\_mapping\_flag;  
 bit(1) obj\_dependencies\_present\_flag;  
 bit(1) obj\_bounding\_box\_present\_flag;  
 bit(1) obj\_priority\_update\_flag;

bit(1) obj\_motion\_vector\_present\_flag;  
 if (obj\_bounding\_box\_present\_flag) {  
 3DPoint();  
 3DBoundingBoxStruct();   
 }  
 if (obj\_spatial\_region\_mapping\_flag) {  
 // mapping object to spatial regions  
 unsigned int(8) obj\_num\_spatial\_regions;  
 for (j=0; j < obj\_num\_spatial\_regions; j++) {  
 obj\_region\_id[j];  
 }  
 } else {  
 // mapping object to tiles  
 unsigned int(8) obj\_num\_tiles;  
 for (j=0; j < obj\_num\_tiles; j++) {  
 unsigned int(6) obj\_atlas\_id[j];  
 bit(2) reserved = 0;  
 unsigned int(16) obj\_tile\_id[j];  
 }  
 }  
 if (obj\_priority\_update\_flag) {  
 unsigned int(4) obj\_priority\_value;  
 bit(4) reserved = 0;  
 }  
 if (obj\_dependencies\_present\_flag) {  
 unsigned int(8) obj\_num\_dependencies[soi\_object\_idx];  
 for (j=0; j < obj\_num\_depedendencies; j++) {  
 unsigned int(8) obj\_dep\_idx\_length[j];  
 unsigned int(obj\_dep\_idx\_length[j] \* 8) soi\_object\_idx[j];  
 }  
 }

if (obj\_motion\_vector\_flag) {  
 unsigned int(8) obj\_motion\_x;  
 unsigned int(8) obj\_motion\_y;  
 unsigned int(8) obj\_motion\_z;  
 }

}  
 unsigned int(8) num\_obj\_cancelled;  
 for (i=0; i< num\_obj\_cancelled; i++) {  
 unsigned int(8) obj\_idx\_length;  
 unsigned int(obj\_idx\_length \* 8) soi\_object\_idx;  
 }  
 }  
}

### Semantics

region\_updates\_flag indicates whether the sample includes updates to 3D spatial regions or not.

object\_updates\_flag indicates whether the sample includes updates to volumetric media scene objects or not.

num\_regions indicates the number of 3D spatial regions signalled in the sample. This may not necessarily be equal to the total number of available regions. Only spatial regions whose position and/or dimensions are being updated are present in the sample.

update\_mapping\_flag is a flag indicating whether the the mapping between the spatial region and atlas tiles is updated. A value of 1 indicates that an updated mapping is present. Value 0 indicates no updates.

num\_track\_groups indicates the number of tile component track groups associated with a 3D spatial region.

component\_track\_group\_id identifies the tile component track group for the tracks which carry the V3C components for the tile(s) associated with the 3D spatial region.

num\_tiles indicates the number of atlas tiles associated with an atlas tile component track group.

tile\_id identifies the atlas NAL units containing atlas tile information for the associated tile component track group. The value of tile\_id shall be equal to the groupID in one of the entries of NALUMapEntry in the V3C track carrying the atlas information.

num\_tile\_tracks indicates the number of tile tracks associated with the spatial region.

tile\_track\_id is an id of an atlas tile track associated with the 3D spatial region.

3DPoint() and 3DBoundingBoxStruct() indicate the 3D bounding box information of an object in the object update list of the sample.

num\_obj\_updates number of volumetric media scene objects updated in the sample.

obj\_idx\_length is the length of the object index, in number of bytes, for an object in the signalled object list.

soi\_object\_idx indicates the value of an object index, as defined by the object scene information SEI message.

obj\_spatial\_region\_mapping\_flag indicates whether a mapping to a spatial region is signalled for an object in the object update list of the sample. Value 0 indicates that the object is mapped to tile track groups. Value 1 indicates that the object is mapped to a 3D spatial region.

obj\_depdendencies\_present\_flag indicates whether object dependency information is available for an object in the object update list of the sample or not. Value 0 indicates that the object does not depend on other objects. Value 1 indicates that the object depends on one or more objects within the volumetric media scene.

obj\_bounding\_box\_present\_flag indicates whether 3D bounding boxing information is available for an object in the object update list of the sample or not. Value 0 indicates that no bounding box information is given. Value 1 indicates that that 3D bounding box information for the object are present.

obj\_priority\_update\_flag indicates whether priority information is available for an object in the object update list of the sample or not. Value 0 indicates that no object priority information is given. Value 1 indicates that object priority information is present.

obj\_motion\_vector\_present\_flag indicates whether motion vector information is available for an object in the object update list of the sample or not. Value 0 indicates that no object motion vector information is given. Value 1 indicates that object motion vector information is present.

obj\_num\_spatial\_regions is the number of 3D spatial regions that an object in the object update list of the sample is associated with.

obj\_region\_id[j] is the id of the j-th spatial region that an object in the object update list of the sample is associated with.

obj\_num\_tiles indicates the number of tiles associated with the object.

obj\_atlas\_id[j] is the id of the atlas to which the j-th tile associated with the object belongs.

obj\_tile\_id[j] identifies the j-th atlas tile associated with the object.

obj\_priority\_value indicates the priority value of an object in the object update list of the sample. The lower the priority value, the higher the priority.

obj\_num\_depedencies is the number of objects that an object in the object update list of the sample depends on.

obj\_dep\_index\_length[j] is the length, in number of bytes, of the index of the j-th object that an object in the object update list of the sample depends on.

obj\_index[j] is the index of the j-th object that an object in the object update list of the sample depends on.

num\_obj\_cancelled indicates the number of cancelled objects signalled in this sample.

obj\_motion\_x, obj\_motion\_y, and obj\_motion\_z specify the x, y, and z coordinate values, respectively, of the motion vector of the object in the Cartesian coordinate system.

## Proposal

We propose to include the above updates in Section 2 in ISO/IEC 23090-10.

# Comments on signaling of static and dynamic regions for V3CD

## Introduction

In the latest draft text of V3CD FIDS [1], static and dynamic region information is signalled using separate data structures, as shown in Appendix. Not only are these data structures different and incompatible, requiring different processing procedures, but rather lengthy and complicated, not consistent with simple design considerations for 2D static and dynamic regions with types ‘2dcc’.

A ‘2dcc’ region is defined in a way that is independent of whether it is static or dynamic. A 2D region being static or dynamic is signalled using where its information data structure is placed with a track:

1. A region is *static*, if its information data structure or identifier is signalled in the sample entry of a media track.
2. A region is *static*, if its information data structure or identifier is signalled in the sample entry of a timed metadata track, as this region applies to all samples.
3. A region is *dynamic*, if its information data structure or identifier is signalled in a sample of a timed metadata track.

This contribution proposes to use these design considerations as a way to simplify signalling for V3CD static and dynamic regions.

## Proposal

Specifically, this contribution proposes to use the following principle for signalling static and dynamic spatial regions for V3CD media content:

* define the spatial region information data structure, V3CSpatialRegionsBox, or *a simplified version of it*, independent of whether regions defined within are static or dynamic;
* signal static V3C spatial regions by placing their information data structure in a sample entry of a V3C media track;
* signal metadata of static V3C spatial regions by placing their information data structure in a sample entry of a V3C timed metadata track – as these regions are invariant across all samples; and
* signal dynamic V3C spatial regions by placing their information data structure in a sample of a V3C timed metadata track.

In this manner, (a) signalling of static regions in a V3C media track can simply put their information data structures in its sample entry, and (b) signalling of static and dynamic regions in a timed metadata track can be signalled as follows:

aligned(8) class DynamicVolumetricMetadataSampleEntry extends MetaDataSampleEntry('dyvm') {  
 V3CSpatialRegionsBox();  
}

aligned(8) class DynamicVolumetricMetadataSample extends MetaDataSample ('dyvm') {  
 V3CSpatialRegionsBox();  
}

If avoiding repeating same data structures in samples for spatial regions in a timed metadata track is desired, an improvement can use region indices in samples as follows:

1. in a sample entry, list all region information data structures and indices of those regions that are static or invariant in all samples, and
2. in a sample entry, list indices of regions that are not listed in the sample entry.

If the above design principle is acceptable by the group, detailed proposals on the syntax and semantics will be provided in some later contributions.

# On Defining Common Metadata for MPEG-I Immersive Media (m57821)

## Introduction

This contribution is in response to the following mandate of the AHG on Systems Technologies for Volumetric Media [1]:

3. Study commonalities among 23090-2, 23090-10 and 23090-18 and propose common metadata definition to be moved to 23090-7.

We like to point out that there had been some previous attempts in defining common metadata definitions for 23090-7. For a number of reasons, some proposed metadata definitions have been captured in MPEG-B Part 10 (Carriage of Timed Metadata in ISOBMFF) [2][6], since the 127th MPEG meeting in Gothenburg, Sweden.

Now, with the standard development progress of 23090-2, 23090-10 and 23090-18, this task of defining common metadata definitions for MPEG-I becomes more important than ever before. To understand this need, this contribution first shows, in Section 2, an example of metadata related to a viewing space/region and a bounding box in the existing documents, 23090-2 (FDIS, 2nd Edition) [3], 23090-10 (FDIS) [4], 23090-18 (Draft FDIS) [5], as well as 23001-10 (MPEG-B Part 10) TuC [2].

Given that the standard status of the specifications, 23090-2 (FDIS, 2nd Edition), 23090-10 (FDIS) and 23090-18 (Draft FDIS), it could be impractical to find commonalities of the metadata definitions across these standards and move them into 23090-7 in a couple of meeting cycles. This contribution, therefore, proposes a gradual approach for achieving the goal in a rather long run; namely,

1. documenting common or related metadata definitions in 23090-7;
2. defining unified metadata definitions in 23090-7; and
3. adopting unified metadata definitions in new editions of the MPEG-I and other MPEG standards.

## An Example of Common Metadata: Viewing Space/Region

### “Viewing Space” in 23090-2 [3]

aligned(8) class ViewingSpaceStruct() {  
 unsigned int(8) viewing\_space\_shape\_type;  
 unsigned int(16) distance\_scale;  
 bit(1) guard\_range\_flag;  
 bit(7) reserved;  
 if(viewing\_space\_shape\_type==0)  
 CuboidStruct(guard\_range\_flag);  
 else if(viewing\_space\_shape\_type==1)  
 SphereStruct(guard\_range\_flag);  
 else if(viewing\_space\_shape\_type==2)  
 CylinderStruct(guard\_range\_flag);   
 else if(viewing\_space\_shape\_type==3)  
 EllipsoidStruct(guard\_range\_flag);  
}

aligned(8) class CuboidStruct(guard\_range\_flag) {  
 signed int(32) x\_Min;  
 signed int(32) x\_Max;  
 signed int(32) y\_Min;  
 signed int(32) y\_Max;  
 signed int(32) z\_Min;  
 signed int(32) z\_Max;  
 if (guard\_range\_flag) {  
 unsigned int(8) guard\_range\_X;  
 unsigned int(8) guard\_range\_Y;  
 unsigned int(8) guard\_range\_Z;  
 }  
}

aligned(8) class SphereStruct(guard\_range\_flag) {  
 unsigned int (32) sphere\_radius;   
 if (guard\_range\_flag){   
 bit(1) reserved;  
 unsigned int(7) guard\_radius\_diff;  
 }  
}

aligned(8) class CylinderStruct(guard\_range\_flag) {  
 unsigned int (32) cylinder\_radius;  
 Point(0);  
 Point(1);  
 if (guard\_range\_flag)  
 unsigned int(8) cylinder\_guard\_radius\_diff;  
}

aligned(8) class Point(i) {  
 signed int(32) x\_pt[i];  
 signed int(32) y\_pt[i];  
 signed int(32) z\_pt[i];  
}

aligned(8) class EllipsoidStruct(guard\_range\_flag) {  
 unsigned int (32) length\_X;  
 unsigned int (32) length\_Y;  
 unsigned int (32) length\_Z;  
 if (guard\_range\_flag) {  
 unsigned int(8) guard\_lenghthX\_diff;  
 unsigned int(8) guard\_lenghthY\_diff;  
 unsigned int(8) guard\_lenghthZ\_diff;  
 }  
}

aligned(8) SphereRegionStruct(range\_included\_flag, interpolate\_included\_flag) {  
 signed int(32) centre\_azimuth;  
 signed int(32) centre\_elevation;  
 signed int(32) centre\_tilt;  
 if (range\_included\_flag) {  
 unsigned int(32) azimuth\_range;  
 unsigned int(32) elevation\_range;  
 }  
 if (interpolate\_included\_flag) {  
 unsigned int(1) interpolate;  
 bit(7) reserved = 0;  
 }  
}

