# Video Contour Map Payload Metadata within the QuickTime Movie File Format

## Format additions

Version 0.9 (Beta)

June 21, 2023

Note: The information contained within this document is preliminary and is subject to change.

**Introduction** ........................................ **3**
**Overview** ........................................ **3**
**References** ........................................ **4**
**Motivations and goals of video contour maps** ........................................ **4**
Characterizing spatial aspects of an associated video frame ........................................ 4
Separation of contour map from encoded video sample ........................................ 4
Integrating time ........................................ 5
Use with captions ........................................ 6
Client and production usage ........................................ 6
**Video contour collection and video contour map boxes** ........................................ **6**
1. Video Contour Collection (‘ctrs’) box ........................................ 7
1.1. Definition ........................................ 7
1.2. Syntax ........................................ 7
1.3. Semantics ........................................ 7
1.4. Video contour map ........................................ 8
1.5. Contour map geometry ........................................ 9
1.6. Contour map operator ........................................ 9
1.7. Contour map element values and element format types ........................................ 10
1.8. Contour map geometry-associated values ........................................ 10
3. Video Contour Map (‘ctrm’) box ........................................ 11
1.1. Definition ........................................ 11
1.2. Syntax ........................................ 11
1.3. Semantics ........................................ 12
**Timed Metadata, Captions and Stereoscopic Video** ........................................ **15**
Caption parallax timed metadata items ........................................ 15
Simplifying use of caption parallax timed metadata ........................................ 15
**Conclusion** ........................................ **16**
**Document Revision History** ........................................ **17**

# Introduction

This document specifies the structure of a metadata payload that can serve to describe parallax values associated with 2D areas of a corresponding stereoscopic video frame. The structure described is more general than parallax but parallax is the identified first use case.

This payload can be carried in metadata items in media samples within QuickTime File Format [QTFF] timed metadata or ISOBMFF [ISOBMFF] multiplexed metadata. Both of these use the `'mebx'` handler type of the `'meta'` track type. The payloads can also occur in fragmented movie files in both QTFF and ISOBMFF. While this specification is focused on use within a timed metadata track, the construct itself should be applicable elsewhere in the format.

The document begins with motivations and goals of the payload definition. This is followed by details of the format definition itself including use of this to describe payload design details and key related information for carrying parallax.

# Overview

Traditionally, captions are placed in the horizontal and vertical axes over video. With the introduction of stereoscopic video, however, there is a risk of depth collision if captions are placed in Z so they might intersect with the video’s stereoscopic elements or cues that have a parallax (i.e., horizontal disparity on the screen plane in Z) that is less than the screen plane. This “depth conflict” can produce viewer discomfort. To account for this, captions can have their parallax adjusted to have a more negative parallax than these potentially conflicting video elements so they do not collide.

Determining and then signaling this minimal parallax needs a mechanism. That mechanism might carry a single parallax value across the entire corresponding video frame. That mechanism might alternatively carry some kind of description of different areas of the video and the parallax for each area. A maximum parallax may be useful in addition to determine something termed the *depth budget*.

The mechanism used with parallax could be more general but still use the same geometries. It might differ in the type of values signaled for areas but nothing else.

Such information may be made more useful if it can be integrated over more than the current associated video frame.

This document describes an atom-based format that can be used in the timed metadata format of the QuickTime File Format [QTFF] and the ISO Base Media Format [ISOBMFF]. It introduces something we term a **video contour map** and a **video contour collection**.

Note: This document uses the ISOBMFF specification syntax [ISOBMFF] and uses the QTFF term “atom” interchangeably with the ISO term “box”. When describing syntax and constructs, “box” is used. While these constructs are likely most applicable to timed metadata items, they might prove useful in other parts of the file formats.

Note: The words "may", "should", and "shall" are used in the conventional specification sense, that is, respectively, to denote permitted, recommended, or required behaviors.

