ISO/IEC JTC 1/SC 29/WG 03 N0619

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

**Document type:** Output Document

**Title:** Technology under Consideration on ISO/IEC 23008-12

**Status:** Approved

**Date of document:** 2022-07-29

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

**No. of pages:** 12 (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 N0619**

**July 2022, Virtual**

|  |  |
| --- | --- |
| **Title** | **Technology under Consideration on ISO/IEC 23008-12** |
| **Source** | **WG 03, MPEG Systems** |
| **Status** | **Approved** |
| **Serial Number** | **21748** |

# Abstract

This document collects following candidate technologies for the High Efficiency Image File Format (HEIF) (ISO/IEC 23008-12).

Table of Contents

[Abstract 1](#_Toc109996517)

[1. Carriage of Text Items 2](#_Toc109996518)

[**1** **Introduction** 2](#_Toc109996519)

[**2** **Proposal** 3](#_Toc109996520)

[2. Region annotations for image sequence or video tracks 7](#_Toc109996521)

[**1** **Proposal (from m60304, MPEG#139, Issue#76)** 7](#_Toc109996522)

[**1.1** **Overview** 7](#_Toc109996523)

[**1.2** **Proposals** 8](#_Toc109996524)

[**2** **Proposal (m59508, MPEG#138)** 12](#_Toc109996525)

[**3** **Discussion** 15](#_Toc109996526)

[3. On carriage of camera properties 15](#_Toc109996527)

[**6.5.39** **Camera extrinsic matrix** 15](#_Toc109996528)

[**6.5.40** **Camera intrinsic matrix** 16](#_Toc109996529)

[4. Matrix-based transformation for image items 20](#_Toc109996530)

[5. Signaling for pre-derived coded image items 20](#_Toc109996531)

# Carriage of Text Items

*[ Ed. (FD):* Decision at MPEG#139 meeting: *“update TuC with contribution m60316]*

*[ Ed. (FD):* Open question from MPEG#137 meeting: *“we need to find a reasonably manageable/compact language: text/plain says too little (well, nothing) about styling, and full-on HTML seems a bit much”]*

Comments by proponents in m60316 at MPEG#139: *“We believe that in simple application uses cases such as memes the content author and the viewer would be interested in only the textual content and rather not on the styling aspect of it. In such uses cases a simple plain text item is sufficient and the styling and rendering operation is left to the viewer application”.*

Comments and questions on this proposal remain open in HEIF [Issue#77](http://mpegx.int-evry.fr/software/MPEG/Systems/FileFormat/HEIF/-/issues/77) on the MPEG GitLab.

1. **Introduction**

Creating images with textual content such as captioning of images, memes and so on is very common and is done globally using various editing tools. Figure 1, shows an example meme image with plain text.



**Figure 1: Example of an meme image with plain text.**

The ISO/IEC 14496-30 standard on Timed text and other visual overlays in ISO base media file format specifies the carriage of timed text and subtitle streams in ISO BMFF tracks. However, neither ISO/IEC 14496-30 standard nor the HEIF standard specify the carriage of text items associated with an image item.

In MPEG 136, Nokia proposed a new item type called the text item for the carriage of textual content associated with an image item in m58143. The ISOBMFF group noted the proposal and provided the following suggestion:

We need to find a middle ground between plain text with no styling and the full complexity of something like HTML or SVG. We'd like to use but not overload the current technologies (such as overlay, MIME typed items, and so on).

Based on the above suggestions, this contribution proposes a mime type item for renderable text. The data in the mime type item is a renderable text.

1. **Proposal**

*Add following definitions in clause 3*

**3.1.X**

**text item**

*item* (3.1.27) whose data is the textual data.

**3.1.Y**

**renderable text item**

*a text item* (3.1.X) that includes possibly size, position, direction, language, font and styling and whose processing produces an output which can be visually rendered.

**3.1.Z**

**font item**

*item* (3.1.27) whose data is the fonts

*Update the subclause 6.5.2.1 as follows (changes highlighted in yellow)*

**6.5.2.1 Image spatial extents**

**6.5.2.2 Definition**

|  |  |
| --- | --- |
| Box type: | 'ispe' |
| Property type: | Descriptive item property |
| Container: | ItemPropertyContainerBox |
| Mandatory (per item): | Yes |
| Quantity (per item): | One |
|  |  |

The ImageSpatialExtentsProperty documents the width and height of the associated image item. Every image item shall be associated with one property of this type, prior to the association of all transformative properties.

The ImageSpatialExtentsProperty may be associated with items whose output can be visually rendered (e.g., renderable text items). When ImageSpatialExtentsProperty is associated with items whose output can be visually rendered, they document the visually rendered width and height of the data which is output from the associated item.

**6.5.2.3 Syntax**

aligned(8) class ImageSpatialExtentsProperty  
extends ItemFullProperty('ispe', version = 0, flags = 0) {  
 unsigned int(32) image\_width;  
 unsigned int(32) image\_height;  
}

**6.5.2.4 Semantics**

image\_width specifies the width of the reconstructed image in pixels, as specified in 6.3.

image\_height specifies the height of the reconstructed image in pixels, as specified in 6.3.

NOTE Item properties, such as decoder configuration or layer selection, can affect the reconstructed image. As a consequence, the width and height of the reconstructed image depend on the presence and content of such properties.

When ImageSpatialExtentsProperty is associated with items whose output can be visually rendered, the image\_width and image\_height specifies the visually rendered width and height, respectively of the data which is output from the associated item.