### “Spatial region bounding box” and “Spatial Region” in 23090-10 [4]

aligned(8) class V3CBoundingBox (anchor\_included, scale\_included) {

if (anchor\_included) { // anchor is not 0,0,0

unsigned int(8) bb\_pos\_precision;

Vector3 bb\_position(bb\_pos\_precision);

}

if (scale\_included) {

unsigned int(8) bb\_scale\_precision;

Vector3 bb\_scale(bb\_scale\_precision);

}

}

aligned(8) class Vector3(int precision = 32) {

int reserved\_bits = 8 - (precision\*3) % 8;

if (reserved\_bits != 8) {

bit(reserved\_bits) reserved = 0;

}

unsigned int(precision) x;

unsigned int(precision) y;

unsigned int(precision) z;

}

aligned(8) class V3CSpatialRegion {

unsigned int(32) size;

unsigned int(16) region\_id;

unsigned int(1) bb\_anchor\_present\_flag;

unsigned int(1) bb\_scale\_present\_flag;

unsigned int(1) tile\_mapping\_present\_flag;

unsigned\_int(1) tm\_spatial\_scalability\_flag;

unsigned\_int(1) object\_collection\_present\_flag;

bit(3) reserved = 0;

if (bb\_anchor\_present\_flag || bb\_scale\_present\_flag) {

V3CBoundingBox bounding\_box(bb\_anchor\_present\_flag, bb\_scale\_present\_flag);

}

if (tile\_mapping\_present\_flag) {

TileMapping tile\_map(tm\_spatial\_scalability\_flag);

}

if (object\_collection\_present\_flag) {

ObjectCollection object\_collection;

}

}

### “Bounding box” and “Spatial Region” in 23090-18 [5]

rectangular cuboid in which the source point cloud frame is included.

aligned(8) class GPCCSpatialRegionStruct(dimension\_included) {  
 unsigned int(16) 3d\_region\_id;

unsinged int(16) anchor\_x;

unsinged int(16) anchor\_y;

unsinged int(16) anchor\_z;

if (dimension\_included)

{  
 unsinged int(16) region\_dx;

unsinged int(16) region\_dy;

unsinged int(16) region\_dz;  
}

}

|  |  |  |  |
| --- | --- | --- | --- |
| **gpsr.spatialRegion**@x | OD | xs:int | The x-coordinate of the reference point for the bounding box defining the spatial region.  If not present, the default value is 0. |
| **gpsr.spatialRegion**@y | OD | xs:int | The y-coordinate of the reference point for the bounding box defining the spatial region.  If not present, the default value is 0. |
| **gpsr.spatialRegion**@z | OD | xs:int | The z-coordinate of the reference point for the bounding box defining the spatial region.  If not present, the default value is 0. |
| **gpsr.spatialRegion**@dx | M | xs:int | The length of the bounding box along the x-axis (i.e., width). Negative values indicate a length that extends in the negative direction of the axis. |
| **gpsr.spatialRegion**@dy | M | xs:int | The length of the bounding box along the y-axis (i.e., height). Negative values indicate a length that extends in the negative direction of the axis. |
| **gpsr.spatialRegion**@dz | M | xs:int | The length of the bounding box along the z-axis (i.e., depth). Negative values indicate a length that extends in the negative direction of the axis. |

### “2DRange” and “3DRange” in 23001-10 [2]

aligned(8) class 2DRangeStruct(shape\_type) {  
 if (shape\_type == 0) { // 2D rectangle   
 unsigned int(32) range\_width;  
 unsigned int(32) range\_height;  
 }  
 if (shape\_type == 1) { // 2D circle  
 unsigned int(32) range\_radius;  
 }  
 // other values of shape\_type are reserved  
}

aligned(8) class 3DRangeStruct(shape\_type) {  
 2DRangeStruct(shape\_type); // including 2D shape types  
 if (shape\_type == 2) { // 3D tile  
 unsigned int(32) range\_width;  
 unsigned int(32) range\_height;  
 unsinged int(32) range\_depth;  
 }  
 if (shape\_type == 3) { // 3D spherical region  
 unsigned int(32) range\_width;  
 unsigned int(32) range\_height;  
 unsinged int(32) range\_depth;  
 }  
 if (shape\_type == 4) { // 3D sphere  
 unsigned int(32) range\_radius;  
 }  
 // other values of shape\_type are reserved  
}

## Proposal

As seen from the example in Section 2, it is rather impractical to find commonalities of metadata definitions from the standards, especially with respect to generality, aggregation, compactness conciseness and entanglements, and simply move them into 23090-7. Thus, we propose the following approach to define common metadata for (MPEG-I) immersive media, to be agreed upon first, before embarking the real definition work for improving the current DIS or creating a new edition of 23090-7:

1. documenting common or related metadata definitions from the standards in 23090-7, from an agreed collection of metadata about immersive media;
2. defining unified metadata definitions in 23090-7, according to a set of rules and principles; and
3. adopting the unified metadata definitions in new editions of the MPEG-I and other MPEG standards (e.g., MPEG-B Part 10).

Clearly, the first step is always achievable, and it will be useful at least in understanding correspondence and relationship of the metadata definitions in the existing standards. To start with, we suggest the following:

* Use the latest TuC of MPEG-B part 10 as a guidance for an agreed collection of metadata definitions to start with:
  + 2D/3D viewing spaces (or bounding boxes) of different shapes;
  + 2D/3D spatial regions of different shapes and their spatial relationships, with viewing spaces;
  + 2D/3D viewpoints and viewports (in terms of spatial regions); and
  + …
* Form a table for grouping and mapping related metadata definitions, both in data structures and file format boxes.

If the above approach is acceptable by the group, detailed proposals on step 1 will be provided in some later contributions.

## References

1. WG03N0279\_20533. “List of WG 03 AHGs established at the 4th meeting”. The 135th MPEG meeting. July 2021.
2. WG03N0286\_20540. “Technologies under Consideration for ISO/IEC 23001-10”. The 135th MPEG meeting. July 2021.
3. WG3N0072. “Text of ISO/IEC FDIS 23090-2 2nd edition OMAF”. The 132nd MPEG meeting. October 2020.
4. WG03N00241. “Text of ISO/IEC FDIS 23090-10 Carriage of Visual Volumetric Video-based Coding Data”. The 135th MPEG meeting. July 2021.
5. WG03N0242\_20304. “Draft text of ISO/IEC FDIS 23090-18 Carriage of Geometry-based Point Cloud Compression Data”. The 134th MPEG meeting. April 2021.
6. SC29WG11\_N18637. “Technologies under Consideration for ISO/IEC 23001-10”. The 127th MPEG meeting. July 2019.
7. WG03N00193\_20255. “Text of ISO/IEC DIS 23090-7 Metadata for Immersive Media”. The 134th MPEG meeting. April 2021.

# On harmonization of metadata for MPEG-I systems (m57815)

## Introduction

MPEG is working on several specifications that rely on 3D concepts. The specifications and their status is presented in the Table 1.

Table 1 - MPEG specifications and their status

|  |  |  |
| --- | --- | --- |
| **ISO/IEC Reference** | **Title** | **Status** |
| 23090-5 | Visual volumetric video-based coding (V3C) and video-based point cloud compression (V-PCC) | FDIS |
| 23090-7 | Metadata for Immersive Media | DIS |
| 23090-9 | Geometry-based Point Cloud Compression, | DIS |
| 23090-10 | Carriage of Visual Volumetric Video-based Coding Dat | FDIS |
| 23090-12 | MPEG Immersive Video | FDIS |
| 23090-18 | Carriage of Geometry-based Point Cloud Compression Data | FDIS (draft) |

Other MPEG documents dealing with 3D concepts could also benefit from the results of this analysis, e.g. OMAF.

Clause 2 of this document presents common observations from the analysis of the source specifications. Clause 3 describes how some of the commonalities could be harmonized.

## Common observations

This clause presents commonalities between specifications listed in Table 1.

### Coordinate systems

Several documents define coordinate system with slightly different semantics. Considering that understanding the coordinate system for each specification is essential, it would be beneficial to align the terminology between the specifications. Especially the reference coordinate system in ISO/IEC DIS 23090-7 seems particularly confusing as it is specific to a unit sphere, it would be preferrable to rename it as unit sphere coordinate system to avoid confusing it with other definitions.

Common reference coordinate system: 3D Cartesian coordinate system with the centre being (X, Y, Z) equal to (0, 0, 0), used as the reference coordinate system for all viewpoints within a viewpoint group – 23090-7

Cartesian co-ordinates: three scalars (x, y, z) with finite precision and dynamic range that indicate the location of a point relative to a fixed reference point – 23090-18

Cartesian coordinates: three scalars (x, y, z) with finite precision and dynamic range that indicate the location of a point relative to a fixed reference point (the origin) – 23090-5

Cartesian co-ordinates: three scalars (x, y, z) with finite precision and dynamic range that indicate the location of a point relative to a fixed reference point – 23090-9

Reference 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. 23090-7

ISO/IEC DIS 23090-7 also defines terms for global coordinate axes and local coordinate axes. 3D graphics generally use terms object space and world space, and optionally local space. World space is the coordinate system for the entire scene, with its origin at the centre of the scene. Object space is the coordinate system from an object’s point of view. The origin of object space is at the object’s pivot or anchor point. World and object space axes may be oriented differently.

Global coordinate axes: coordinate axes that are associated with audio, video, and images representing the same acquisition position and intended to be rendered together – 23090-7

Local coordinate axes: the coordinate axes obtained after applying rotation to the global coordinate axes – 23090-7

It might be beneficial to introduce concepts of object space and world space and describe the related transformations moving from one space to the other to improve the current definitions in ISO/IEC DIS 23090-7.

### Positions, offsets, dimensions, translation and scaling

All the surveyed specifications contain triplets of values describing positions, offsets, dimensions, scaling or translation. These commonly consist of x-, y- and z-components that are applied to the corresponding world or object space axes. E.g. as illustrated by GPCCSpatialRegionStruct in ISO/IEC 23090-18 below.

aligned(8) class GPCCSpatialRegionStruct(dimension\_included) {

unsigned int(16) 3d\_region\_id;

unsinged int(16) anchor\_x;

unsinged int(16) anchor\_y;

unsinged int(16) anchor\_z;

if (dimension\_included)

{

unsinged int(16) region\_dx;

unsinged int(16) region\_dy;

unsinged int(16) region\_dz;

}

}

ISO/IEC FDIS 23090-10 has taken a different approach and defined a common syntax structure that can be used to provide these triplets of data as illustrated below. The syntax structure allows to define precision for the x-, y- and z-components of the structure. Vector3 may then be used in every occurrence where these triplets are used to describe positions, offsets, dimensions translation or scaling.

aligned(8) class Vector3(int precision = 32) {

int reserved\_bits = 8 - (precision\*3) % 8;

if (reserved\_bits != 8) {

bit(reserved\_bits) reserved = 0;

}

unsigned int(precision) x;

unsigned int(precision) y;

unsigned int(precision) z;

}

Adoption of Vector3 would allow to rewrite the GPCCSpatialRegionStruct as follows:

aligned(8) class GPCCSpatialRegionStruct(dimension\_included) {

unsigned int(16) 3d\_region\_id;

Vector3 anchor(16);

if (dimension\_included)

{

Vector3 dimensions(16);

}

}

This would simplify the syntax structures and result in less specification text. Individual components of the vector may be addressed in semantics by referring to anchor.x or dimensions.z.

Specifications seem to use different type of values of describing position, some use unsigned integers, some use signed integers and, in some cases, even floating points are used. This could mean that at least three types of Vector3 need to be created, i.e. Vector3Float, Vector3UInt, Vector3Int. The decision of which structure to use should not be made arbitrarily. E.g. for normalized positions as described in ISO/IEC DIS 23090-9 Vector3UInt should be used. Whereas viewport related signaling may require use of Vector3Float.

Geometry coding related position: (x, y, z) co-ordinates of a point, where the values are normalized by the bounding box so that the values of the positions shall be equal to or greater than 0. – 23090-9

### Rotations and orientations

Specifications should clarify the difference between orientation and rotation. Orientation is the result after applying a rotation. These should not be used interchangeably. Different terminology for rotations and orientations exist in specifications. Some use quaternions (x,y,z,w), some use rotations with azimuth, elevation and tilt or yaw, pitch and roll.