## References

This document references these external specifications:

[QTFF] QuickTime File Format specification

[ISOBMFF] ISO Base Media File Format or sometimes ISO BMFF as specified in ISO/IEC 14496-12 (2020)

## Motivations and goals of video contour maps

### Characterizing spatial aspects of an associated video frame

Characterizing features of areas of the video frame may prove useful or important for particular rendering or UI treatments. An example is characterizing local minimum parallax values (i.e., horizontal disparity in stereoscopic video) to be used to place captions or other elements in a way that won't produce depth collisions in viewing the stereoscopic video frame.

This document introduces a general notion termed here as a **video contour** which is a characterization of values across a two-dimensional video frame or image. This characterization can be encoded in a **video contour map** that defines a geometry that is mapped to element values. The geometries might include a row count by column count tiling of the two-dimensional area or an array of rectangles where the tiles or rectangles respectively map to a value. Another geometry, termed a "null" geometry here, is a single value true across the video frame and is expressible with a 1x1 tiling. Element values can be of a number of predefined **element format types** such as a parallax value (and analogous to a pixel format type such as ARGB). Video contour maps also have the notion of an **operator** which characterizes the processing that was used to derive element values such as a minimum or maximum of finer-grained source values that are summarized in the contour map.

Note: Something like minimum parallax is the result of combining a minimum operator with values having a parallax element format type.

Because multiple simultaneous characterizations for the video frame may be useful, more than one video contour map can be collected together into a **video contour collection**, a `Box` structure holding video contour map boxes. In this way, a tiled minimum parallax video contour map might be included with a maximum-parallax, null-geometry parallax contour map. This combination of minimum and maximum parallax allows something termed "depth budget" to be signaled. Another use of a collection might be minimum-parallax contour maps at different resolutions perhaps to accommodate clients with different rendering compute capabilities.

Note: As a shorthand, a video contour map carrying parallax element format values can be called a **parallax contour map**. A parallax video contour can be called a **parallax contour**.

### Separation of contour map from encoded video sample

The video contour map design described here will typically be carried in a timed metadata track (i.e., the '`mebx`' media subtype format described in QTFF and in ISOBMFF).

While an alternative might have been to include the video contour map semantics in a Supplemental Enhancement Information (SEI) message, the use of timed metadata and separation of the video contour maps from the encoded video frame offers benefits including:

- Multiple metadata tracks with different contour maps can be associated with the video track.
- New contour maps can be introduced without needing to rewrite the video.
- Because it is not in the encoded video, the video contour map can be delivered to the client without requiring delivery and decode (or even scanning) of the encoded video frame.
- Because it is not in the encoded video, different resolutions of video can be encoded and used with the same contour map metadata. This might be useful in streaming tiers differing by resolution.
- The same metadata sample and its video contour collection can be applied to more than one video sample if the video contour is the same or deemed sufficiently similar.

While video contour collections and contour maps are most likely to be used with timed metadata tracks, the design should be able to be used with static metadata as an item so long as no timing is signaled in the metadata item payload.

## Integrating time

Video contour map metadata items are carried as timed metadata track samples. Each metadata sample can correspond to one video sample's presentation time and duration or correspond to more than one video sample.