*Add the following subclause in Clause 6*

**6.A Text item and Renderable text item**

**6.A.1 Definition**

A text item is an item with item\_type value set to 'mime' and the data in the text item is text, for example, ‘html’ or ‘plain text’. The content\_type in ItemInfoEntry of the ItemInfoBox is set equal to the mime type of the data in the text item. Example values for content\_type field may include ‘text/html’ for html formatted text or ‘text/plain’ for plain text.

The text item is associated with the image item on which the textual data is displayed/rendered using an item reference of type 'cdsc' from the text item to the image item. A text item shall be associated with multiple image items only when all the associated image items have the same size.

[NOTE: As an alternate to the use of the text item may be used as an overlay to the image item. However, this aspect needs to be further discussed]

The text item may be associated with the font item using an item reference of type 'font' from the text item to the font item. The font item carries the fonts used for rendering the text item.

The text item is associated with the ImageSpatialExtentsProperty which documents the visually rendered width and height of the data which is output from the text item.

The text item is associated with the TextLayoutProperty which documents the visually rendered size, position and language of the data which is output from the text item.

When a text item is not associated with any item or item property which documents possibly size, position, direction, language, font and styling for visual rendering, then, the data in the text item should contain the textual data together with possibly size, position, direction, language, font and styling for visual rendering of the text item and is called the renderable text item.

The renderable text data may be further encoded with either gzip or deflate or any other alogithm defined for content-encoding of Http/1.1. The encoding of renderable text data shall be defined by the content\_encoding parameter in ItemInfoEntry of the ItemInfoBox for the mime type text item.

If the renderable text data is encoded with any of the alogithm defined for content-encoding of Http/1.1, the data needs to be decoded before interpreting it as the mime type text item identified by the content\_type in ItemInfoEntry of the ItemInfoBox.

If the content\_encoding parameter in ItemInfoEntry of the ItemInfoBox has an empty string, then no content encoding is applied on the renderable text data.

The mime type item of renderable text only carries the text data required for rendering, however it does not provide any information on the display/layout conditions, for example the position, size and direction of the renderable text. Hence we propose a item property for the mime type text item which carries the information on the display/layout conditions.

[NOTE: As an alternate to the use of text layout property, the following approach may be used The ImageSpatialExtentsProperty to document the width and height.

The ImageOverlay to document the reference width and reference height and the position of the renderable text item

The ExtendedLanguageBox to document the language of the textual data. However, these aspect needs to be further discussed.

**Comments from proponents in m60316 at MPEG#139**: *With respect to using alternatives to the use of text layout property which documents the layout information of the associated text item. using alternatives such as ImageSpatialExtentsProperty and ImageOverlay will need extending the corresponding item properties to accommodate text items as currently they are limited to image items. For using the ExtendedLanguageBox a new item property needs to be defined as currently there are no corresponding data structure for items.*

*The text layout property documents all the required information in a single structure without the hassle of updating the specification text in multiple cases. Hence, we propose to include the text item and font item properties in the HEIF amendment*]

**6.B Text Layout Information**

**6.B.1 Definition**

Box type: 'txlo'