23090-7 Clause 5.3.1.2 contains the following

aligned(8) class RotationStruct() {

signed int(32) rotation\_yaw;

signed int(32) rotation\_pitch;

signed int(32) rotation\_roll;

}

Conscious decision should be made between usage of rotation and orientation. E.g. orientation is idea for describing how an object should be oriented at given point in time, whereas rotation is ideal for describing how the object should rotate to reach said orientation. Furthermore, when a common syntax for rotation and orientation is specified, it should be used consistently between specification.

Rotation is always dependent on the order of applying individual rotations to an object. E.g. applying yaw, pitch and roll may lead in different orientation than applying pitch, roll and yaw. It would be therefore important to also specify the order of rotations in a shared document.

### Bounding box

Several documents describe bounding box, which seems to mostly mean the same thing.

3D bounding box: volume defined as a cuboid solid having six rectangular faces placed at right angles. 23090-5

Bounding box: rectangular cuboid in which the source point cloud frame is included – 230990-18

bounding box: rectangular cuboid in which the source point cloud frame is included. – 230990-9

The syntax for bounding box however seems to be somewhat different, but consist of the same components. V3CBoundingBox is defined in ISO/IEC FDIS 23090-10, GPCCSpatialRegionStruct in ISO/IEC FDIS 23090-18 contains essentially a bounding box in it and vui\_parameters() in ISO/IEC FDIS 23090-5 also define a bounding box.

aligned(8) class V3CBoundingBox (anchor\_included, scale\_included)

{

if (anchor\_included) { // anchor is not 0,0,0

unsigned int(8) bb\_pos\_precision;

Vector3 bb\_position(bb\_pos\_precision);

}

if (scale\_included) {

unsigned int(8) bb\_scale\_precision;

Vector3 bb\_scale(bb\_scale\_precision);

}

}

aligned(8) class GPCCSpatialRegionStruct(dimension\_included) {

unsigned int(16) 3d\_region\_id;

unsinged int(16) anchor\_x;

unsinged int(16) anchor\_y;

unsinged int(16) anchor\_z;

if (dimension\_included)

{

unsinged int(16) region\_dx;

unsinged int(16) region\_dy;

unsinged int(16) region\_dz;

}

|  |  |
| --- | --- |
| if( **vui\_display\_box\_info\_present\_flag**) { |  |
| for( d = 0; d < 3; d++ ) { |  |
| **vui\_display\_box\_origin**[ d ] | u(v) |
| **vui\_display\_box\_size**[ d ] | u(v) |
| } |  |
| } |  |

Common definition for a bounding box and consistent usage would help consolidating specifications and results in less specification text overall.

### Viewport

Many specifications seem to utilize viewport signalling for different purpose. The viewport is typically signalled using extrinsic and intrinsic camera parameters. Viewport related terminology exists in 23090-10 clause 10. 23090-5 Clause F.2.15, and 23090-12 Clause 8.2.1.6.2. There are also ongoing discussions in 23090-18. In addition 23090-7 contains a definition for viewport which seems quite specific to their use case.

viewport: region of omnidirectional image or video suitable for display and viewing by the user – 23090-7

#### Extrinsic camera parameters

Extrinsic camera parameters are commonly used to place a virtual camera in a scene. It describes the position and orientation of the camera in world space. E.g. 23090-10 describes it as follows:

Aligned(8) class ExtCameraInfo () {

unsigned int(8)[4] cam\_pos\_x;

unsigned int(8)[4] cam\_pos\_y;

unsigned int(8)[4] cam\_pos\_z;

signed int(32) cam\_quat\_x;

signed int(32) cam\_quat\_y;

signed int(32) cam\_quat\_z;

}

23090-12 has almost exactly same definition but different precision for the orientation.

|  |  |
| --- | --- |
| camera\_extrinsics( viewID ) { | **Descriptor** |
| **ce\_view\_pos\_x**[ viewID ] | fl(32) |
| **ce\_view\_pos\_y**[ viewID ] | fl(32) |
| **ce\_view\_pos\_z**[ viewID ] | fl(32) |
| **ce\_view\_quat\_x**[ viewID ] | i(16) |
| **ce\_view\_quat\_y**[ viewID ] | i(16) |
| **ce\_view\_quat\_z**[ viewID ] | i(16) |
| } |  |

23090-5 contains slightly more information as it is intended for dynamic signalling of the extrinsic information.

|  |  |
| --- | --- |
| viewport\_position( payloadSize ) { | **Descriptor** |
| **vp\_viewport\_id** | ue(v) |
| **vp\_camera\_parameters\_present\_flag** | u(1) |
| if( vp\_camera\_parameters\_present\_flag ) |  |
| **vp\_vcp\_camera\_id** | u(10) |
| **vp\_cancel\_flag** | u(1) |
| if( !vp\_cancel\_flag ) { |  |
| **vp\_persistence\_flag** | u(1) |
| for( d = 0 ; d < 3; d++) |  |
| **vp\_position**[d] | fl(32) |
| **vp\_rotation\_qx** | i(16) |
| **vp\_rotation\_qy** | i(16) |
| **vp\_rotation\_qz** | i(16) |
| **vp\_center\_view\_flag** | u(1) |
| if( !vp\_center\_view\_flag ) |  |
| **vp\_left\_view\_flag** | u(1) |
| } |  |
| } |  |

Defining the position of extrinsic parameters should be done in a way that allows the viewport to move in the world space, thus several specifications have opted for using floating points to define such data.

The rotation component requires slightly more thought. It seems that quaternions are preferred for this purpose, which makes sense as it describes the orientation of the virtual camera. However, if the intention is to describe how a camera rotates to the target orientation, a rotation could be considered equally as well. This could be useful for more compact description of camera rotation, e.g. consider what is required to describe 360° degree rotation around z-axis with quaternion or rotation.

#### Intrinsic camera parameters

Intrinsic camera parameters are combined with extrinsic parameters to create a viewport into the scene. Whereas extrinsic parameters are used to describe how camera is positioned and oriented in space. The intrinsic parameters defined how the 3D scene is projected into 2D plane, also called as viewport. There are a couple of different projection formats which are also reflected in the intrinsic parameter syntax structure.

Defined in 23090-10

aligned(8) class IntCameraInfo () {

unsigned int(10) camera\_id;

bit(3) reserved = 0;

unsigned int(3) camera\_type;

if (camera\_type == 0) {

signed int(32) erp\_horizontal\_fov;

signed int(32) erp\_vertical\_fov;

}

if (camera\_type == 1) {

signed int(32) perspective\_horizontal\_fov;

unsigned int(8)[4] perspective\_aspect\_ratio;

}

if (camera\_type == 2) {

unsigned int(8)[4] ortho\_aspect\_ratio;

unsigned int(8)[4] ortho\_horizontal\_size;

}

unsigned int(8)[4] clipping\_near\_plane;

unsigned int(8)[4] clipping\_far\_plane;

}

And In 23090-12

|  |  |
| --- | --- |
| camera\_intrinsics( viewID, mode ) { | **Descriptor** |
| **ci\_cam\_type**[ viewID ] | u(8) |
| **ci\_projection\_plane\_width\_minus1**[ viewID ] | u(16) |
| **ci\_projection\_plane\_height\_minus1**[ viewID ] | u(16) |
| if( ci\_cam\_type[ viewID ] == 0 ) { /\* equirectangular \*/ |  |
| **ci\_erp\_phi\_min**[ viewID ] | fl(32) |
| **ci\_erp\_phi\_max**[ viewID ] | fl(32) |
| **ci\_erp\_theta\_min**[ viewID ] | fl(32) |
| **ci\_erp\_theta\_max**[ viewID ] | fl(32) |
| } else if( ci\_cam\_type[ viewID ] == 1 ) { /\* perspective \*/ |  |
| **ci\_perspective\_focal\_hor**[ viewID ] | fl(32) |
| **ci\_perspective\_focal\_ver**[ viewID ] | fl(32) |
| **ci\_perspective\_principal\_point\_hor**[ viewID ] | fl(32) |
| **ci\_perspective\_principal\_point\_ver**[ viewID ] | fl(32) |
| } else if( ci\_cam\_type[viewID] == 2 ) { /\* orthographic \*/ |  |
| **ci\_ortho\_width**[ viewID ] | fl(32) |
| **ci\_ortho\_height**[ viewID ] | fl(32) |
| } |  |
| } |  |

And in 23090-5

|  |  |
| --- | --- |
| viewport\_camera\_parameters( payloadSize ) { | **Descriptor** |
| **vcp\_camera\_id** | u(10) |
| **vcp\_cancel\_flag** | u(1) |
| if( vcp\_camera\_id > 0 && !vcp\_cancel\_flag ) { |  |
| **vcp\_persistence\_flag** | u(1) |
| **vcp\_camera\_type** | u(3) |
| if( vcp\_camera\_type == 0 ) { /\* equirectangular \*/ |  |
| **vcp\_erp\_horizontal\_fov** | u(32) |
| **vcp\_erp\_vertical\_fov** | u(32) |
| } else if( vcp\_camera\_type == 1 ) { /\* perspective \*/ |  |
| **vcp\_perspective\_aspect\_ratio** | fl(32) |
| **vcp\_perspective\_horizontal\_fov** | u(32) |
| } else if( vcp\_camera\_type == 2 ) { /\* orthographic \*/ |  |
| **vcp\_ortho\_aspect\_ratio** | fl(32) |
| **vcp\_ortho\_horizontal\_size** | fl(32) |
| } |  |
| **vcp\_clipping\_near\_plane** | fl(32) |
| **vcp\_clipping\_far\_plane** | fl(32) |
| } |  |
| } |  |

#### Viewport

Combining the extrinsic and intrinsic parameters defines the viewport to the scene. Different specifications combine them differently. 23090-5 and 23090-12 use camera or view identifiers to combine them in different structures, and 23090-10 explicitly wraps them in the same structure as follows:

aligned(8) class ViewportInfo (ext\_camera\_flag, int\_camera\_flag) {

if (ext\_camera\_flag == 1) {

unsigned int(1) center\_view\_flag;

bit(6) reserved = 0;

if (center\_view\_flag == 0) {

unsigned int(1) left\_view\_flag;

} else {

bit(1) reserved = 0;

}

ExtCameraInfo extCamInfo();

}

if (int\_camera\_flag == 1) {

IntCameraInfo intCamInfo();

}

}

For defining recommended viewport it should be considered if syntax for signalling center-view or left-view is actually required. Viewport describes how the scene is projected into 2d space and does not depend on which eye the result is displayed. Applications that intend to render 3D content as stereo typically depend on the viewing device, which provide information about the viewer such as interpupillary distance or gaze orientation data. It is unclear how center-view and left-view would be used by an application.

### DASH Streaming

It may make sense to add DASH streaming related syntax structures in a common specification. E.g. 23090-10 and 23090-18 define identical v3sr.spatialRegion and gpsr.spatialRegion descriptors.

## Harmonization aspects

Proposed harmonizations were adopted in WD.

## References

[1] MDS20303, “Text of ISO/IEC FDIS 23090-10 Carriage of Visual Volumetric Video-based Coding Data”, August 2021.

[2] ISO/IEC, “Information technology — Coded representation of immersive media — Part 5: Visual volumetric video-based coding (V3C) and video-based point cloud compression (V-PCC)”ISO/IEC 23090-5:2021”, June 2021

[3] MDS20255, “Text of ISO/IEC DIS 23090-7 Metadata for Immersive Media”, May 2021.

[4] MDS19088, “Text of ISO/IEC CD 23090-9 Geometry-based Point Cloud Compression”, April 2020.

[5] MDS20001, “Text of ISO/IEC DIS 23090-12 MPEG Immersive Video”, January 2021.

