**ISO/IEC JTC 1/SC 29/WG 03 N0295**

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

**Document type:** Output Document

**Title:** Technologies under Consideration for Derived visual tracks including further visual derivations

**Status:** Approved

**Date of document:** 2021-07-16

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

**No. of pages:** 20 (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 N0295**

**July 2021, Virtual**

|  |  |
| --- | --- |
| **Title** | **Technologies under Consideration for Derived visual tracks including further visual derivations** |
| **Source** | **WG 03, MPEG Systems** |
| **Editors** | **Frédéric Mazé** |
| **Status** | **Approved** |
| **Serial Number** | **20549** |

**Abstract**

This document collects following candidate technologies for the Derived visual tracks (ISO/IEC 23001-16):

[1. Transform operations under study 3](#_Toc77331485)

[1.1 Contrast 3](#_Toc77331486)

[1.2 Dynamic Track Overlay Composition 3](#_Toc77331487)

[1.3 Transformation Matrix Composition 3](#_Toc77331488)

[1.4 Track Grouping Composition 4](#_Toc77331489)

[1.5 Matrix Transformations 5](#_Toc77331490)

[2. Derivation transformations to support Immersive Media Processing 5](#_Toc77331491)

[2.1 Stitching 5](#_Toc77331492)

[2.2 Projection 6](#_Toc77331493)

[2.3 Packing 7](#_Toc77331494)

[2.4 Viewport 8](#_Toc77331495)

[3. Derived Immersive Tracks 8](#_Toc77331496)

[3.1 Motivation 9](#_Toc77331497)

[3.2 Proposal text 9](#_Toc77331498)

# Transform operations under study

See also: <http://mpegx.int-evry.fr/software/MPEG/Systems/FileFormat/DerivedVis/issues/5>

## Contrast

|  |  |  |
| --- | --- | --- |
| ***Operation*** | ***Inputs*** | ***Parameters*** |
| Contrast | 1 | 1. Signed adjustment |

## Dynamic Track Overlay Composition

### Definition

Box Type: 'dtoc'  
Mandatory (per sample): No  
Quantity (per sample): Any  
Inputs: (1) visual track, (2) overlay timed metadata track

The dynamic track overlay composition 'dtoc' derivation transformation, when present, requires that the number of inputs is equal to 2, and the first input is a visual track and the second input is an overlay timed metadata track.

This derivation transformation defines a derived visual track, each of whose samples is a sample of the first input track as a background, overlaid with sample images of other visual tracks, according to the timed metadata signaled in the second metadata track.

Note that the possible blending is not signaled. It should be included in the overlay timed metadata track definition in future versions of ISO/IEC 23090-2.

### Syntax

aligned(8) class DynamicTrackOverlayComposition   
extends VisualDerivationBase ('dtoc', flags) {  
}

## Transformation Matrix Composition

### Definition

Box Type: 'tmcp'  
Mandatory (per sample): No  
Quantity (per sample): Any  
Inputs: At least one (1)

The 'tmcp' derivation transformation, when present, requires that the number of inputs is greater or equal to 1, and the inputs are visual tracks.

This derivation transformation defines a derived visual track, each of whose sample images is a larger canvas overlaid with sample images of one or more input tracks in the layering order that is same as the order they are listed, i.e., the bottom-most input image first and the top-most input image last. The size of the canvas is specified by output\_width and output\_height.The time-parallel samples of all input tracks are spatially arranged onto the canvas according to the syntax and semantics of the matrix values in the input track headers; that is, the sizes and locations of sample images of the input tracks are specified by width, height, and matrix within TrackHeaderBox. Any portion of an arranged input image that is out of the canvas will be ignored.

NOTE The canvas/image/video background is used for the situation where the input sample images do not cover the entire canvas background. This is useful, for example, for 360° VR content where the VR video does not cover the entire spherical surface.

### Syntax

aligned(8) class TrackMatrixComposition   
extends VisualDerivationBase ('tmcp', flags) {  
}

## Track Grouping Composition

### Definition

Box Type: 'tgcp'  
Mandatory (per sample): No  
Quantity (per sample): Any  
Inputs: At least one (1)

The 'tgcp' derivation transformation, when present, requires that the number of inputs is greater or equal to 1, and the inputs are visual tracks. All the input tracks shall belong to a same track group, all containing the sub-picture composition Track Group Box '2dsr' in ISO/IEC 23090-2 with a same track\_group\_id value, but no any two of the tracks belong to a same alternate track group (i.e., they contain no TrackHeaderBox with a same non-zero alternate\_group value that indicates they belong to a same alternate group for the purpose of selecting only one from the alternate group).