Property type: Descriptive item property

Container: ItemPropertyContainerBox

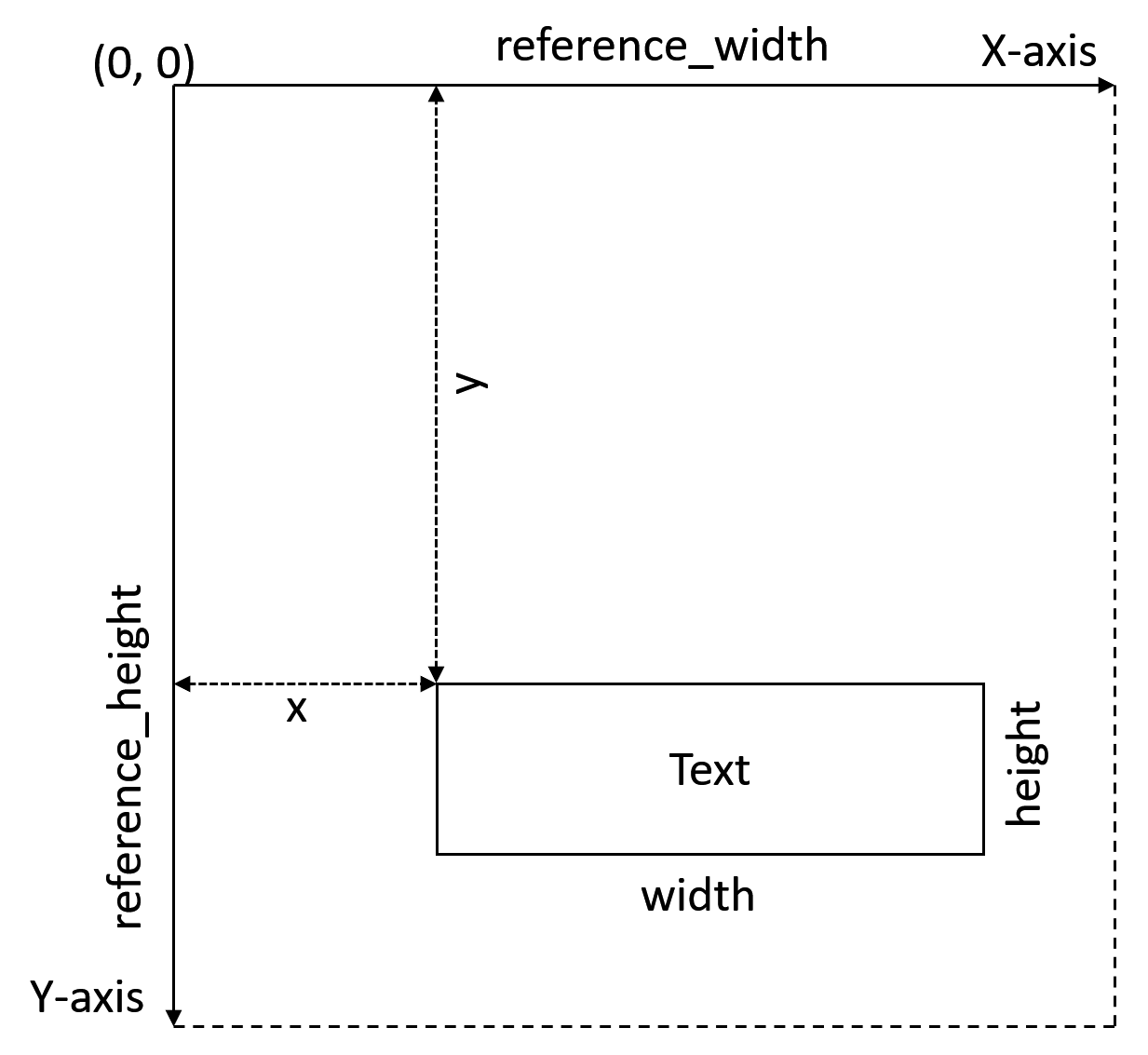
Mandatory (per item): No

Quantity (per item): One

The TextLayoutProperty documents the layout information of the associated text item. A text item shall be associated with one TextLayoutProperty prior to the association of all transformative properties.

The data in the TextLayoutProperty define the size, position and the language of the textual data to be displayed/rendered on the associated image item. The size and position information are used to display/render the textual data inside a reference space that is mapped to the image item with which the text item is associated after any transformative item property is applied to the image item.

The reference space is defined as a 2D coordinate system with the origin (0,0) located at the top-left corner and a maximum size defined by reference\_width and reference\_height; the x-axis is oriented from left to right and the y-axis from top to bottom. Figure 2, provides an illustration of text item in the reference space. The placement of textual data inside the associated image item is obtained after applying the implicit resampling caused by the difference between the size of the reference space and the size of the associated image item. If the text item has transformative item properties, then the implicit resampling shall be performed on the text item before the first of its transformative item properties is applied.



**Figure 2: An illustration of text item in reference space.**

**6.B.2 Syntax**

aligned(8) class TextLayoutProperty  
extends ItemFullProperty('txlo', version = 0, flags = 0) {  
 unsigned int (8) version = 0;   
 unsigned int (8) flags;   
 field\_size = ((flags & 1) + 1) \* 16;   
 unsigned int(field\_size) reference\_width;   
 unsigned int(field\_size) reference\_height;  
 signed int(field\_size) x;  
 signed int(field\_size) y;   
 utf8string language;   
}

**6.B.3 Semantics**

version shall be equal to 0.

(flags & 1) equal to 0 specifies that the length of the fields x, y, width, height is 16 bits. (flags & 1) equal to 1 specifies that the length of the fields x, y, width, height is 32 bits. The values of flags greater than 1 are reserved.

reference\_width, reference\_height specify, in pixel units, the width and height, respectively, of the reference space on which the text items are placed.

x, y specify the top, left corner of the text item relatively to the reference space. The value (x = 0, y = 0) represents the position of the top-left pixel in the reference space.

NOTE Negative values for the x or y fields enable to specify top-left corners that are outside the image. This can be useful for updating text items during the edition of an HEIF file.

language is a character string containing an RFC 5646 compliant language tag string, such as "en-US", "fr-FR", or "zh-CN“, representing the language of the text. When language is empty, the language is unknown/undefined.

**6.C Font item**

**6.C.1 Definition**

A font item is an item with the item\_type value set to 'mime' and the data in the font item are fonts for example ‘woff’ (Web Open Font Format) or ‘ttf’ (true type font). The content\_type in ItemInfoEntry of the ItemInfoBox is set equal to the mime type of the data in the font item. Example values for content\_type field may include ‘font/ttf’ for true type fonts or ‘font/woff’ for web open font format fonts.

The font item may be associated with the text item using an item reference of type 'font' from the text item to the font item.

The font data may be further encoded with either gzip or deflate or any other alogithm defined for content-encoding of Http/1.1. The encoding of font data shall be defined by the content\_encoding parameter in ItemInfoEntry of the ItemInfoBox for the font item.

If the font is encoded with any of the alogithm defined for content-encoding of Http/1.1, the data needs to be decoded before interpreting it as the mime type font item identified by the content\_type in ItemInfoEntry of the ItemInfoBox.

If the content\_encoding parameter in ItemInfoEntry of the ItemInfoBox has an empty string, then no content encoding is applied on the font data.