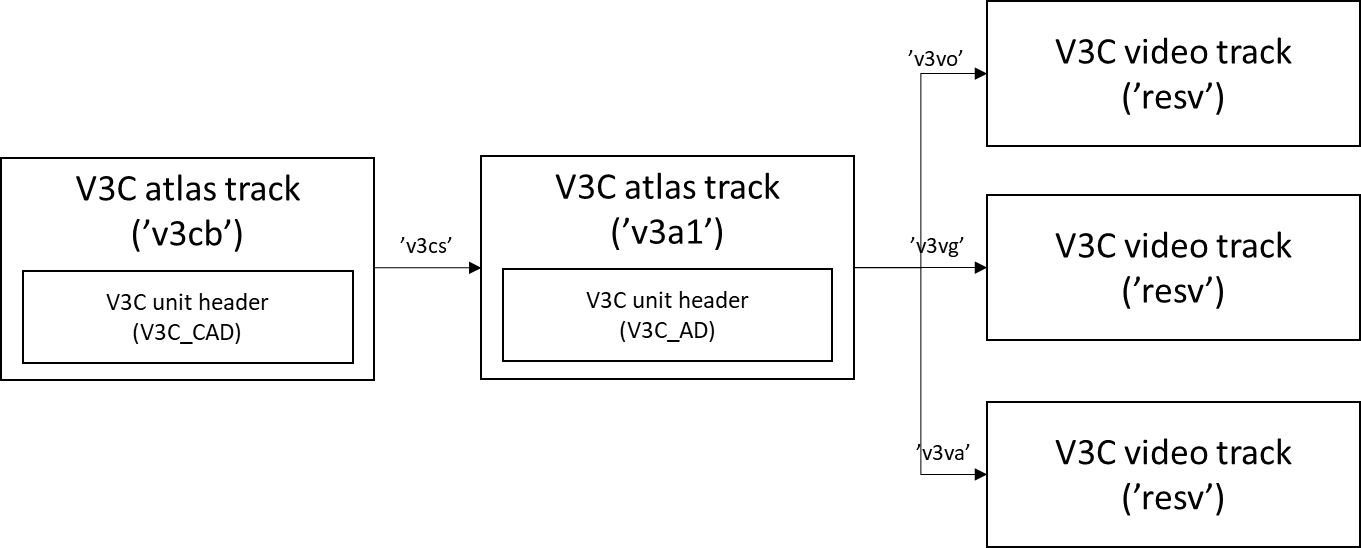
[6] MDS20304, “Draft text of ISO/IEC FDIS 23090-18 Carriage of Geometry-based Point Cloud Compression Data”, July 2021.

# Shared common atlas and atlas data track

ISO/IEC 23090-10 supports storage of multiple atlases and common atlas data. The authors argue in this contribution that the design for the storage is sub-optimal when only a single atlas is used in combination with the common atlas data, a typical MIV use case. Considering that ISO/IEC 23090-5 allows storing NAL units of common atlas data and atlas data in a single NAL unit stream, it should be considered if this functionality should be supported by the systems specification as well.

## Introduction

ISO/IEC 23090-10 [1] enables storage of V3C bitstream consisting of multiple atlases, which is the case for volumetric video compress according to ISO/IEC 23090-12. In which case, V3C bitstream contains one common atlas bitstream, identified through V3C unit header type V3C\_CAD, and one or more atlas bitstreams identified through V3C unit header type V3C\_AD. According to ISO/IEC 23090-10 each atlas bitstream is stored as samples of one or more V3C atlas tracks, where samples of each V3C atlas track can only contain data belonging to a single atlas, identified uniquely by atlas ID in V3C unit header. Common atlas bitstream is stored separately as its own track, which means that in ISOBMFF for MIV encoded content there will always be at least two atlas tracks for storing atlas data. One base V3C atlas track of type 'v3cb' for common atlas data and one V3C atlas track of type 'v3a1', or 'v3ag' for the atlas data. An example of the file format structure consisting of common atlas data track and atlas data track is shown below in Figure 1.



**Figure 1** **Current file format design of common atlas data and atlas data in two separate tracks**

We believe that mandating the storage of common atlas data and atlas data as separate tracks in ISOBMFF is sub-optimal in case when the V3C bitstream contains one common atlas bitstream and only one atlas bitstream. Requiring storage of common atlas bitstream and atlas bitstream in separate ISOBMFF tracks, creates overhead, for example by duplicating sample data related information for the atlas track and the common atlas track. It unnecessarily increases the parsing complexity and requires further synchronization of one additional track.

While 23090-5 [2] considers both atlas bitstream and common atlas bitstream as separate sub-bitstreams of the V3C bitstream, they both consist of NAL units that don’t have overlapping nal\_unit\_type values. This means that NAL units consisting of common atlas data and atlas data could be easily interleaved in a single NAL unit bitstream.

## Proposal

### General

A more optimal solution would be to allow storing common atlas data and atlas data in the same track, when only single atlas is present. This could be easily enabled by adding support for storing two V3C unit header boxes in the sample entry of the V3C atlas track. This is illustrated in Figure 2.

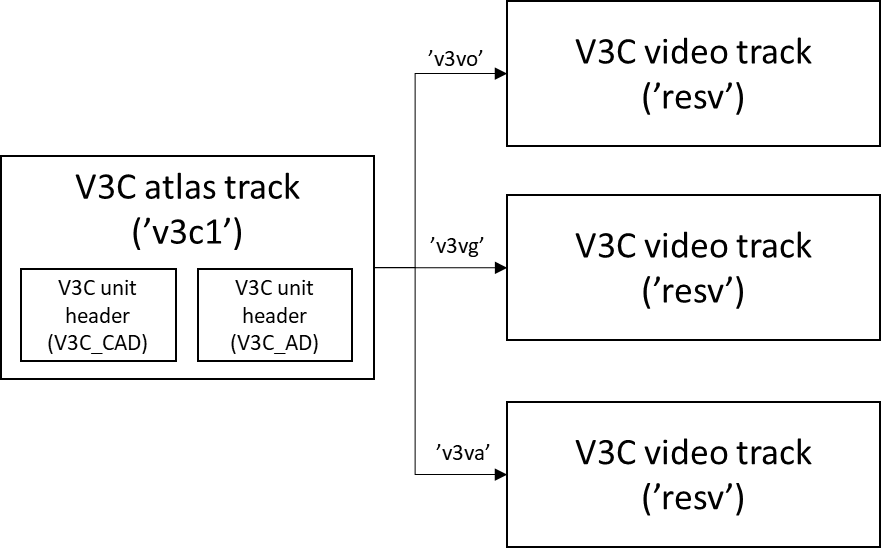


Figure 2 File format design allowing storage of two V3C unit headers in a track.

The introduction of common atlas data in MIV does not solely serve the purpose of having multiple atlases, but also intends to support applications which leverage the updated input camera models and still only encode information in a single atlas. To support such content, which the authors believe is a fairly common use case, the file format should not introduce further complexities for encapsulation.

Further semantic clarification should be added in the text of ISO/IEC 23090-10 to enable such encapsulation as well as removing some restrictions. Furthermore, it may be considered to define a dedicated 4CC sample entry code to indicate that a track contains both common atlas data and atlas data NAL units. However, the specification already defines several sample entry types, so if possible this should be avoided. It should be considered, if it is sufficient to simply allow storing two V3C unit header boxes (one with vuh\_unit\_type == V3C\_CAD and one with vuh\_unit\_type == V3C\_AD) in a sample entry of an atlas track and consider this as indication for interleaved common atlas and atlas NAL units.

The semantic and syntax related changes to clause 7.4.2 of ISO/IEC 23090-10 presented in the next subclause would enable the functionality discussed above.

### Specification Text

**7.4.2.1 Definition**

V3C atlas tracks use V3CAtlasSampleEntry which extends VolumetricVisualSampleEntry with a sample entry type of 'v3c1', 'v3cg', 'v3cb', 'v3a1', or 'v3ag'. Following restrictions are set for V3C atlas tracks:

* A V3C atlas track shall not carry ACL NAL units belonging to more than one atlas.
* A V3C atlas track sample entry contains a V3CConfigurationBox as defined in 7.2.2 and one or two V3CUnitHeaderBox(es)as defined in 7.2.3.
* When the sample entry contains only one V3CUnitHeaderBox, each sample in a V3C atlas track with sample entry of type 'v3c1' or 'v3cg', correspond to a single coded atlas access unit.
* When the sample entry contains two V3CUnitHeaderBoxes, each sample in a V3C atlas track with sample entry of type 'v3c1' or 'v3cg', correspond to a single coded atlas access unit and/or coded common atlas access unit
* Each sample in a V3C atlas track with sample entry of type 'v3a1' or 'v3ag', correspond to a single coded atlas access unit.
* Each sample in a V3C atlas track with sample entry of type 'v3cb', correspond to a single coded common atlas access unit.

NOTE – V3C atlas tracks with sample entry type 'v3cb', 'v3a1' and 'v3ag' are relevant for carrying multiple atlases.

**7.4.2.2 Syntax**

aligned(8) class V3CAtlasSampleEntry() extends VolumetricVisualSampleEntry (type) {

// type is 'v3c1', 'v3cg', 'v3cb ', 'v3a1', or 'v3ag'

V3CConfigurationBox config;

V3CUnitHeaderBox unit\_header;

V3CUnitHeaderBox cad\_unit\_header; // optional

}

**7.4.2.3 Semantics**

compressorname in the base class VolumetricVisualSampleEntry indicates the name of the compressor used with the value "\012V3C Coding" being recommended; the first byte is a count of the remaining bytes, here represented by \012, which (being octal 12) is 10 (decimal), the number of bytes in the rest of the string.

config contains a single instance of V3CConfigurationBox as defined in 7.2.2.

unit\_header contains a single instance of V3CUnitHeaderBox representing the atlas bitstream unit header as defined in 7.2.3. The vuh\_unit\_type in the V3CUnitHeaderBox equals V3C\_AD.

cad\_unit\_header contains a single instance of V3CUnitHeaderBox representing the common atlas bitstream unit header as defined in 7.2.3. The vuh\_unit\_type in the V3CUnitHeaderBox equals V3C\_CAD.

## Proposal & Decision

In #138 it was agreed, to put the contribution in TuC, so it could be added in the proper future output document.

## References

[1] MDS20303, “Text of ISO/IEC FDIS 23090-10 Carriage of Visual Volumetric Video-based Coding Data”, August 2021.

[2] MDS20761, “Text of ISO/IEC DIS 23090-5 Visual Volumetric Video-based Coding and Video-based Point Cloud Compression 2nd Edition”, July 2021.

# Timed and non-timed V3C components in ISOBMFF

## Introduction

ISO/IEC 23090-10 specifies how to encapsulate V3C bitstream into timed ISOBMFF structure (i.e., tracks) and into non-timed ISOBMFF structure (i.e., items). Those mechanisms are viable for most scenarios of V-PCC and MIV applications of V3C. However, use cases where V3C bitstream is composed of non-timed and timed components and stored as both tracks and items were not considered so far.

This contribution describes scenarios where such mixed storage needs to be considered, as well as proposes possible solution to support the described scenarios.

## Scenarios

The following scenarios based on existing applications of V3C (V-PCC, MIV) requiring mixing of timed and non-timed were identified:

* A use case where atlas data is static for the duration of the volumetric video, and it is stored as item while geometry and attribute components are changing over time and are stored as tracks. The use case could be for simplified MIV encoding, where cameras are fixed in the capture scene and don’t move during the sequence.
* A use case where atlas data and geometry are static for the duration of the volumetric video and are stored as items while attribute components are changing over time and are stored as tracks. The use case could be for example for placing animated texture for a fixed geometry object, like a screen on the TV.

The same scenarios could be also considered for the future application of V3C bitstream such as V-DMC currently under development in WG07, where an animated mesh scenario it is quite often considered to have a fixed texture for an animated geometry.

## Specification Consideration

ISO/IEC 23090-10 provides the information on how to store V3C bitstream as number of tracks or items and how to indicate the relations between the tracks or between the items. That means which tracks and items contain which V3C components to allow parser to properly extract V3C bitstream. However, the specification does not define how to signal dependency between items and tracks that contain data originated from the same V3C bitstream. Considering that the entry point to V3C bitstream stored in ISOBMFF is either and atlas track or item, the missing information from the specification is:

* How to indicate dependency between atlas track and video component stored in an item.
* How to indicate dependency between an atlas item and video component stored as a track.

ISOBMFF specification provides two ways to address the above issue.

The first way is to utilize EntityToGroupBox that allows to group items and tracks and signal the dependency information. A new grouping type could be defined to indicate the dependency between the items and tracks that originated from the same V3C bitstream.

The second way is to utilize the 'unif' brand, which would be indicated in compatible\_brands of a TypeCombinationBox together with the V3C brands 'v3mt' and 'v3nt', in an ExtendedTypeBox, when a V3C bitstream is stored with tracks and items. Once the brands are present in TypeCombinationBox referencing items from tracks can be done with track references, and referencing tracks from items can be done with item references. We could extend the semantics of already existing reference codes, such as 'v3vo', 'v3va' and 'v3vg' to allow referencing between items and tracks. Alternatively, new track and item references types corresponding to 'v3vo', 'v3va' and 'v3vg' (e.g., to 'v3vO', 'v3vA' and 'v3vG' ) could be defined where the new types carry the additional semantics that referencing between items and tracks is allowed.

## Proposal

It is proposed to consider the scenarios from Clause 2 as well as the specification discussion from Clause 3 and include them in Technology under Consideration. It is also proposed to start a new Amendment to ISO/IEC 23090-10 to include further improvements that have been in principle agreed, but for which proper output document is missing.