This derivation transformation defines a derived visual track, each of whose sample images is a larger canvas overlaid with sample images of one or more input tracks in the layering order that is same as the order they are listed, i.e., the bottom-most input image first and the top-most input image last. The time-parallel samples of all tracks of the same sub-picture composition track group are spatially arranged according to the syntax and semantics of the track group; that is, the size of the canvas is specified by composition\_width and composition\_height, and the sizes and locations of sample images of the input tracks are specified by track\_width, track\_height, track\_x and track\_y in the associated sub-picture composition track group box '2dsr' in ISO/IEC 23090-2.

This derivation transformation supports specifying a single color canvas background or a single image background, and signaling the image overlay blending.

### Syntax

aligned(8) class SubPictureTrackGroupComposition   
extends VisualDerivationBase ('tgcp', flags) {  
}

## Matrix Transformations

### Definition

Box Type: 'matt'  
Mandatory (per sample): No  
Quantity (per sample): Any  
Inputs One (1)

The sample matrix transformation 'matt' derivation transformation transforms the input image item or sample of an input track according to the transformation defined by a 3x3 matrix.

### Syntax

aligned(8) class MatrixTransformation   
extends VisualDerivationBase (‘matt’, flags) {  
 int (32)[9] matrix;  
}

### Semantics

matrix specifies a 3x3 transformation matrix {a, b, u, c, d, v, x, y, w}.

# Derivation transformations to support Immersive Media Processing

See also:

* <http://mpegx.int-evry.fr/software/MPEG/Systems/FileFormat/DerivedVis/issues/18>
* <http://mpegx.int-evry.fr/software/MPEG/Systems/FileFormat/DerivedVis/issues/7>

## Stitching

### Definition

Box Type: 'stch'  
Mandatory (per sample): No   
Quantity (per sample): Any  
Inputs: (1) visual input A, (2) visual input B

The Stitching derivation transformation provides information for the process of stitching images of two visual inputs and map them onto to a projection surface to form a stitched visual output.

This derivation transformation specifies width, visual\_widthA and visual\_widthB, and height, visual\_heightA and visual\_heightB, of each of the two visual inputs, and a projection surface type of a derived sample resulting from stitching corresponding images of the two visual inputs.

### Syntax

aligned(8) class Stitching  
extends VisualDerivationBase('stch', flags){  
 unsigned int(16) visual\_widthA; // parameter 1  
 unsigned int(16) visual\_heightA; // parameter 2  
 unsigned int(16) visual\_widthB; // parameter 3  
 unsigned int(16) visual\_heightB; // parameter 4  
 bit(4) reserved = 0; // not a parameter  
 unsigned int(4) projection\_surface\_type; // parameter 5  
}

### Semantics

visual\_widthA, visual\_heightA, visual\_widthB and visual\_heightB specify, respectively, the width and height of the two visual inputs in units of luma samples.

projection\_surface\_type specifies a type of surface the stitched image is to be projected onto, according to the following table.

|  |  |
| --- | --- |
| **Value** | **Projection Surface** |
| 0 | Rectilinear |
| 1 | Spherical |
| 2 | Cylindered |
| 3 | Cubic |
| 4 ~ 15 | Reserved |

## Projection

### Definition

Box Type: 'proj'  
Mandatory (per sample): No  
Quantity (per sample): Any  
Inputs One

The Projection derivation transformation provides information for the process of projecting images of an input track onto a 2D plane to form a derived track, according to a projection format such as the Equi-rectangular Projection (ERP) and Cube Map Projection (CMP) as given in ISO/IEC 23090-2. An indicator is\_reverse is used to indicate whether the operation is a (forward) projection construction or reverse projection one.