# Region annotations for image sequence or video tracks

*[Ed. (FD). At MPEG#139, region extrapolation has been proposed as in section 1 below]*

*[Ed. (FD). It has been decided at MPEG#138 to move Region Annotations for image sequence and video tracks as proposed in m59508 to HEIF CDAM,* ***except the interpolate flag that may need further thoughts*** *(the yellow highlighted syntax and semantics (*[*comment on MPEG Gitlab*](http://mpegx.int-evry.fr/software/MPEG/Systems/FileFormat/HEIF/-/issues/69#note_60556)*).]*

1. **Proposal (from m60304, MPEG#139, [Issue#76](http://mpegx.int-evry.fr/software/MPEG/Systems/FileFormat/HEIF/-/issues/76))**
   1. **Overview**

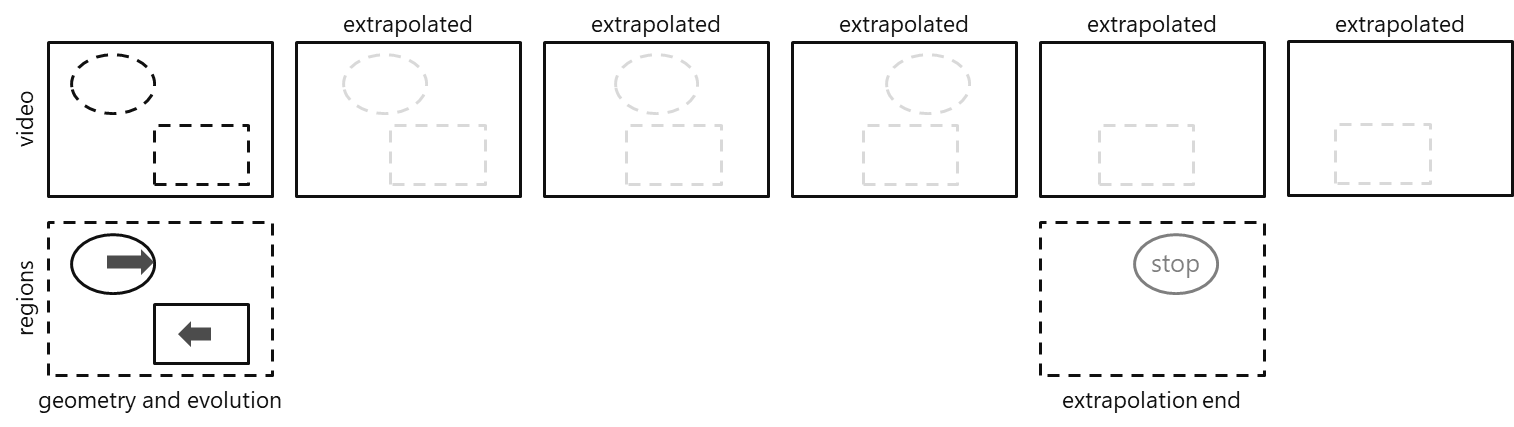


Figure 1: example region description for tracks using extrapolation

Figure 1 shows an example of describing several regions using extrapolation. The video track shown at the top contains two regions, an elliptic one and a rectangular one. The elliptic one is present in the four first samples of the video track and moves to the right of the image. The rectangular one is present in all the samples of the video track and moves to the left of the image.

The region track shown at the bottom describes these two regions. In a first sample, corresponding to the first sample of the video track, these two regions are described with their positions and sizes and the evolution of their respective positions and sizes. There are no region samples corresponding to the three following video samples. The region sample corresponding to the fifth video sample signals that the interpolation of the elliptic region ends.

* 1. **Proposals**

*Update the definition of a region track ( section 7.5.4.1) by adding the following paragraphs:*

The geometry of a region may be defined by specifying the shape, position and size of the region in a sample of the region track. The geometry of a region may also be defined as an initial geometry and its evolution over time by specifying the initial geometry of the region and its evolution in a sample of the region track.

The evolution of a region over time is optional. It can be represented by the evolution speed of some of its parameters inside the reference space. The evolution speed of the parameters is signaled using a scaling factor for increasing its precision. The parameters defining the evolution of a region depend on the geometry of the region as follows:

— When the geometry of a region is represented by a point, the evolution of the region is defined by the evolution of the position of this point.

— When the geometry of a region is represented by a rectangle or an ellipse, the evolution of the region is defined by the evolution of the position and the size of the rectangle or ellipse.

— When the geometry of a region is represented by a polygon or a polyline, the evolution of the region is defined by the evolution of the position of each point of the polygon or polyline. The number of points in the polygon or polyline doesn’t change.

— When the geometry of a region is represented by a mask, the evolution of the region is defined by the evolution of the position of the mask.

The evolution of a region stops when another sample contains a region with the same region identifier. The evolution of a region shall stop for each sync sample of the source track.

*Update the Sample format (section 7.5.4.2.1) with the following paragraph*

When the extrapolate flag is set to 1 for a region inside a sample of a region track, the region is an evolving region defined by an initial geometry and its evolution over time.

The value of each evolving parameter defining the geometry of the region at a given composition time *T* can be computed as follows:

where:

* *param0* is the initial value of the parameter as defined in the initial geometry of the region at time T0.
* *Δparam* is the evolution of the parameter as defined in the evolution of the region.
* *evolution\_scale* is a scaling factor for the evolution values equal to , where is the field\_size and is equal to ((RegionTrackConfigBox.field\_length\_size & 1) + 1) \* 16.
* *T0* is the composition time of the sample defining the evolving region.
* *ΔT* is the duration of the sample defining the evolving region.