# Supporting to decode with single decoder instance for V-PCC content

## Introduction

In 12th MPEG systems meeting, we proposed the examples to be able to support playback for client with single video decoder instance capability and the signalling to reduce client processing at that example[1].

The first proposal provided two implementation examples and suggested to be added to the informative annex.

The first example uses VDI supported by the ISO/IEC 23090-13[2]. The client uses “inserting” function in VDE to merge all video component bitstreams into a single bitstream and decode it with a single decoder instance.

The second example uses bitstream reconstruction of VVC subpicture supported by the ISO/IEC 14496-15[3]. The content generator encodes the video component as a VVC subpicture and adds a VVC merge base track. Figure 1 shows the tack structure.

A screenshot of a computer

Description automatically generated

Figure 1 track structure using bitstream reconstruction of VVC subpicture

When the client plays, the playback processes are following:

1. To find the atlas track.
2. To identify the VVC merge base track. The client chooses the VVC base track that the tracks referenced by the ‘subp’ track reference in the VVC base track are identical to the video component tracks referenced by the track reference in the atlas track.
3. To generate merged bitstream and decode. The client uses the VVC merge base track to merge video components into a single bitstream and decodes it with a single decoder instance.
4. To generate video component pictures. The client divides the decoded picture into video component pictures using the information of placement and resolution in the VVC merge base track.

Because these examples are use cases in which the Carriage of V-PCC uses a function specified in other standard, it is useful to add the examples in ISO/IEC 23090-10 in term of facilitating the client implementation.

There were supportive comments on git ([URL](http://mpeg.expert/software/MPEG/Systems/PCC-SYS/V-PCC/-/issues/185#note_79536)) regarding the first proposal.

The second proposal provided a new track reference from Atlas track to VVC merge base track to reduce playback processing when using bitstream reconstruction of VVC subpicture. (Figure 2)

This track reference makes the process simpler when identifying the VVC merge base track above the second example (2) of the first proposal.

A screenshot of a computer

Description automatically generated

Figure 2 track structure with reference from atlas track to VVC merge base track

The second proposal was not reached a consensus as it is good enough to specify the playback process for facilitating client implementation.

Therefore, in this contribution, we propose the first proposal.

## Proposal

It proposes to add examples of decoding all video components of V-PCC content with single decoder instance to promote implementation.

The examples describe the usage of VDI function in 23090-13 or VVC subpicture function in 14496-15 for merging bitstreams of video components to single bitstream and decode it with single decoder instance.

## Proposed text

Add the normative reference.

ISO/IEC 23090-13:202x , Information technology — Coded representation of immersive media — Part 13: Video decoding interface for immersive media

Add the new Annex.

**Annex X (informative) Implementation examples of** **decoding all video components of V3C contents with single decoder instance**

**X.1 General**

This annex describes implementation examples for decoding all video components of V3C content with single decoder instance using function defined in other specifications.

**X.2 Example of decoding with a single decoder instance using ISO/IEC 23090-13**

ISO/IEC 23090-13 Video decoding interface specifies the interface of Video Decoding Engine (VDE) and operation of elementary stream in VDE.

The “inserting” function is one of the specified operations in ISO/IEC 23090-13 and it can insert an elementary stream into another elementary stream and generate one merged elementary stream.

An example is shown in Figure X.1. Each bitstream of the V3C video component (occupancy, geometry, attribute) is input to the VDE. The first “inserting” function of VDI operation insert the geometry bitstream into the occupancy bitstream. Then, the second “inserting” function insert the attribute bitstream into the bitstream which generated by the first “inserting” function. As a result, generated bitstream (as merged bitstream in the figure) contains occupancy, geometry, and attribute. The bitstream contained occupancy, geometry, and attribute is decoded with single decoder instance. The decoded sequence which is output from decoder is divided into occupancy, geometry, and attribute sequences.

Note: the “inserting” function needs to regenerate parameter sets to ensure the bitstream conformance.

A computer screen shot of a diagram

Description automatically generated

**Figure X.1 decoding process using ISO/IEC 23090-13**

**X.3 Example of decoding with a single decoder instance using bitstream reconstruction of VVC subpicture**

It specifies the bitstream reconstruction using independent VVC subpictures and VVC merge base tracks in the ISO/IEC 14496-15.

In this example, the V3C video component is encoded as an independent VVC subpicture. And the VVC merge base track is generated to merge VVC subpicture tracks which are occupancy, geometry, attribute video component track. Figure X.2 shows the structure of VVC content. The “subp” track reference in VVC merge base track refers to the occupancy, geometry, attribute video component track.

A screenshot of a computer

Description automatically generated

**Figure X.2 track structure of V3C content using bitstream reconstruction of VVC subpicture**

When the client plays the V3C content, it identifies the VVC merge base track after finding the atlas track. To identify the VVC merge base track, the client chooses the VVC base track that the tracks referenced by the ‘subp’ track reference in the VVC base track are identical the video component tracks referenced from the track reference in the atlas track.

The decoding process of all video components is shown in Figure X.3. The merged bitstream is generated from VVC merge base track and VVC subpicture tracks which are occupancy, geometry, attribute video component track according to bitstream reconstruction defined in ISO/IEC 14496-15.

The bitstream (as merged bitstream in Figure X.3) contained occupancy, geometry, and attribute is decoded with a single decoder instance. The decoded sequence which is output from decoder is divided into occupancy, geometry, and attribute sequences using the information of placement and resolution in ‘trif’ sample group in the VVC merge base track.

A computer screen with text and images

Description automatically generated with medium confidence

**Figure X.3 decoding process using bitstream reconstruction of VVC subpicture**

Note : The VVC picture shall be rectangular picture. In other words, the merged picture shall be rectangular picture. To conformant this condition, it need to consider the resolution of each video component and placement of each video component in the merged picture. For example, if all video components are the same resolution, the merged picture can be rectangular picture by arranging from pictures horizontally (or vertically).

## Conclusion

It proposes informative annex as implementation example of decoding with a single decoder instance using VDI operation or bitstream reconstruction of VVC subpicture into amendment.

## References

1. m64180, “[VOL-SYS] Supporting to decode with single decoder instance for V-PCC content”, WG03, MPEG2023, Geneva, July 2023
2. “Text of ISO/IEC FDIS 23090-13 Video Decoding Interface for Immersive Media”, WG03, N0700, MPEG2022, Mainz, October 2022
3. ISO/IEC 14496-15:2022, 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

# Supplemental media in volumetric video in ISOBMFF

## Introduction

During MPEG 142 contribution m63056 (<http://mpeg.expert/software/MPEG/Systems/PCC-SYS/V-PCC/-/issues/177>) was presented as a use case for supplemental media for volumetric video. The contribution received several comments and clarification questions. In this contribution we continue the discussion started in m63056 and address the questions during the previous meeting.

The discussion continued during ad-hoc period between MPEG #143 and #144. It was agreed that the functionality for indicating supplemental media for packed video in V3C is missing from the file format, while it exists for multi-track encapsulated content with separate atlas tile tracks. During the discussion not enough opinions were provided to conclude if the functionality would be needed to support supplemental media indication for packed video. Thus, it was recommended to think about the need for such functionality. The contribution was resubmitted to conclude, if sufficient interest for the feature exists.

## Background

ISO/IEC 14496-15 6th edition DAM2 [2] specifies picture-in-picture concept for 2D video, where multimedia applications may offer a functionality, sometimes referred to as picture-in-picture service, for displaying a video with a smaller spatial resolution over a video with a larger spatial resolution. The mechanism allows to replace a portion of NAL units in main video with NAL units from supplemental video before decoding is performed. In this context, the videos with the smaller and larger spatial resolutions are referred to as the supplementary video and the main video, respectively.

ISO/IEC 23090-10 specifies the carriage of visual volumetric data in ISOBMFF. In some scenarios the volumetric video may be also accompanied by supplemental information, either to improve the user experience or to provide additional information to the end user. Some of the use cases where supplemental information may be used are the following.

* Advertisement displayed over the volumetric video
* Region annotations to provide hints to the user
* Local camera feed integrated with volumetric video from the server
* Adapting the content for different needs, e.g., friendly to children

Supplemental media (picture-in-picture) in volumetric video can be done on two levels

* Supplemental media can replace portion or all of the attribute of volumetric video (V3C attribute video component). For example, it can only replace the logo on football player t-shirt as shown in Figure 1.
* Supplemental media can replace the portion of the attribute of volumetric video (V3C attribute video component) and as well portion of the geometry of volumetric video (V3C atlas component and V3C geometry/occupancy video component). For example, it can replace the ball with a different object that is juggled by the football player as shown in Figure 1. Alternatively it could be used to inject reginally/contextually optimized advertizing in a V3C object so that it alters the geometry as well. For example a 3D effect may be added to the advertisement using the geometry component which makes the content come more to life.

A person in a red shirt

Description automatically generated

Figure 1 An example of rendered volumetric video frame

If the supplemental media is provided only on video level (e.g., simple texture replacement) a solution from the video specification, e.g., from ISO/IEC 14496-15, could be utilized. However, when the V3C atlas and V3C geometry components of volumetric video need to be addressed as well, then neither ISO/IEC 23090-10, nor any other specification, provide the necessary support for supplemental media to be integrated with volumetric video.

## Discussion

In m63056 we have presented a use case depicted on Figure 2. It was pointed out that this scenario can be address by already existing tools. One way to address a supplemental media with existing tools would be to provide alternatives to use V3C Atlas Tile Tracks and object switch alternatives box 'swpc' as presented on Figure 3.

A screenshot of a cell phone

Description automatically generated

Figure 2 An example of supplemental information in multi-track encapsulation mode as discussed in m63056.

A screenshot of a computer

Description automatically generated

Figure 3 An example of supplemental information in multi-track encapsulation mode with the use of existing functionality.

The approach presented in Figure 3 requires more video decoders and is not suitable for packed video data which targets only one video decoder. The original proposal supports both options. An example of this use case is presented in Figure 4

A screenshot of a computer

Description automatically generated

Figure 4 An example of supplemental information with packed video data track using approach discussed in m63056.

## Specification Consideration

### Overview

We propose to align concept of supplemental volumetric video to the one of supplemental video in ISO/IEC 14496-15 6th edition DAM2 [1].

In order to introduce supplemental volumetric video in ISO/IEC 23090-10, we would need to:

* define concepts of supplemental volumetric video,
* define AtlasTileReplacementSampleGroupEntry for supplemental V3C atlas track
* define track reference 'supa' to reference the V3C atlas track from the supplemental V3C atlas track.

### Supplemental information

The supplemental information is composed of one or more supplemental tracks, either atlas and/or video tracks. Each supplemental track is referenced to the corresponding. For example, the 'supm' track reference, defined in ISO/IEC 14496-15 6th edition DAM2, can be re-used for video tracks and a new 4CC code 'supa' can be defined for atlas tracks. Additionally, a track group including all supplemental tracks (atlas and video tracks) containing supplemental information is defined. For example, 4CC code 'svvg' code for such grouping could be used. An example of supplemental information and it relation to the main volumetric video is presented in Figure 2.

A black screen with white text

Description automatically generated

Figure 5 An example of supplemental information in multi-track encapsulation mode.

### SupplementalVolumetricVideoGroupTypeBox

A track group SupplementalVolumetricVideoGroupTypeBox includes all supplemental V3C tracks (atlas and video tracks).

class SupplementalVolumetricVideoGroupTypeBox() extends TrackGroupTypeBox ('svvg') {  
}

### AtlasTileReplacementEntry

Supplemental V3C atlas track contains AtlasTileReplacementEntry, the entry indicates that the NAL units representing the target supplementary atlas tile in the main V3C atlas track can be replaced with the corresponding NAL units of the supplementary V3C atlas track. The absence of this sample group would indicate that it is unknown whether such replacement is possible.

When this sample group is present, the player may choose to replace the NAL units representing the target supplementary tile in the main V3C atlas track with the corresponding NAL units of the supplementary V3C atlas track before sending the atlas to decoder for decoding. In this case, for a particular atlas tile in the main V3C atlas track, the corresponding NAL units of the supplementary V3C atlas track are all the NAL units in the decoding-time-synchronized sample in the supplementary V3C atlas track.