This derivation transformation given below assumes the input and (derived) output tracks are an ISO/IEC 23090-2 compliant tracks. When the operation is a (forward) projection construction (is\_reverse == 0), the input track is an un-projected picture track, the output track is a projected picture track, and the projection format packing structure ProjectionFormatStruct() is signaled (within ProjectionFormatProperty) to indicate the projection format in the projected pictures. When the operation is a reverse projection construction (is\_reverse == 0), the input track is a projected picture track which has a projection format item property containing a ProjectionFormatStruct() structure, the output track is an un-projected picture track, and it is the projection format structure ProjectionFormatStruct() in the input track is used to indicate the projection format in the projected pictures.

### Syntax

aligned(8) class Projection  
extends VisualDerivationBase('proj', flags) {  
 bit(1) is\_reverse; // parameter 1  
 bit(2) reserved = 0; // not a parameter  
 unsigned int(5) projection\_type; // parameter 2  
}

### Semantics

is\_reverse indicates if the operation is a (forward) projection ('=0') or a reverse projection ('=1').

projection\_type has the same syntax and semantics as defined in ISO/IEC 23090-2.

## Packing

### Definition

Box Type: 'pack'  
Mandatory (per sample): No  
Quantity (per sample): Any  
Inputs: One

The Packing derivation transformation provides information for the process of transformation, resizing, and relocating of regions of an input track to form a derived track, according to a packing scheme such as the region-wise packing given in ISO/IEC 23090-2. An indicator is\_reverse is used to indicate whether the operation is a (forward) packing construction or reverse unpacking one.

This derivation transformation given below assumes the input and (derived) output tracks are an ISO/IEC 23090-2 compliant tracks. When the operation is a (forward) packing construction (is\_reverse == 0), the input track is a projected picture track, the output track is a packed picture track, and the region-wise packing structure RegionWisePackingStruct() is signaled to indicate the location, shape, and size of each packed region in the packed pictures. When the operation is a reverse packing (or unpacking) construction (is\_reverse == 0), the input track is a packed picture track which has a region-wise packing item property containing a RegionWisePackingStruct() structure, the output track is a projected picture track, and it is the region-wise packing structure RegionWisePackingStruct() in the input track is used to indicate the location, shape, and size of each packed region in the packed pictures.

### Syntax

aligned(8) class Packing   
extends VisualDerivationBase('pack', flags) {  
 bit(1) is\_reverse; // parameter 1  
 bit(7) reserved = 0; // not a parameter  
 if (is\_reverse == 0)   
 RegionWisePackingStruct();  
}

### Semantics

is\_reverse indicates if the operation is a (forward) packing ('=0') or a reverse packing ('=1').

RegionWisePackingStruct() has the same semantics as the metadata defined in ISO/IEC 23090-2.

## Viewport

### Definition

Box Type: 'vpot'  
Mandatory (per sample): No  
Quantity (per sample): Any  
Inputs: (1) visual track, (2) spatial region timed metadata track

The Viewport derivation transformation constructs (or extracts) viewport sample images from spherical sample images of an input omnidirectional video track, according to a viewport definition such as the one for a (timed) sphere region given in ISO/IEC 23090-2.

The input track for the corresponding image operation shall be an omnidirectional video track and a sphere region timed metadata track, such as the 'rosc' (sphere region) or 'rcvp' (recommended viewport) timed metadata track, with a 'cdsc' track reference to the video track.

The Viewport derivation transformation uses the sphere region metadata of the timed metadata track to extract viewport samples from the samples of the input video track. In other words, the sphere region metadata track is applied prescriptively to the video track that is the input entity for the Viewport derivation transformation . The output of the image operation contains only the sphere region specified by the sphere region metadata track.

### Syntax

aligned(8) class Viewport   
extends VisualDerivationBase('vpot', flags) {

}

# Derived Immersive Tracks

See also: <http://mpegx.int-evry.fr/software/MPEG/Systems/FileFormat/DerivedVis/issues/17>

## Motivation

A derived visual track is defined as a “video or picture track contains a timed sequence of derived samples”, where a derived sample as a “sample containing an ordered list of derivation operations”. Thus, any output track generated by a derived visual track from its input tracks will be either a video or picture track.

Therefore, derived visual tracks are not applicable to derivation or generation of output tracks that are “visual”, but not video or picture, tracks such as those volumetric visual tracks newly defined for “[Carriage of Visual Volumetric Video-based Coding Data](https://www.mpegstandards.org/standards/mpeg-i/carriage-of-visual-volumetric-video-based-coding/)” (carriage of V3C data), including carriage of Video-based Point Cloud Compression and Visual Volumetric Video-based Coding (MPEG-I part 5) and Immersive Video (MPEG-I part 12).