*Update the syntax of Sample format (section 7.5.4.2.2) as follows*

aligned (8) class RegionSample {  
 unsigned int field\_size = ((RegionTrackConfigBox.field\_length\_size & 1) + 1) \* 16;  
// this is a temporary, non-parsable variable  
 unsigned int(32) region\_count;  
 for (r=0; r < region\_count; r++) {  
 unsigned int(32) region\_identifier;  
 unsigned int(8) geometry\_type;  
 unsigned int(1) extrapolate;  
 unsigned int(7) reserved;  
 if (geometry\_type == 0) {  
 // point  
 signed int(field\_size) x;  
 signed int(field\_size) y;  
 if (extrapolate == 1) {  
 signed int(field\_size) delta\_x;  
 signed int(field\_size) delta\_y;  
 } }  
 else if (geometry\_type == 1) {  
 // rectangle  
 signed int(field\_size) x;  
 signed int(field\_size) y;  
 unsigned int(field\_size) width;  
 unsigned int(field\_size) height;  
 if (extrapolate == 1) {  
 signed int(field\_size) delta\_x;  
 signed int(field\_size) delta\_y;  
 signed int(field\_size) delta\_width;  
 signed int(field\_size) delta\_height;  
 }  
 }  
 else if (geometry\_type == 2) {  
 // ellipse  
 signed int(field\_size) x;  
 signed int(field\_size) y;  
 unsigned int(field\_size) radius\_x;  
 unsigned int(field\_size) radius\_y;  
 if (extrapolate == 1) {  
 signed int(field\_size) delta\_x;  
 signed int(field\_size) delta\_y;  
 signed int(field\_size) delta\_radius\_x;  
 signed int(field\_size) delta\_radius\_y;  
 }  
 }  
 else if (geometry\_type == 3 || geometry\_type == 6) {  
 // polygon or polyline  
 unsigned int(field size) point\_count;  
 for (i=0; i < point\_count; i++) {  
 signed int(field\_size) px;  
 signed int(field\_size) py;  
 }  
 if (extrapolate == 1) {  
 for (i=0; i < point\_count; i++) {  
 signed int(field\_size) delta\_px;  
 signed int(field\_size) delta\_py;  
 }  
 }  
 }  
 else if (geometry\_type == 4) {  
 // referenced mask  
 signed int(field\_size) x;  
 signed int(field\_size) y;  
 unsigned int(field\_size) width;  
 unsigned int(field\_size) height;  
 unsigned int(field\_size) track\_mask\_idx;  
 if (extrapolate == 1) {  
 signed int(field\_size) delta\_x;  
 signed int(field\_size) delta\_y;  
 }  
 }  
 else if (geometry\_type == 5) {  
 // inline mask  
 signed int(field\_size) x;  
 signed int(field\_size) y;  
 unsigned int(field\_size) width;  
 unsigned int(field\_size) height;  
 unsigned int(8) mask\_coding\_method;  
 if (mask\_coding\_method != 0)  
 unsigned int(32) mask\_coding\_parameters;  
 bit(8) data[];  
 if (extrapolate == 1) {  
 signed int(field\_size) delta\_x;  
 signed int(field\_size) delta\_y;  
 }  
 }  
 else if (geometry\_type == 7) {  
 // empty region  
 }  
 }  
}

*Update the semantics of Sample format (section 7.5.4.2.3) with the following text:*

7: the region is an empty region used for signalling the end of the evolution of a previous region with the same region identifier.

Other values are reserved.

extrapolate is a flag indicating whether the geometry changes of the region are specified or not. When equal to 0, it indicates that no geometry changes are specified for the region. When equal to 1, it indicates that both the geometry and the geometry changes are specified for the region.

(…)

evolution\_scale is the scaling factor for the specification of the evolution values, equal to , where is the field\_size and is equal to ((RegionTrackConfigBox.field\_length\_size & 1) + 1) \* 16.

delta\_x, delta\_y specify, in 1/evolution\_scale units of the reference space, the evolution of the x and y fields for the region.

delta\_width, delta\_height specify, in 1/evolution\_scale units of the reference space the evolution of the width and height fields for the region.

delta\_radius\_x, delta\_radius\_y specify, in 1/evolution\_scale units of the reference space the evolution of the radius\_x and radius\_y fields for the region.

delta\_px, delta\_py specify, in 1/evolution\_scale units of the reference space the evolution of the px, py fields for a point of the region.

1. **Proposal (m59508, MPEG#138)**

*[Ed. (FD)This TuC only contains parts requiring further discussion (the interpolate flag in sample format for region tracks)]The HEIF CDAM defines the region track*

**X.X Region track and region annotations for an image sequence or video track**

**X.X.1 General**

A metadata track with a sample entry 'rgan' is a region track whose samples define one or more regions inside images carried in samples of an associated image sequence or video track (also denoted source track in the following).

*(… see HEIF CDAM for full text… )*

**X.X.3 Sample format**

**X.X.3.1 Definition**

This subclause defines the sample format for region track. A sample of a region track defines one or more regions.