The syntax of AtlasTileReplacementSampleGroupEntry is as follows.

class AtlasTileReplacementSampleGroupEntry() extends VolumetricVisualSampleGroupEntry ('pptr') {  
 bit(5) reserved = 0;  
 unsigned int(3) tile\_id\_type;  
 unsigned int(8) num\_tile\_ids\_minus1;  
 for(i=0; i<=num\_tile\_ids\_minus1; i++)  
 unsigned int(16) tile\_id[i];  
}

The semantics of AtlasTileReplacementSampleGroupEntry is as follows:

tile\_id\_type equal to 0 is reserved. When tile\_id\_type is equal to 1, the tile IDs are the groupID values in the NAL unit map sample group for the NAL units that may be replaced by the NAL units of the supplementary V3C atlas track. tile\_id\_type values greater than 1 are reserved.

num\_tile\_ids\_minus1 plus 1 specifies the number of the following tile\_id[i] fields.

tile\_id[i] specifies the i-th ID for the NAL units representing the target atlas tile.

## Proposal

It is proposed to consider the presented use cases and the specification discussion from Clause 4 and include it the second Amendment to ISO/IEC 23090-10.

## References

[1] ISO/IEC 14496-15 6th edition DAM2 on picture-in-picture support and other extensions, ISO/IEC JTC1/SC29/WG3 output document MDS22325\_WG03\_N00801, January 2023, online meeting.

# Support of 2D snapshot (m66538)

## Introduction

Compressing volumetric contents has strong benefit for saving resources for storage and delivery of the contents. However, it introduces a challenge for quick preview or trick play of the contents similar to any compressed video data has. Several video codecs should be initiated and more than one video frames for each components of a compressed volumetric frame should be decoded and a volumetric frame should be reconstructed with the decoded results for a quick preview of volumetric content. As one directional or bidirectional dependent coding could have been also applied to further enhance compression efficiency, more than one video frame should be decoded to get a specific frame of volumetric content. If random access points of the components are not aligned or the frame rates of the components are different each other, a greater number of video frames should be decoded to get the result. So, quick preview or trick play of a volumetric contents which used be a straight forward easy job for uncompressed volumetric contents become quite complicated resource and time-consuming thing when the contents are compressed.

To solve such issues and make quick preview or trick play operation simpler and easier 2D snapshot images of volumetric contents at certain points of time with a camera at a certain position and direction could be provided. A client can decode and present 2D snapshot images instead of volumetric contents by decoding compressed bitstreams and compositing/rendering volumetric contents when it is not really needed.

This contribution proposes a method to add one or more tracks to carry 2D snapshot images for a volumetric content.

## 2D snapshot image track

### Overview

The 2D snapshot image track contains one or more samples of coded bitstream of 2D image of a coded volumetric frame rendered at a certain location and direction. Each sample contains a 2D projected image of a coded volumetric frame whose composition time is same with such sample. There can be more than one 2D snapshot image track for a single CVS and each of them contains different version of snapshot image.

### Restriction to the track

The value of handler\_type of the 2D snapshot image track shall be ‘vide.’ In other words, the track used as a 2D snapshot image should be the one which could have ‘vide’ as a value for handler\_type. All samples in the 2D snapshot image track shall be sync samples.

### Track references

To associate a 2D snapshot image track with the tracks containing V3C data, track reference tool of ISO/IEC 14496-12 shall be used. One or more TrackReferenceTypeBoxes shall be added to a TrackReferenceBox within the TrackBox of the V3C atlas track or V3C atlas tile track, one for each 2D snapshot image tracks. The TrackReferenceTypeBox shall contain array of track\_IDs designating the tracks containing 2D snapshot images which the V3C atlas track or V3C atlas tile track references. The 4CC value of reference\_type of such TrackReferenceTypeBox shall be ‘2dsi.’

### Indication of camera used for rendering snapshot images

Information about the camera used to render 2D snapshot images is provided as viewport information timed-metadata track. A viewport sample whose composition time is same with a snapshot image provide information about the camera used to render such image. When such viewport timed-metadata track is provided the value of viewport\_type is set to ‘0.’ The viewport information timed-metadata track shall reference corresponding 2D snapshot image track instead of V3C atlas track and ‘2dci’ shall be used for reference\_type.

## Conclusion

It is proposed to add a method to carry the 2D snapshot images to support simple and easy preview or scanning of compressed volumetric contents. By adding this feature compressed volumetric contents will also become useful for the use cases where decoding, composition and rendering of compressed volumetric contents is challenging.

# Support of multi-directional 2D snapshot (m67720)

## Introduction

Compressing volumetric contents has strong benefit for saving resources for storage and delivery of the contents. However, it introduces a challenge for quick preview or trick play of the contents similar to any compressed video data has. Several video codecs should be initiated and more than one video frames for each components of a compressed volumetric frame should be decoded and a volumetric frame should be reconstructed with the decoded results for a quick preview of volumetric content. As one directional or bidirectional dependent coding could have been also applied to further enhance compression efficiency, more than one video frame should be decoded to get a specific frame of volumetric content. If random access points of the components are not aligned or the frame rates of the components are different each other, a greater number of video frames should be decoded to get the result. So, quick preview or trick play of a volumetric contents which used be a straight forward easy job for uncompressed volumetric contents become quite complicated resource and time-consuming thing when the contents are compressed.

To solve such issues and make quick preview or trick play operation simpler and easier 2D snapshot images of volumetric contents at certain points of time with a camera at a certain position and direction could be provided. A client can decode and present 2D snapshot images instead of volumetric contents by decoding compressed bitstreams and compositing/rendering volumetric contents when it is not really needed.

At the previous meeting, a method to add one or more tracks to carry 2D snapshot images for a volumetric content has been presented. The contribution proposes to provide a track referenced from a volumetric content track containing 2D snapshot images. It proposes to optionally include viewport information timed-metadata track to indicate information about the camera use to render the 2D snapshot images as [m66538](https://dms.mpeg.expert/doc_end_user/documents/145_OnLine/wg11/m66538-v2-m66538v22Dsnapshotimagetrack.zip).

In this contribution, a method to provide more than one 2D snapshot image for a certain coded volumetric frame is proposed. As a volumetric frame can be viewed from multiple directions a single snapshot image may not be sufficient to provide enough information about a volumetric frame. It would be possible to provide more than one snapshots by adding more than one snapshot image tracks but it would be not quite efficient for certain use case, e.g. camera is fixed for entire volumetric frames in a track. As modern video codecs such as HEVC, VVC or EVC provides mechanisms to include more than one subset of pictures such as tiles or subpictures, and ISO/IEC 14496-15 can indicate each of them as a separate sub-sample, this contribution proposes a method to include more than one snapshot images by using such features when the camera information is same for entire samples in a track.

## Multi-directional snapshot image track

### Overview

The multi-directional snapshot image track contains one or more samples of coded bitstream of 2D image of a coded volumetric frame. Each sample consist of more than one sub-samples and each sub-sample contains a 2D projected image of a coded volumetric frame where location and direction of projection of each sub-sample are not same each other. SubSampleInformationBox shall present in a multi-directional snapshot image track to provide subsample information. The set of location and direction of cameras used for projection shall remain same for a single track. Composition time of the 2D images shall be same with the sample of coded volumetric frame.

### Track references

To associate a multi-directional snapshot image track with the tracks containing V3C data, track reference tool of ISO/IEC 14496-12 shall be used. One or more TrackReferenceTypeBoxes shall be added to a TrackReferenceBox within the TrackBox of the V3C atlas track or V3C atlas tile track, one for each 2D snapshot image tracks. The TrackReferenceTypeBox shall contain array of track\_IDs designating the tracks containing 2D snapshot images which the V3C atlas track or V3C atlas tile track references. The 4CC value of reference\_type of such TrackReferenceTypeBox shall be ‘mdsi.’

### Restriction to the track

The track referenced from a V3C atlas track or V3C atlas tile track with reference\_type ‘mdsi shall be represented in the file as restricted video and shall use a generic sample entry ‘resv’ with following additional requirements:

— SchemeTypeBox shall be present in RestrictedSchemeInfoBox and scheme\_type is set to 'mdst'

— All samples in the track shall be sync samples

— SubSampleInformationBox shall be present and the value of subsample\_count of SubSampleInformationBox shall be same for all entries and greater than one.

#### Multi-dimensional snapshot camera information box

##### Definition

Box Type: 'mdst'

Container: SchemeInformationBox

Mandatory: Yes (when the SchemeType is 'mdst')

Quantity: One

The Multi-dimensional snapshot camera information box is used to indicate the information about the camera used to render the snapshots of volumetric frames for each sub-samples indicated by SubSampleInformationBox. i-th view port information shall indicate i-th sub-sample in bitstream order within a sample.

##### Syntax

aligned(8) class MultiDimSnapshotCameraInfoBox extends extends FullBox('mdst', version = 0, 0)

{

unsigned int(8) num\_viewports;

for (int i=1; i <= num\_viewports; i++){

unsigned int(1) camera\_extrinsic\_flag[i];

unsigned int(1) camera\_intrinsic\_flag[i];

bit(6) reserved = 0;

ViewportInfo (camera\_extrinsic\_flag[i], camera\_intrinsic\_flag[i]);

}

}

##### Semantics

num\_viewport indicates the number of viewport signaled in the sample. The value of this field shall be equal to the value of the value of subsample\_count of SubSampleInformationBox. i-th viewport provides information about the camera for the i-th subsample in bitstream order.

camera\_intrinsic\_flag[i] equal to 1 indicates that the intrinsic camera parameters are present in the i-th viewport.

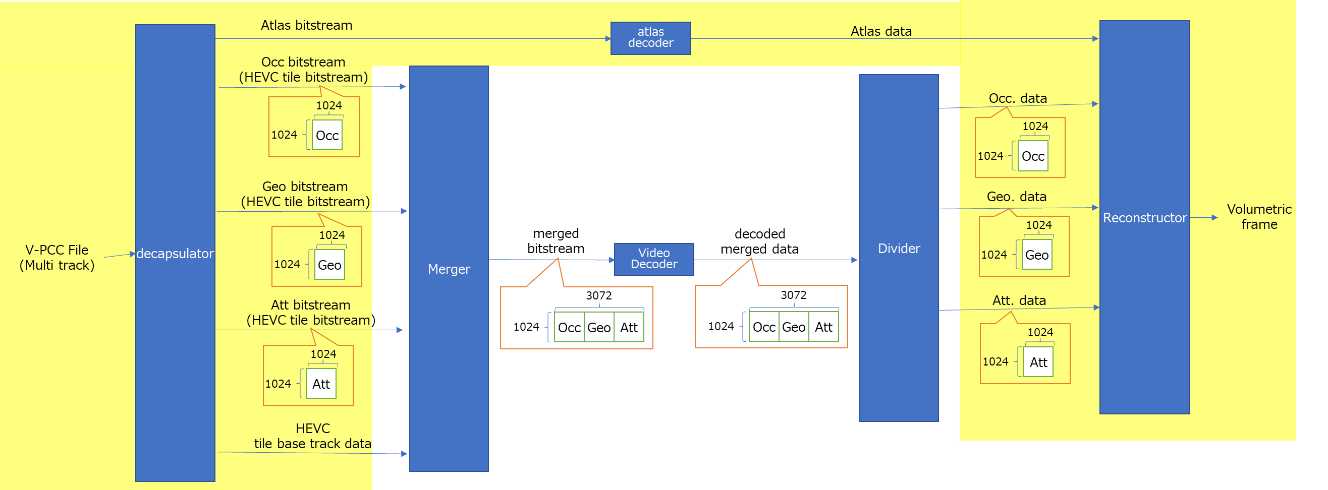
camera\_extrinsic\_flag[i] equal to 1 indicates that the extrinsic camera parameters are present in the i-th viewport.