This proposal is to provide a definition of derived immersive tracks and a list of (3D, immersive) derivation transformations corresponding to the list of (2D, visual) derivation transformations currently provided

## Proposal text

A derived immersive track, like its derived visual track counterpart, describes a timed sequence of derived samples composed of an ordered list of derivation operations, each derivation operation applying a derivation transformation for the duration of the derived sample on an ordered list of visual inputs represented in the same presentation.

A derived immersive track shall be an immersive media track (with the the 'volv' handler type in the HandlerBox of the MediaBox as defined in ISO/IEC 14496‑12). A derived immersive track is identified by its containing sample entry of type 'ditk' DerivedImmersiveSampleEntry. Each sample described by a DerivedImmersiveSampleEntry is a derived sample.

A derived visual track shall include a TrackReferenceTypeBox with reference\_type equal to 'ditk' referring to all its input tracks. Each reference shall be one of:

1. the track\_ID of a track used by derived samples in the track, or, if unified IDs are in use as defined by ISO/IEC 14496‑12, a track\_group\_id;
2. the item\_ID of an image item, in the file-level MetaBox, used by derived samples in the track.

An ID value in the track references is resolved to a track\_ID whenever the file contains a track with such ID, is resolved to a track\_group\_id whenever unified IDs are in use and the file contains a track group with such ID, and is resolved to an item\_ID otherwise.

NOTE 1 A track\_ID may be an ID of a derived immersive track or a derived visual track.

If a referenced track is a member of an alternate group, or if the reference is to a track group, then the reader should pick a track from the group as the input to the derived immersive track. Similarly, if a referenced image item is a member of an alternate group, then the reader should pick an image item from the group as the input to the derived immersive track.

A derived sample contains an ordered list of the derivation operations to be performed, each derivation operation applying a derivation transformation on an ordered list of input tracks. The layer syntax element in TrackHeaderBox has no impact on ordering the input tracks for derived samples.

The four-character codes of derivation transformation from all derivation operations used by the derived samples in the track are listed in the DerivedImmersiveSampleEntry, and also default inputs and parameter values can be supplied there. A derived sample in the track may use all or some of the derivation operations listed in the linked DerivedImmersiveSampleEntry, but derived samples shall not use a derivation operation not listed in the sample entry.

The derived sample durations document the time over which the derivation represented by the ordered list of derivation operations is active. Therefore, the number of samples defined in a derived immersive track does not necessarily match 1:1 with the number of input image items or samples of input tracks that are being transformed. A single derivation duration may span multiple samples in the source track(s), and also derivation transformations in derived samples may have 'internal time structure' (e.g. a cross-fade) so the immersive media may change during the sample duration. This is in contrast to 'classic video'.

Derived immersive tracks do not use composition-time re-ordering on input tracks. They operate on the composition timeline (i.e. before the application of edit lists) of their input tracks (including on derived immersive tracks when used as visual inputs). However, the input tracks shall not have edit lists. Any edit lists of the input tracks shall be ignored if present.

NOTE 2 When time-alignment adjustment between input tracks is needed, signed composition offsets in input tracks may be used.

NOTE 3 A derived immersive track may have an edit list; thus, a derived immersive track using the identity transform, and with an edit-list, can provide a visual output that is a temporal re-mapping of the input track.

The inputs for a derivation operation in a derived sample can be either input image items from file-level MetaBox or intervals (possibly spanning multiple samples) of input video tracks, image sequence tracks, or metadata tracks, the visual output of a preceding derivation operation or a default input fill picture.

Transformative item properties or transformations (e.g. clean aperture, track matrix etc…) associated with input image items or samples of input tracks are always applied before performing the derivation operation.

NOTE 4 If a derived sample needs to refer to one explicit sample value in a referred track (other than the time-aligned sample value), an item may be created and referred to that has the same data as the desired sample value.