**X.X.3.2 Syntax**

aligned (8) class RegionSample {  
 unsigned int field\_size = ((RegionTrackConfigBox.field\_length\_size & 1) + 1) \* 16;   
// this is a temporary, non-parsable variable  
 unsigned int(7)reserved;  
 unsigned int(1)interpolate;  
 unsigned int(16) region\_count;  
 for (r=0; r < region\_count; r++) {  
 unsigned int(32) region\_identifier;  
 unsigned int(8) geometry\_type;  
 if (geometry\_type == 0) {  
 // point  
 signed int(field\_size) x;  
 signed int(field\_size) y;  
 }  
 else if (geometry\_type == 1) {  
 // rectangle  
 signed int(field\_size) x;  
 signed int(field\_size) y;  
 unsigned int(field\_size) width;  
 unsigned int(field\_size) height;  
 }  
 else if (geometry\_type == 2) {  
 // ellipse  
 signed int(field\_size) x;  
 signed int(field\_size) y;  
 unsigned int(field\_size) radius\_x;  
 unsigned int(field\_size) radius\_y;  
 }  
 else if (geometry\_type == 3 || geometry\_type == 6) {  
 // polygon or polyline  
 unsigned int(field size) point\_count;  
 for (i=0; i < point\_count; i++) {  
 signed int(field\_size) px;  
 signed int(field\_size) py;  
 }  
 }  
 else if (geometry\_type == 4) {   
 // referenced mask  
 signed int(field\_size) x;  
 signed int(field\_size) y;  
 unsigned int(field\_size) width;  
 unsigned int(field\_size) height;  
 unsigned int(field\_size) track\_mask\_idx;  
 }  
 else if (geometry\_type == 5) {   
 // inline mask   
 signed int(field\_size) x;  
 signed int(field\_size) y;  
 unsigned int(field\_size) width;  
 unsigned int(field\_size) height;  
 unsigned int(8) mask\_coding\_method;   
 if (mask\_coding\_method != 0)   
 unsigned int(32) mask\_coding\_parameters;  
 bit(8) data[];  
 }  
 }  
}

**X.X.3.3 Semantics**

interpolate indicates the continuity in time of the successive samples. When true, the application may linearly interpolate values of the region geometries between the previous sample and the current sample. When false, there shall not be any interpolation of values between the previous and the current samples.

NOTE 1 When using interpolation, it is expected that the interpolated samples match the presentation time of the samples in the referenced source track. For instance, for each video sample of a video track, one interpolated region sample is calculated.

region\_count specifies the number of regions defined in the sample.

region\_identifier specifies the identifier of the region.

geometry\_type specifies the type of the geometry of a region. The following values for geometry\_type are defined:

0: the region is described as a point.

1: the region is described as a rectangle.

2: the region is described as an ellipse.

3: the region is described as a polygon.

4: the region is described as a mask defined in a referenced image item or in a sample of a referenced track.

5: the region is described as a mask defined inside the data of this sample.

6: the region is described as a polyline.

Other values are reserved.

x, y specify the coordinates of the point composing the region relatively to the reference space when its geometry is a point. x, y specify the top, left corner of the region relatively to the reference space when its geometry is a rectangle or a mask. x, y specify the centre of the region relatively to the reference space when its geometry is an ellipse. The value (x = 0, y = 0) represents the position of the top-left pixel in the reference space.

NOTE 2 Negative values for the x or y fields enable to specify points, top-left corners, and/or centres that are outside the image. This can be useful for updating region annotations during edition.

width, height specify, relatively to the reference space, the width and the height of the region when its geometry is a rectangle or a mask. When geometry\_type equals 4, the value 0 indicates that the corresponding width or height value is provided by the ImageSpatialExtentsProperty associated with the item containing the mask or the width or height of the track containing the mask. When geometry\_type does not equal 4, the value 0 is reserved.

radius\_x specifies, relatively to the reference space, the radius on the x-axis of the region when its geometry is an ellipse.

radius\_y specifies, relatively to the reference space, the radius on the y-axis of the region when its geometry is an ellipse.

point\_count is the number of points contained in a polygon or a polyline.

NOTE 3 A polygon specifying the geometry of a region is always closed and therefore there is no need to repeat the first point of the polygon as the ending point of the polygon.

px, py specify the coordinates of the points composing the polygon or the polyline relatively to the reference space. The value (px = 0, py = 0) represents the position of the top-left pixel in the reference space.

track\_mask\_idx specifies the index of the track reference of type 'mask' referring to the track from which to retrieve the mask to apply. The sample in that track from which mask data is retrieved is the one that is temporally aligned with the current sample in the source track or the nearest preceding one in the media presentation timeline. The first track reference has the index value 1; the value 0 is reserved.

mask\_coding\_method indicates the coding method applied on the mask contained in data. The following values are defined:

0: No mask encoding scheme is applied.

1: Mask is compressed with deflate() as defined in IETF RFC 1951.

Other values are reserved.

mask\_coding\_parameters indicates additional encoding parameters needed for successfully processing the coded mask data. When mask\_coding\_method is equal to 1, mask\_coding\_parameters indicates the number of bytes in the coded mask array data. The value of mask\_coding\_parameters is reserved when the value of mask\_coding\_method is greater than 1.

data contains the coded or uncompressed representation of a mask that contains the pixels for an inline mask in raster-scan order. Each pixel is represented using a single bit and 8 pixels are packed in one byte. Byte packing shall be in big-endian order. No padding shall be put at the end of each line if the width of the mask is not a multiple of 8 pixels. Only the last data byte shall be padded with bits set to 0.

1. **Discussion**

About the interpolate flag: The purpose is to avoid declaring a sample in the region track for each sample of the media track when regions are moving linearly between two positions. Imagine a sample A in the region track with a region at a starting position A and this region is moving linearly to the arrival position B nine samples later. Instead of declaring ten samples in the region track, you can only declare two samples, sample A with a duration corresponding to nine samples in the media track, followed by sample B providing the arrival position B. We should clarify that since the interpolate flag applies to all regions in the sample, the number of regions shall be the same in sample A and B.

# On carriage of camera properties

*[Ed. (FD). At MPEG#139, there is also an open question on using the ExtrinsicCameraParametersBox 'ecam' and IntrinsicCameraParametersBox 'icam' specified in ISO/IEC 14996-15 for the storage of camera extrinsic and intrinsic parameters for image items*.*]*

More details on the related discussion can be found in [Issue#78](http://mpegx.int-evry.fr/software/MPEG/Systems/FileFormat/HEIF/-/issues/78).