ViewportInfo provides information about the camera. Then syntax and semantics of this class is specified in subclause 10.2.3 of ISO/IEC 23090-10

# Supporting to decode with single decoder instance for V-PCC content (m67634)

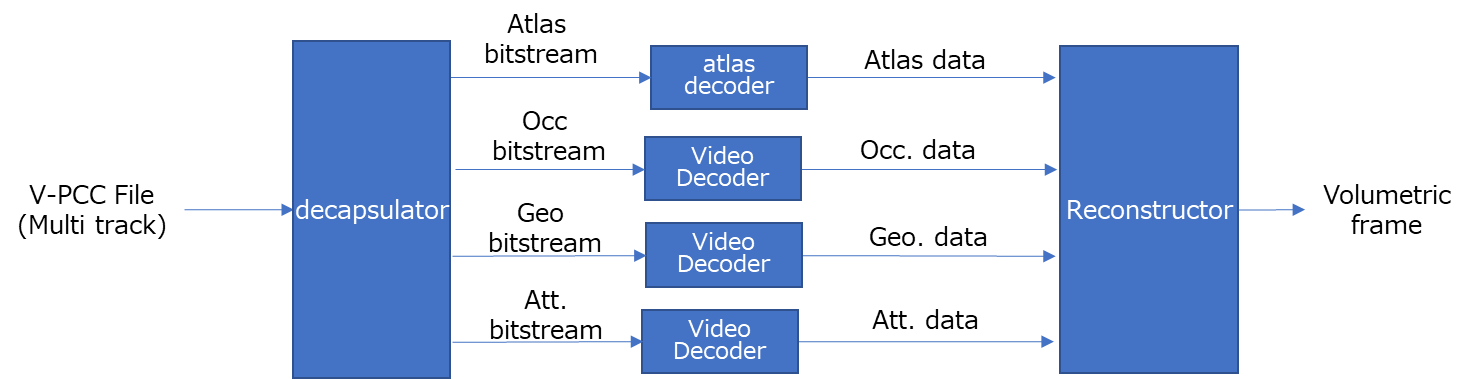
## Proposal

In this contribution, it proposes following:

1. Update the process diagram which include a non-video component, for comment resolution.  
   Add an Atlas bitstream and a reconstruction process using decoded atlas data and each decoded video data to each implementation example diagram.  
   Figure 1 shows an example of a complete decoder structure using HEVC tiles. The added part is highlighted in yellow. It also updates figures which using VDI operation and using VVC subpicture in the same way.  
     
   

**Figure 1 complete decoder structure using HEVC tiles**

1. Adds the basic decoding V-PCC content process with multiple video decoder instance because it helps the readers understand by clarifying the difference from decoding process with single video decoder instance. Specifically, it adds the Figure 2 and process description in the ‘General’ clause.



**Figure 2 complete decoder structure for V-PCC content**

## Proposal text

The updates as explained in the section 2 are integrated into previous proposal text [2] and are highlighted in yellow.

-------

Add the Bibliography.

[3] ISO/IEC 23090-13:202x , Information technology — Coded representation of immersive media — Part 13: Video decoding interface for immersive media

Add the new Annex.

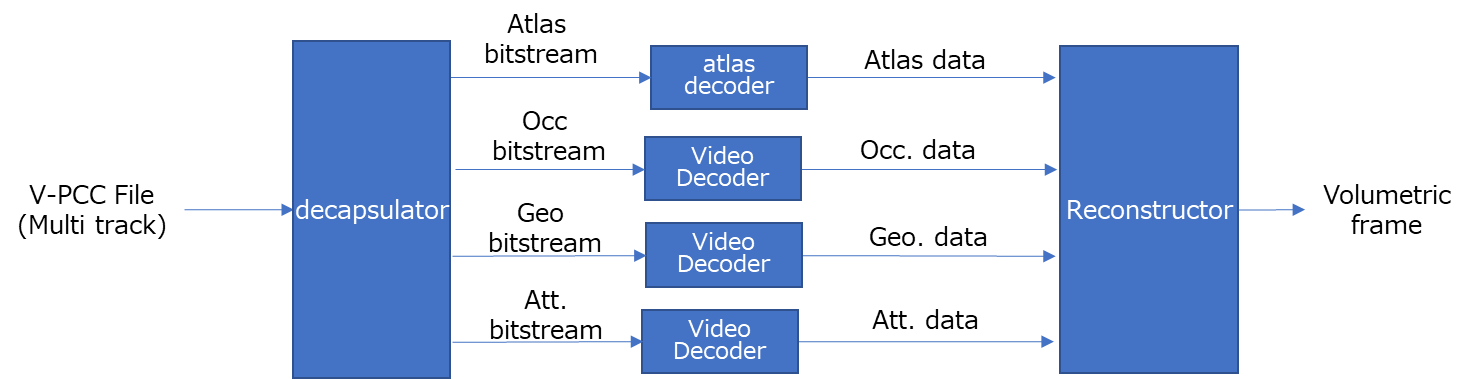
**Annex X (informative) Implementation examples of decoding all video components of V3C contents with single decoder instance**

**X.1 General**

For V3C content with multiple video components, each video component will be decoded individually.

For example, Figure X.1 shows the processing when V-PCC file which multi-track encapsulated atlas, occupancy, geometry, and attribute encoded data.

V-PCC file is decapsulated to atlas, occupancy, geometry, and attribute bitstream in decapsulator, respectively All bitstream are decoded in decoder, respectively. It is reconstructed into a volumetric frame using all decoded data. This case is typically handled by multiple video decoder instances.



**Figure X.1 decoding process of V-PCC content**

This annex describes implementation examples for decoding all video components of V3C content with single video decoder instance using function defined in other specifications.

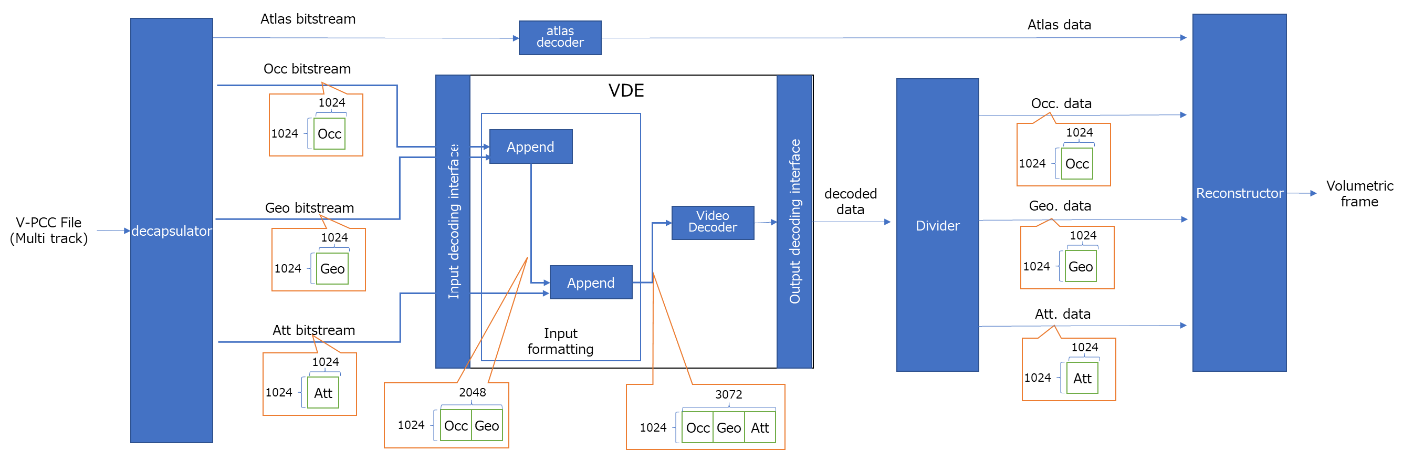
**X.2 Example of decoding with a single decoder instance using ISO/IEC 23090-13**

ISO/IEC 23090-13 Video decoding interface specifies the interface of Video Decoding Engine (VDE) and operation of elementary stream in VDE.

The “inserting” function is one of the specified operations in ISO/IEC 23090-13 and it can insert an elementary stream into another elementary stream and generate one merged elementary stream.

An implementation example is shown in Figure X.2. A V-PCC file which multi-track encapsulated atlas, occupancy, geometry, and attribute encoded data is decapsulated to atlas, occupancy, geometry, and attribute bitstream in decapsulator, respectively. The occupancy, geometry, and attribute bitstream are input to the VDE. The first “inserting” function of VDI operation insert the geometry bitstream into the occupancy bitstream. Then, the second “inserting” function insert the attribute bitstream into the bitstream which generated by the first “inserting” function. As a result, generated bitstream (as merged bitstream in the figure) contains occupancy, geometry, and attribute. The bitstream contained occupancy, geometry, and attribute is decoded with single decoder instance. The decoded data which is output from decoder is divided into occupancy, geometry, and attribute data. These data and atlas data are reconstructed to the volumetric frame.

Note: the “inserting” function needs to regenerate parameter sets to ensure the bitstream conformance.

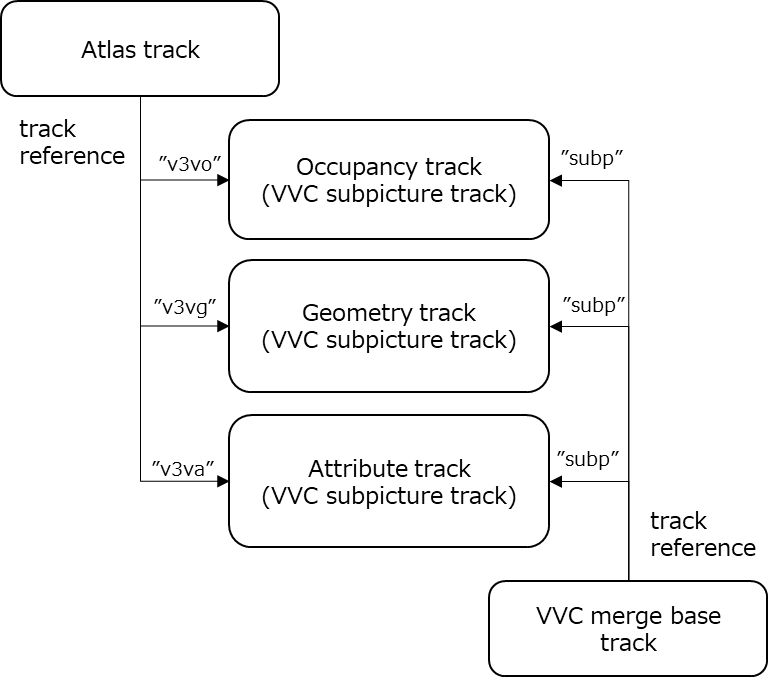
****

**Figure X.2 decoding process using ISO/IEC 23090-13**

**X.3 Example of decoding with a single decoder instance using bitstream reconstruction of VVC subpicture**

It specifies the bitstream reconstruction using independent VVC subpictures and VVC merge base tracks in the ISO/IEC 14496-15.

In this example, the V3C video component is encoded as an independent VVC subpicture. For example, all V3C video components may be encoded as independent subpictures in one picture, and then divide into tracks for each V3C video component(VVC subpicture). And the VVC merge base track is generated to merge VVC subpicture tracks which are occupancy, geometry, attribute video component track. Figure X.3 shows the structure of VVC content. The “subp” track reference in VVC merge base track refers to the occupancy, geometry, attribute video component track.

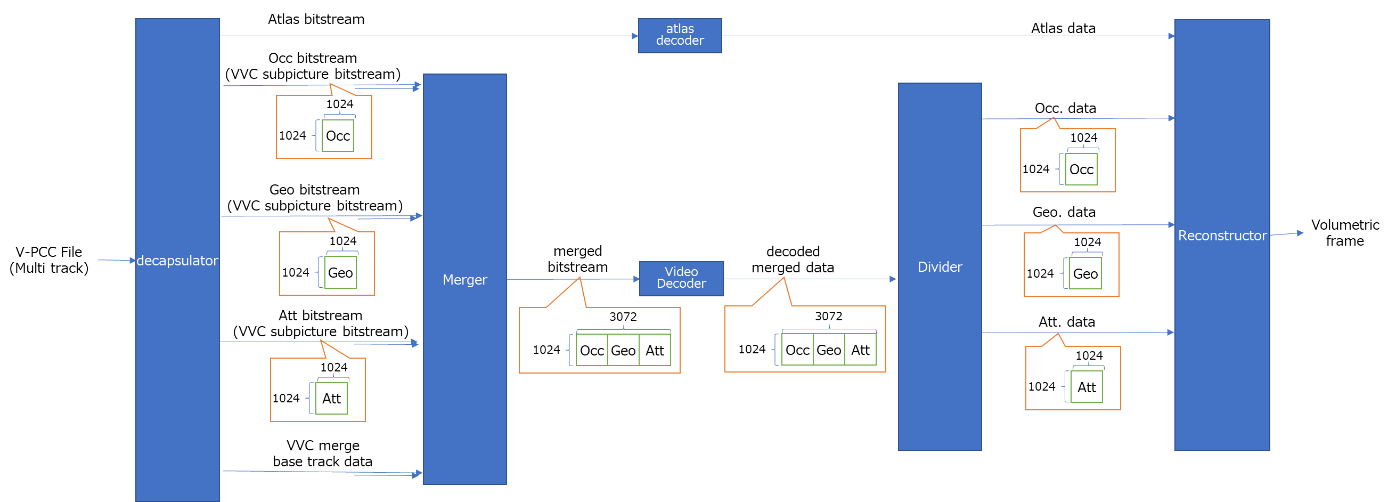


**Figure X.3 track structure of V3C content using bitstream reconstruction of VVC subpicture**

When the client plays the V3C content, it may identify the VVC merge base track after finding the atlas track. To identify the VVC merge base track, the client may choos the VVC base track that the tracks referenced by the ‘subp’ track reference in the VVC base track are identical the video component tracks referenced from the track reference in the atlas track.