The visual inputs in a derived sample shall have consistent pixel aspect ratio and bit depth. The input image items, samples of input tracks or derived samples may have various width and height as well as depth (for 3D cases). When differences in width, height and depth result in pixels that never get ‘painted’ by a derivation operation, those empty pixels are filled according to the value of default\_derivation\_input parameter signalled in DerivedVisualTrackConfigRecord (black, white or grey pixels). When differences in width, height and depth result in pixels that end up outside the visual output size by a derivation operation, those pixels are cropped. This default behaviour may be overridden by derivation operation specifications.

A derived sample is reconstructed by performing the specified derivation operations in sequence. Some derivation operations can be marked as non-essential which indicates that the derivation operation may be skipped by the reader. However, the operations marked as essential shall be used in order to obtain a valid derived sample.

When more than one derivation operation is listed in a derived sample, the derivation operation that is not first in the list may include the output result (e.g. the visual output) of any of the previous derivation operations, only new inputs, or a combination of both.

In many cases the source tracks pointed to by the 'ditk' track reference are not intended for display. When a track is not intended for display, track\_in\_movie shall be equal to 0 for that track.

The visual output of a derived sample is the output from the last derivation operation in the sample. If there is no derivation operation, an empty derived sample (i.e. sample size of 0) is equivalent to an empty edit, i.e., there is no visual output from the derived immersive track at that time.

Using derived immersive tracks, it is possible to build either a chain of derivation operations on one single derived immersive track or a hierarchy of multiple derived immersive as well as visual tracks when they are used as an input to another derived immersive or visual track. The latter should only be used when each derived immersive or visual track in the hierarchy is also needed on its own.

1. **Derivation transformations**
   1. **Overview**

**Table 1** is the list of basic derivation transformations defined for derived immersive tracks. Further derivation transformations may be defined in derived specifications or added in future editions. The four-character codes of derivation transformations can and should be registered with the maintenance agency as defined in ISO/IEC 14496‑12. Vendor-specific derivation transformation may also be defined by using the code 'uuid' and an extended type UUID identifying the vendor-specific derivation transformation.

**Table 1: List of basic derivation transformations**

|  |  |  |
| --- | --- | --- |
| ***Operation*** | ***Inputs*** | ***Parameters*** |
| Identity | 1 | (none) |
| sRGB Fill | 0 | 1. Red fill 2. Green fill 3. Blue fill 4. Opacity 5. Output width 6. Output height 7. Output depth |
| Dissolve | 2 (from, to) | 1. Initial proportion 2. Final proportion |
| Crop | 1 | 1. Width numerator 2. Width denominator 3. Height numerator 4. Height denominator 5. Depth numerator 6. Depth denominator 7. X offset numerator 8. X offset denominator 9. Y offset numerator 10. Y offset denominator 11. Z offset numerator 12. Z offset denominator |
| Rotate | 1 | * X-Y Angle (0º, 90º, 180º, 270º) * X-Z Angle (0º, 90º, 180º, 270º) * Y-Z Angle (0º, 90º, 180º, 270º) |
| Mirror | 1 | 1. Horizontal, Vertical or Stacked mirroring |
| Scaling | 1 | 1. Target width numerator 2. Target width denominator 3. Target height numerator 4. Target height denominator 5. Target depth numerator 6. Target depth denominator |
| Region-of-interest | 2 (immersive visual track, ROI metadata) | (none) |
| Grid composition | N | 1. Output width 2. Output height 3. Output depth 4. Rows 5. Columns 6. Layers |
| Overlay composition | 2 (input, backdrop) | 1. horizontal\_offset 2. vertical\_offset 3. depth\_offset |

* 1. **Identity**
     1. **Definition**

Box Type: 'idtt'  
Mandatory (per sample): No  
Quantity (per sample): Zero or one  
Inputs: One

The Identity derivation transformation reproduces the immersive visual input. When a derivation operation with the Identity derivation transformation is present in a derived sample, no other derivation operations should be present in the same derived sample.

* + 1. **Syntax**

aligned(8) class Identity()  
extends VisualDerivationBase ('idtt', flags) {  
}

* 1. **sRGB Fill**
     1. **Definition**

Box Type: 'cfil'  
Mandatory (per sample): No  
Quantity (per sample): Any  
Inputs: None

The SRGBFill derivation transformation generates an immersive visual output of a single colour with visual size output\_width, output\_height and output\_depth pixels.