* + 1. **Camera extrinsic matrix**

*[ Ed. (FD): Decisions at MPEG#139: Add in the TuC:*

*The information on the cameras GPS position and alignment to geomagnetic north as part of the ViewpointGpsPositionStruct and ViewpointGeomagneticInfoStruct, respectively, under version > 0 of the cameraExtrinsicProperty.*

Text for camera extrinsic matrix is in HEIF potential improvements of CDAM MDS21765\_WG03\_N00636*]*

Proposed additional parameters to the syntax:

* + - 1. **Syntax**

aligned(8) class CameraExtrinsicsProperty  
extends ItemFullProperty('cmex', **version**, flags) {  
  
 (… parameters from CDAM …)  
  
 if (version >=1 ) {   
 ViewpointGpsPositionStruct() vwpt\_gps\_pos;  
 ViewpointGeomagneticInfoStruct() vwpt\_geomagnetic\_info;   
 }  
}

* + - 1. **Semantics (proposed additional parameters)**

vwpt\_gps\_pos is defined in ISO/IEC 23090-7 and indicates the GPS position information of the camera.

vwpt\_geomagnetic\_info is defined in ISO/IEC 23090-7 and indicates the geomagnetic position information of the camera.

* + 1. **Camera intrinsic matrix**
       1. **Definition**

|  |  |
| --- | --- |
| Box type: | 'cmin' |
| Property type: | Descriptive item property |
| Container: | ItemPropertyContainerBox |
| Mandatory (per item): | No |
| Quantity (per item): | Zero or one |
|  |  |

The CameraIntrinsicsProperty descriptive item property describes the characteristics of the camera that captured the associated image item.

One general form of specifying the intrinsics matrix for a pinhole camera is as follows:

|  |  |  |
| --- | --- | --- |
| *fx* | *s* | *cx* |
| *0* | *fy* | *cy* |
| *0* | *0* | *1* |

|  |  |  |
| --- | --- | --- |
| *Linear Distortion* | *s* |  |

|  |  |  |
| --- | --- | --- |
| *Non-Linear Distortion* | *ep* | *rd and tg* |

where:

*fx*: horizontal focal length  
*fy*: vertical focal length  
*s*: skew factor

*ep*: entrance pupil parameter

*rd*: radial distortion parameter

*tg*: tangential distortion parameter  
*cx*: principal point x  
*cy*: principal point y

NOTE 1 For most cameras, pixels are square and there is no skew. This corresponds to *s* being zero and *fx* being equal to *fy*.

NOTE2 Most cameras are highly distorted with radial, tangential and entrance pupil distortions.

The flags field is used to define the values of denominator and distortion denominator.

The variable *denominator* is set equal to (1 << *denominatorShiftOperand*) where *denominatorShiftOperand* is equal to ((flags & 0x001F00) >> 8).

The variable *distortionDenominator* is set equal to (1 << *distortionDenominatorShiftOperand)* where *distortionDenominatorShiftOperand* is equal to((flags & 0x1F0000) >> 16).

The values of the above intrinsics matrix can be calculated as follows:

*fx* = focal\_length\_x × image\_width / *denominator*

*fy* = focal\_length\_y × image\_height / *denominator*

*cx* = principal\_point\_x × image\_width / *denominator*

*cy* = principal\_point\_y × image\_height / *denominator*

*s* = skew\_factor / *distortionDenominator*

*ep[j]* = ep\_coeff[j]/ *distortionDenominator*

*rd[k]* = rd\_coeff[k]/ *distortionDenominator*

*tg[l]* = tg\_coeff[l]/ *distortionDenominator*

where

image\_width and image\_height come from the ImageSpatialExtentsProperty associated with the image item.

NOTE 3 By specifying the focal lengths and principal point as normalized by image dimensions, this allows for the intrinsics matrix to be scale invariant. For a camera system without skew, this means that the same intrinsics matrix can be used even if the sensor uses pixel binning to output images with varying number of pixels.

The entrance pupil distortion, radial distortion and tangential distortion parameters are formulated as below.

A camera model with the entrance pupil, is given in equation (6.5.40-1)

(6.5.40-1)

where

Rotation and Translation are extrinsic parameters that are used to transform world point P(X,Y,Z) to camera coordinate point (, , ).

is the incidence angle from the optical centre, of a camera lens. The incidence angle for any particular point is calculated from the camera coordinate point (, , ) as in equation (6.5.40-2)

(6.5.40-2)

The entrance pupil distortion is calculated as in equation 6.5.40-3.

+ ++ +… (6.5.40-3)

Equation 6.5.40-2 is further simplified with the introduction of normalized EP function which is expressed on the focal length as below:

(6.5.40-4)

(6.5.40-5)

(6.5.40-6)

where

is the normalized entrance pupil parameter, as represented in equation 6.5.40-7, below.

+ ++ + / ….. (6.5.40-7)

and is the pixel sizes in pixels on x, y-axes, ,respectively.

NOTE 4 The values of and may be available from the manufacturer of the camera device.

is a normalized coefficient, which is determined from radial distance, as shown in equation 6.5.40-8, where represents a normalized radial distance on x, y, respectively.

(6.5.40-8)

Additionally, the value of x can be calculated as below

, and . (6.5.40-9)

A point on the distorted image in the normalized coordinate is represented as (,). A corresponding point on the undistorted or distortion free image coordinate may be defined as (. The formulation for the radial and tangential distortion is similarly defined below:

+ ++ (6.5.40-10)

+ ++ ) (6.5.40-11)

NOTE 5 In practice the following coefficients are sufficient for radial, entrance pupil and tangential distortions, respectively. Other higher order coefficients may not be necessary.