The decoding process of V-PCC file which multi-track encapsulated atlas, occupancy, geometry, and attribute encoded data is shown in Figure X.4. V-PCC file which multi-track encapsulated atlas, occupancy, geometry, and attribute encoded data is decapsulated to atlas, occupancy, geometry, and attribute bitstream in decapsulator, respectively.. The merged bitstream is generated from VVC merge base track and VVC subpicture tracks which are occupancy, geometry, attribute video component track according to bitstream reconstruction defined in ISO/IEC 14496-15.

The bitstream (as merged bitstream in Figure X.4) contained occupancy, geometry, and attribute is decoded with a single decoder instance. The decoded merged data which is output from decoder is divided into occupancy, geometry, and attribute data using the information of placement and resolution in ‘trif’ sample group in the VVC merge base track. These divided data and atlas data are reconstructed to the volumetric frame.



**Figure X.3 decoding process using bitstream reconstruction of VVC subpicture**

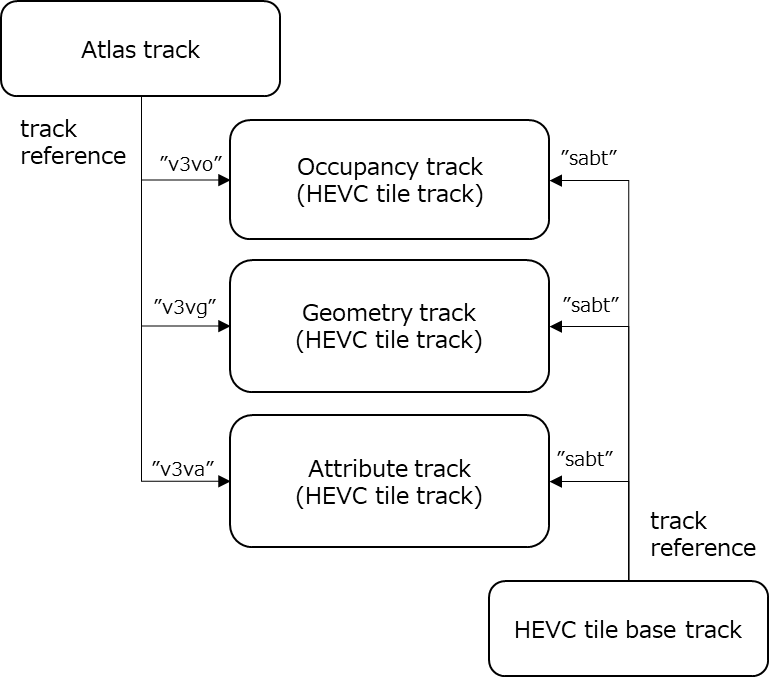
Note1 : The VVC picture is rectangular picture. In other words, the merged picture is rectangular picture. To conformant this condition, it need to consider the resolution of each video component and placement of each video component in the merged picture. In the Figure X.3, all video components are the same resolution, And the merged picture can be rectangular picture by arranging from pictures horizontally.

Note2 : The placement of subpictures in the merged bitstream is configured in the VVC merge base track. The placement of subpictures may be arranged not only horizontally, but also vertically, or both horizontally and vertically

**X.4 Example of decoding with a single decoder instance using bitstream reconstruction of HEVC tile**

It specifies the bitstream reconstruction using independent HEVC tiles and HEVC tile base tracks in the ISO/IEC 14496-15.

In this example, the V3C video component is encoded as an independent HEVC tile. For example, all V3C video components may be encoded as independent HEVC tile in one picture, and then divide into tracks for each V3C video component (HEVC tile). And the HEVC tile base track is generated to merge HEVC tile tracks which are occupancy, geometry, attribute video component track. Figure X.5 shows the structure of V3C content. The “sabt” track reference in VVC merge base track refers to the occupancy, geometry, attribute video component track.

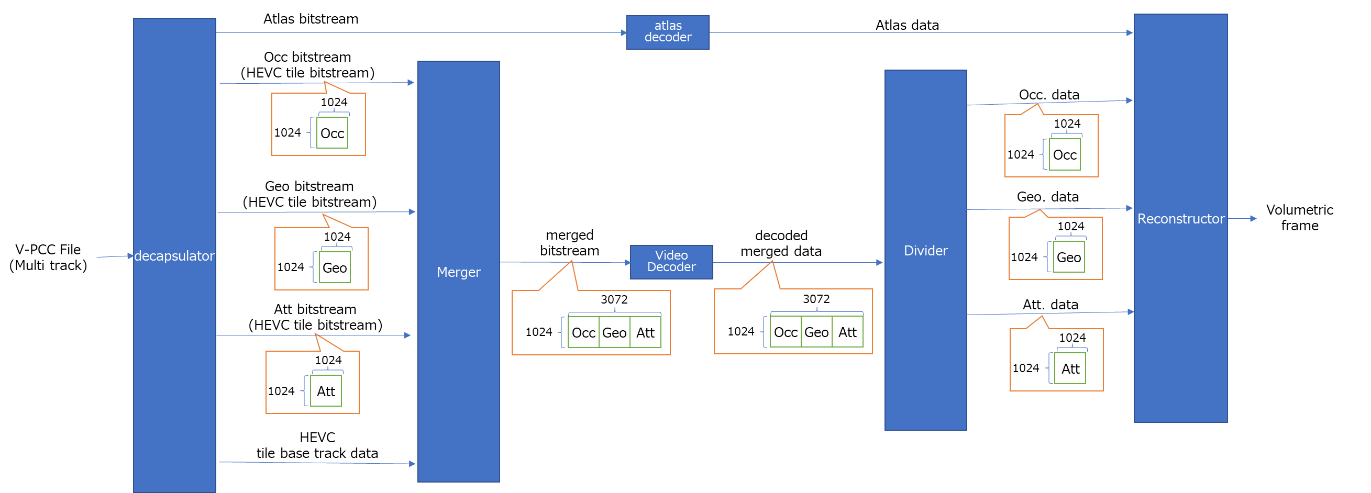


**Figure X.5 track structure of V3C content using bitstream reconstruction of HEVC tile**

When the client plays the V3C content, it can identify the HEVC tile base track after finding the atlas track. To identify the HEVC tile base track, the client can choos the HEVC tile track that the tracks referenced by the ‘sabt’ track reference in the VVC base track are identical the video component tracks referenced from the track reference in the atlas track.

The decoding process of V-PCC file which multi-track encapsulated atlas, occupancy, geometry, and attribute encoded data is shown in Figure X.6. V-PCC file which multi-track encapsulated atlas, occupancy, geometry, and attribute encoded data is decapsulated to atlas, occupancy, geometry, and attribute bitstream in decapsulator, respectively. The merged bitstream is generated from HEVC tile base track and HEVC tile tracks which are occupancy, geometry, attribute video component track according to bitstream reconstruction defined in ISO/IEC 14496-15.

The bitstream (as merged bitstream in Figure X.6) contained occupancy, geometry, and attribute is decoded with a single decoder instance. The decoded sequence which is output from decoder is divided into occupancy, geometry, and attribute sequences using the information of placement and resolution in ‘trif’ sample group in the HEVC tile base track. These divided data and atlas data are reconstructed to the volumetric frame.



**Figure X.6 decoding process using bitstream reconstruction of HEVC tile**

Note1 : The HEVC picture is rectangular picture. In other words, the merged picture is rectangular picture. To conformant this condition, it need to consider the resolution of each video component and placement of each video component in the merged picture. For example, if all video components are the same resolution, the merged picture can be rectangular picture by arranging from pictures horizontally (or vertically).

Note2 : The placement of HEVC tile in the merged bitstream is configured in the HEVC base track. The placement of subpictures may be arranged not only horizontally, but also vertically, or both horizontally and vertically.

# References

1. ISO/IEC 14496-12:2015 “Information technology – Coding of audio-visual objects – Part 12: ISO Base Media File Format”.
2. N17990, “Technologies under Consideration for ISOBMFF”. October 2018, Macau, CN.
3. Marius Preda. “Point Cloud Compression in MPEG”. MP20 Workshop, Hong-Kong October 2017.
4. N17675. “First idea on Systems technologies for Point Cloud Coding”. April 2018, San Diego, US.
5. N17771. “PCC WD V-PCC (Video-based PCC)”. July 2018, Ljubljana, SI.
6. N18059. “WD of Storage of V-PCC in ISOBMFF Files”. October 2018, Macau, CN.
7. m45100. “On encapsulation of V-PCC bitstream in single ISOBMFF track”. October 2018, Macau, CN.
8. m44800. “Media Processing related Track Derivations in ISOBMFF”. October 2018, Macau, CN.
9. N17984. “WD of Network based Media Processing”. October 2018, Macau, CN.
10. m42509. “Signaling Rectangular and Spherical Regions in 2D and 3D Coordinate Systems for VR Media Content in Timed Metadata Tracks”. April 2018, San Diego, USA.
11. ISO/IEC 23001-10:2015 “Information technology – MPEG systems technologies — Part 10: Carriage of timed metadata metrics of media in ISO base media file format”.
12. Text of ISO/IEC CD 23090-5 Video-based Point Cloud Compression, N18030, MPEG2018, Macao, China, October 2018
13. “A method of 3D grid representation for Spatial Random Access in V-PCC”, M44975, MPEG2018, Macao, China, October 2018
14. M46042, “Carriage of MPEG V-PCC layers in ISOBMFF”, MPEG#125, Marrakech, Jan. 2019
15. M46433, “V-PCC entry point in ISOBMFF”, MPEG#125, Marrakech, Jan. 2019
16. N18180, “Study Text of ISO/IEC CD 23090-5 Video-based Point Cloud Compression”, MPEG 125, January 2019
17. N18266, “WD of ISO/IEC 23090-10 Carriage of PC Data”, MPEG 125, January 2019
18. ISO/IEC 14496-12, “Coding of Audio-Visual Objects, Part 12: ISO Base Media File Format”, 2015
19. N18312, “Revised text of ISO/IEC FDIS 23009-1 3rd edition”, MPEG 125, January 2019
20. Removed
21. ISO/IEC 14496-12, “ISO Base Media File Format”
22. N18180, Study Text for ISO/IEC 23090-5:2019 CD “Information technology — Coded Representation of Immersive Media — Part 5: Video-based Point Cloud Compression”, MPEG#125, Marrakech, Morocco, Jan, 2019.
23. Sketchfab 3D asset store: <https://sketchfab.com/feed>
24. w18266, “ISO/IEC 23090-10: WD of Carriage of Point Cloud Data”, MPEG 125, Marrakech, Jan 2019
25. ISO/IEC 23008-12, “High Efficiency Image File Format”
26. N18188, “CE 2.19 on V-PCC tiles”, MPEG 125, January 2019
27. N13753, “PCC Requirements”, MPEG 121, January 2018
28. m49105, “On Carriage of V-PCC Layers in ISOBMFF”, MPEG 127, July 2019.
29. m47436, “[VPCC] DASH Signaling for V-PCC”, MPEG 126, March 2019
30. m49107, “Supporting V-PCC Tile Groups and Spatial Access in MPEG-I Part 10”, MPEG 127, July 2019
31. “[VPCC] Meeting notes from joint 3DG/Systems meeting on system technologies of V-PCC”, m48265, MPEG 127, Gothenburg, Jul 2019
32. “Attribiute not present SEI message”, m49135, MPEG 127, Gothenburg, Jul 2019 (merged contribution ID: m49113)
33. N18607, “Technologies under Consideration for Carriage of PC Data”, MPEG 127, Jul 2019
34. M49878 [OMAF] Signaling and storage of sub-picture timed metadata
35. ISO/IEC 23009-5: Information Technology — Dynamic adaptive streaming over HTTP (DASH) — Part 5: Server and network assisted DASH (SAND)
36. w19441: Potential Improvements to ISO/IEC DIS 23090-10 Carriage of Visual Volumetric Video-based Coding Data
37. m63055 [37.1][VOL-SYS]Timed and non-timed V3C components in ISOBMFF
38. m66538, “[38.2] 2D snapshot image track for V3C content,” MPEG145, January 2024
39. m66720, “[37.2] Supporting multi-directional 2D snapshot for V3C content,” MPEG146, April 2024
40. m67634, “[VOL-SYS] Supporting to decode with single decoder instance for V-PCC content,” MPEG146, April 2024