* + 1. **Syntax**

aligned(8) class SRGBFill   
extends VisualDerivationBase ('cfil', flags) {  
 unsigned int(16) red\_fill\_value = 0; // parameter 1  
 unsigned int(16) green\_fill\_value = 0; // parameter 2  
 unsigned int(16) blue\_fill\_value = 0; // parameter 3  
 unsigned int(16) opacity\_value = 65535; // parameter 4  
 unsigned int(32) output\_width = sample entry width; // parameter 5  
 unsigned int(32) output\_height = sample entry height; // parameter 6  
 unsigned int(32) output\_depth = sample entry depth; // parameter 7  
}

* + 1. **Semantics**

red\_fill\_value indicates the pixel value for the red channel according to sRGB colour space as defined in IEC 61966‑2‑1.

green\_fill\_value indicates the pixel value for the green channel according to sRGB colour space as defined in IEC 61966‑2‑1.

blue\_fill\_value indicates the pixel value for the blue channel according to sRGB colour space as defined in IEC 61966‑2‑1.

opacity\_value indicates the opacity value ranging from 0 (fully transparent) to 65535 (fully opaque).

output\_width, output\_height, output\_depth specify the width, height and depth, respectively, of the immersive visual output in pixels.

* 1. **Dissolve**
     1. **Definition**

Box Type: 'dslv'  
Mandatory (per sample): No  
Quantity (per sample): Any  
Inputs: (1) starting immersive visual input A, (2) ending immersive visual input B

The Dissolve derivation transformation provides smooth blending of two immersive visual inputs gradually fading from the first immersive visual input to the second immersive visual input. The visual output co-located pixel value, O(x,y,z) is computed by the weighted summation of the two immersive visual inputs where the weights are time-based transitions of the proportions provided by start\_weight and end\_weight. In the following equations, T transitions linearly from 0 at the time of the derived sample start to 1 at the time of the derived sample end.

The sizes of the visual inputs are normalized to the size of the larger one before the operation.

O(x,y,z) = A(x,y,z) \* (T \* (end\_weight – start\_weight) + start\_weight) / 256

+ B(x,y,z) \* ((1-T) \* (end\_weight - start\_weight) + start\_weight) / 256,

where '/' is a division by truncation. O(x,y,z) is saturated to the dynamic range of the pixel values.

* + 1. **Syntax**

aligned(8) class Dissolve   
extends VisualDerivationBase ('dslv', flags) {  
 unsigned int(8) end\_weight = 1; // parameter 1  
 unsigned int(8) start\_weight = 255; // parameter 2  
}

* + 1. **Semantics**

end\_weight, start\_weight A value between 1 and 255 that gives the initial and final weights to be multiplied to collocated input pixels. Default value for end\_weight is 1, and default value for start\_weight is 255. The value 0 is reserved.

* NOTE The identity derivation transformation can be used in a preceding or following derived sample, if a copy of either immersive visual input is needed.
  1. **Crop**
     1. **Definition**

Box Type: 'crop'  
Mandatory (per sample): No  
Quantity (per sample): Any  
Inputs: One

The SampleCrop derivation transformation defines a cropping transformation of the immersive visual input.

* + 1. **Syntax**

aligned(8) class SampleCrop   
extends VisualDerivationBase ('crop', flags) {  
 unsigned int(32) cleanApertureWidthN = width of the visual input;  
 // parameter 1  
 unsigned int(32) cleanApertureWidthD = 1; // parameter 2  
  
 unsigned int(32) cleanApertureHeightN = height of the visual input;  
 // parameter 3  
 unsigned int(32) cleanApertureHeightD = 1; // parameter 4  
  
 unsigned int(32) cleanApertureDepthN = depth of the visual input;  
 // parameter 5  
 unsigned int(32) cleanApertureDepthD = 1; // parameter 6

unsigned int(32) horizOffN = 0; // parameter 7  
 unsigned int(32) horizOffD = 1; // parameter 8  
  
 unsigned int(32) vertOffN = 0; // parameter 9  
 unsigned int(32) vertOffD = 1 // parameter 10

unsigned int(32) depOffN = 0; // parameter 11  
 unsigned int(32) depOffD = 1 // parameter 12  
}

* + 1. **Semantics**

The semantics of the syntax elements within the SampleCrop derivation transformation are the same as those specified for the syntax elements of CleanApertureBox as defined in ISO/IEC 14496‑12.