* + - 1. **Syntax**

aligned(8) class CameraIntrinsicsMatrix  
extends ItemFullProperty('cmin', version = 0, flags) {

unsigned int(1) skew\_flag;

unsigned int(1) ep\_flag;

unsigned int(1) radial\_flag;

unsigned int(1) tangential\_flag;

bit(4) reserved;  
 signed int(32) focal\_length\_x;

signed int(32) focal\_length\_y;  
signed int(32) principal\_point\_x;  
signed int(32) principal\_point\_y;

if (skew\_flag){ // pin-hole camera model   
 signed int(32) skew\_factor;

}

if (ep\_flag){ // entrance pupil coefficient

unsigned int(8) num\_ep\_coeffs;  
 for (int j=0; j<num\_ep\_coeffs; j++)  
 signed int(32) ep\_coeff[j];

}

if (radial\_flag){ // radial distortion coefficient

unsigned int(8) num\_rd\_coeffs;  
 for (int k=0; k<num\_rd\_coeffs; k++)  
 signed int(32) rd\_coeff[k];

}

if (tangential\_flag){ // tangential distortion coefficient

unsigned int(8) num\_tg\_coeffs;  
 for (int l=0; l<num\_tg\_coeffs; l++)  
 signed int(32) tg\_coeff[l];

}  
}

*[Ed. Note: It was suggested at MPEG#139 to enclose the* *additional distortion models from m60319 under version > 0 of the CameraIntrinsicProperty.*]

* + - 1. **Semantics**

skew\_flag when set to 0 indicates that no skew factor is present. skew\_flag when set to 1 indicates that skew factor is present.

ep\_flag when set to 0 indicates that no entrance pupil distortion parameter is present. ep\_flag when set to 1 indicates that entrance pupil distortion parameter is present.

radial\_flag when set to 0 indicates that no radial distortion parameter is present. radial\_flag when set to 1 indicates that radial distortion parameter is present.

tangential\_flag when set to 0 indicates that no tangential distortion parameter is present. tangential\_flag when set to 1 indicates that tangential distortion parameter is present.

focal\_length\_x specifies the horizontal focal length of the camera in image widths.

focal\_length\_y specifies the vertical focal length of the camera in image heights. When not present, the value shall be implied to be focal\_length\_x \* image\_width / image\_height.

principal\_point\_x specifies the principal point x-coordinate in image widths.

principal\_point\_y specifies the principal point y-coordinate in image heights.

skew\_factor specifies the camera system skew factor. When not present its value shall be implied to be 0.

num\_ep\_coeffs specifies the number coefficients used for specifying the entrance-pupil distortion. The value of num\_ep\_coeffs is in the range of 0 to 3.

ep\_coeff[j] specifies the jth entrance pupil coefficient of the entrance pupil distortion.

num\_rd\_coeffs specifies the number coefficients used for specifying the radial distortion. The value of num\_rd\_coeffs is in the range of 0 to 3.

rd\_coeff[k] specifies the kth radial distortion coefficient of the radial distortion.

num\_tg\_coeffs specifies the number coefficients used for specifying the tangential distortion. The value of num\_tg\_coeffs is in the range of 0 to 1.

tg\_coeff[l] specifies the lth tangential distortion coefficient of the tangential distortion.

# Matrix-based transformation for image items

*[[ Ed. (FD): MPEG#129: it was questioned:”* Should we also add ‘matrix’ as an image derivation in the HEIF? “. It was warned that “We would need to be clear about the meaning of outputs that don’t have horizontal and vertical sides; if that’s overlaid, the meaning is clear, but what if it’s supposed to be displayed?”*]]*

# Signaling for pre-derived coded image items

*Replace the clause 6.4.7 with the following text:*

**6.4.7** **Pre-derived coded images**

[Ed. (FD): In the following, differences with HEIF 2nd edition (w18310) are highlighted in blue]

If a coded image has been derived from others — for example, a composite HDR image derived from exposure-bracketed individual images, or a panorama derived from a set of images — then it shall be linked to those images by item references of type 'base'. Item references may be from the coded image to all images it derives from, or when unique IDs are used, from the coded image to all entity groups or images it derives from. When unique IDs are used, a to\_item\_ID value in the SingleItemTypeReferenceBox or SingleItemTypeReferenceBoxLarge is resolved to an item identifier whenever the embedding MetaBox contains an item with such identifier, and is resolved to an entity group identifier otherwise.

An image item including a 'base' item reference is referred to as a pre-derived coded image.

NOTE In this version of this document, the exact derivation process used to produce the image is not described.

[[Ed. (FD): At MPEG#129, it was commented that “The slight snag here is defining what it means when the entity group does NOT imply a single output (e.g. a slide show); what does pre-derivation mean? ]]

*Add the following clause as section 6.4.7.1:*

**6.4.7.1 Signaling of the derivation method for pre-derived coded image items**

A pre-derived coded image shall be linked to images it derives from by an item reference of type 'base' to the entity group containing all images the pre-derived coded images derives from. The grouping\_type of the EntityToGroupBox specifies the purpose of grouping and implicitly signals the type of the derivation operation which was applied to generate the pre-derived coded image.

[[Ed. (FM): At MPEG#126, it was commented that “we somehow need to indicate the derivation operation, rather than the nature of the input set”]]

[[Ed. (FD): At MPEG#129, it was commented that “We could allow a pre-derivation of the implied derivation of that entity group.”]]