* 1. **Rotation**
     1. **Definition**

Box Type: 'srot'  
Mandatory (per sample): No  
Quantity (per sample): Any  
Inputs: One

The SampleRotation derivation transformation rotates the immersive visual input in anti-clockwise directions in units of 90 degrees in XY, XZ and YZ planes.

* + 1. **Syntax**

aligned(8) class SampleRotation   
extends VisualDerivationBase ('srot', flags) {  
 unsigned int (2) reserved = 0; // not a parameter  
 unsigned int (2) angleXY = 0; // parameter 1  
 unsigned int (2) angleXZ = 0; // parameter 2  
 unsigned int (2) angleYZ = 0; // parameter 3  
}

* + 1. **Semantics**

angleXY \* 90, angleXZ \* 90 and angleYZ \* 90 specify the angles (in anti-clockwise direction) in units of degrees, in the XY, XZ and YZ planes, respectively.

* 1. **Mirror**
     1. **Definition**

Box Type: 'smir'  
Mandatory (per sample): No  
Quantity (per sample): Any  
Inputs: One

The SampleMirror derivation transformation mirrors the immersive visual input about a vertical (Y), horizontal (X) or stacked (Z) axis.

* + 1. **Syntax**

aligned(8) class SampleMirror   
extends VisualDerivationBase ('smir', flags) {  
 unsigned int (6) reserved = 0; // not a parameter  
 unsigned int (2) axis = 0; // parameter 1  
}

* + 1. **Semantics**

axis specifies a vertical (axis = 0), horizontal (axis = 1) or stacked (axis = 2) axis for the mirroring transformation. Note that vertical mirror is a left-right one, horizontal mirror is a top-down one, and a stacked mirror is a front-back one.

* 1. **Scaling**
     1. **Definition**

Box Type: 'sscl'  
Mandatory (per sample): No  
Quantity (per sample): Any  
Inputs: One

The SampleScaling derivation transformation scales the visual input to a target size.

The computation of the target size from the syntax elements is the same as the one specified for ImageScaling as defined in ISO/IEC 23008‑12

* + 1. **Syntax**

aligned(8) class SampleScaling   
extends VisualDerivationBase ('sscl', flags) {  
 unsigned int (16) target\_width\_numerator = 1; // parameter 1  
 unsigned int (16) target\_width\_denominator = 1; // parameter 2  
 unsigned int (16) target\_height\_numerator = 1; // parameter 3  
 unsigned int (16) target\_height\_denominator = 1; // parameter 4  
 unsigned int (16) target\_depth\_numerator = 1; // parameter 5  
 unsigned int (16) target\_depth\_denominator = 1; // parameter 6  
}

* + 1. **Semantics**

The semantics of the syntax elements within the SampleScaling derivation transformation are the same as those specified for the syntax elements of ImageScaling as defined in ISO/IEC 23008‑12.

* 1. **Region of interest (ROI) selection**
     1. **Definition**

Box Type: '3dcc'  
Mandatory (per sample): No  
Quantity (per sample): Any  
Inputs: (1) immersive visual track, (2) ROI timed metadata track

The number of inputs shall be equal to 2 for this derivation transformation, and the inputs for this derivation transformation shall be a visual track and a ROI timed metadata track carrying 3D Cartesian coordinates as defined in ISO/IEC 23001‑10.

The ROISelection derivation transformation uses the 3D Cartesian coordinates carried in the second input track to crop the samples of the first input track. In other words, the ROI timed metadata track carrying 3D Cartesian coordinates is applied prescriptively to the visual track that is the input for the ROI selection derivation transformation. The visual output of the derivation transformation contains only the rectangle specified by the 3D Cartesian coordinates of the ROI timed metadata track.

* + 1. **Syntax**

aligned(8) class ROISelection   
extends VisualDerivationBase ('3dcc', flags) {  
}

* 1. **Grid composition**
     1. **Definition**

Box Type: 'gdcp'  
Mandatory (per sample): No  
Quantity (per sample): Any  
Inputs: Exactly rows\*columns. First visual input is the top-left cell of the grid, last visual input is the bottom-right cell of the grid

The GridComposition derivation transformation provides a composition of immersive visual inputs in a given 3D grid order.

The visual inputs are inserted in row-major order, top-row first, left to right, in the order they are listed as the derivation operation inputs. The number of visual inputs shall be equal to rows\*columns\*layers. All immersive visual inputs (possibly after being resized) shall have exactly the same width, height and depth, width, height and depth. The immersive visual inputs, when composed together, completely “cover” the immersive visual output of the transformation according to the 3D grid, where width\*columns is equal to output\_width, height\*rows is equal to output\_height, and depth\*layers is equal to output\_depth. In other words, the visual output of the transformation is formed by tiling the visual inputs into a 3D grid with a column width equal to width and a row height equal to height, and a layer depth equal to depth, without any gap or overlap.

* + 1. **Syntax**

aligned(8) class GridComposition   
extends VisualDerivationBase ('gdcp', flags) {  
 unsigned int(8) rows\_minus\_one = 0; // parameter 1  
 unsigned int(8) columns\_minus\_one = 0; // parameter 2  
 unsigned int(8) layers\_minus\_one = 0; // parameter 3  
 unsigned int(32) output\_width = sample entry width; // parameter 4  
 unsigned int(32) output\_height = sample entry height; // parameter 5  
 unsigned int(32) output\_depth = sample entry depth; // parameter 6  
}

* + 1. **Semantics**

output\_width, output\_height, output\_depth specify the width, height and depth, respectively, of the reconstructed immersive visual output on which the immersive visual inputs are placed.

rows\_minus\_one, columns\_minus\_one, layers\_minus\_one specify the number of rows, columns and layers in the 3D grid; the value is one less than the number of rows, columns or layers, respectively. Visual inputs populate the top row first, followed by the second and following, in the listing order of the derivation operation inputs.

* 1. **Overlay composition**
     1. **Definition**

Box Type: 'sovl'  
Mandatory (per sample): No  
Quantity (per sample): Any  
Inputs: (1) overlay immersive visual input (2) backdrop visual input

The OverlayComposition derivation transformation provides a composition of an immersive visual input over another visual input representing the backdrop.

The overlay immersive visual input is copied over the backdrop visual input at the horizontal\_offset, vertical\_offset, stacked\_offset offsets. The size of the reconstructed visual output is equal to the size of the backdrop visual input. Pixels of the overlay visual input that end up outside the backdrop visual input size by the copy operation are cropped. When a visual input has an associated alpha plane, alpha blending shall be performed on the visual input.

A chain of derivation operations can be used to create a sequence of multiple OverlayComposition derivation transformations. This allows overlaying multiple visual inputs over an initial backdrop visual input in sequence order. The backdrop visual input of the first OverlayComposition derivation transformation in the sequence represents the initial backdrop visual input. In subsequent OverlayComposition derivation transformation in the sequence, the backdrop visual input should designate the visual output of the preceding OverlayComposition derivation transformation in the sequence. Therefore, the first OverlayComposition derivation transformation in the sequence represents the overlay composition of the bottom-most overlay visual input over the initial backdrop visual input. The last OverlayComposition derivation transformation in the sequence represents the overlay composition of the top-most overlay visual input over the reconstructed visual output of preceding overlay derivation transformations in the sequence.

* NOTE The visual output of a sRGB fill derivation transformation may be used as an initial backdrop visual input.
  + 1. **Syntax**

aligned(8) class OverlayComposition   
extends VisualDerivationBase ('sovl', flags) {  
 signed int(32) horizontal\_offset = 0; // parameter 1  
 signed int(32) vertical\_offset = 0; // parameter 2  
 signed int(32) stacked\_offset = 0; // parameter 3  
}

* + 1. **Semantics**

horizontal\_offset, vertical\_offset, stacked\_offset specify the offset, from the top-left-front corner of the backdrop visual input, to which the overlay immersive visual input is located. Pixel locations with a negative offset value are not included in the reconstructed immersive visual output. Horizontal pixel locations greater than or equal to the width of the backdrop visual input are not included in the reconstructed immersive visual output. Vertical pixel locations greater than or equal to the height of the backdrop visual input are not included in the reconstructed immersive visual output. Stacked pixel locations greater than or equal to the depth of the backdrop visual input are not included in the reconstructed immersive visual output.