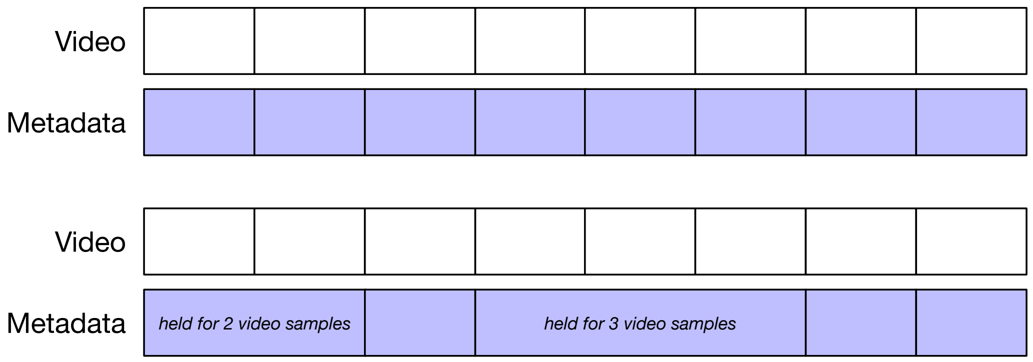

**Fig 1**. Video and associated timed metadata (1:1 and varied This is the "tiled" geometry)

While a metadata sample can span multiple video frames, it is recommended where possible to have one metadata sample with video contours per video frame. This simplifies HTTP Live Streaming (a.k.a., "HLS") segmentation. Also, if there is any change in video contour from frame to frame a new metadata sample is required at each change. Even if the same video contour is possible across a span of video frame times, it is still recommended that you write repeated metadata samples in a 1:1 relationship with video frame samples.

Timed video contour metadata can optionally consider a time horizon of frames beyond the current video-frame sample. A use might be for a forward time window that integrates the minimum parallax contours of all frames from the current frame forward for some duration. This can serve to smooth parallax and account for changes in upcoming frames that would otherwise produce a change in parallax every frame. There is no requirement to do so.

For scenes in an asset (or “shots”), the time integration might cover the contours of all video frames from the first to last frames in the scene. Note that caption activation and retirement timing is unlikely to align with scene boundaries. Scene duration integration may be more useful for video contour maps with other kinds of element format types (e.g., average or maximum luminance).

### Use with captions

The initial goal for the design of video contour maps is related to their use with caption media. Here, captions means both subtitles and closed captions and uses of them such as subtitles for deaf and hard of hearing.

Use of video contour maps with captions includes:

- Characterizing the most negative parallax so the renderer can place captions in Z over a stereoscopic view to avoid depth collisions
- Describing the minimum parallax for different areas of the frame so captions can be placed at an effective Z (by adjusting horizontal disparity) that is farther from the viewer than the overall minimum parallax (e.g., for an area where objects are farther back or absent)
- Allowing the captions used to be decoupled from the video so new tracks/streams of captions in varying languages and layouts can be used without needing to adapt each to the related video (i.e, the contour map characterizes the video and the captions adapt to the contour map)
- Adapting caption Z placement to Accessibility preferences changes in the default size (i.e., preferences are dynamic and cannot be predicted *a priori* when delivering content)
- Supporting independence of the caption format from its use over stereoscopic content (i.e., CEA-608 or WebVTT source are unchanged)
- Decoupling of the caption render time from the current video frame being rendered so captions do not have to be absolutely synchronized with video

### Client and production usage

The use of contour maps might be different in production and client delivery.

As an example, in a first pass, production might generate higher resolution parallax maps characterizing the corresponding video frames. The first pass might also generate several resolutions. Subsequent passes might produce contour maps that integrate longer time ranges of already determined contour maps might be produced. In this way, the contour map construct described here is both generated and used as source for generating more contour maps.

Client-delivered metadata might include a subset of what production generates. This might be done to reduce bandwidth or to tailor the resolution or other detail based upon client-compute capability.

## Video contour collection and video contour map boxes

This section describes the structure of the Video contour collection box and the constituent video contour map box.

Current box types in the box hierarchy:

| FourCC | FourCC | Box syntax element |
|---|---|---|
| ctrs | | VideoContourCollectionBox |
| | ctrm | VideoContourMapBox |
| | free | FreeSpaceBox |

## 1. Video Contour Collection ('ctrs') box

This section describes the Video Contour Collection box.

### 1.1. Definition

Box Type: 'ctrs'
Container: Timed metadata access unit metadata item Box
Mandatory: No
Quantity: Zero or one

A Video Contour Collection is a QuickTime File Format atom [QTFF] or Box in ISO/IEC 14496-12 [ISOBMFF] with the ISO Box boxtype of 'ctrs' (for "contours"). It can contain zero or more child Boxes that together signal one or more contours or shapes of the video frame associated with Video Contour Collection box. The typical usage of a Video Contour Collection will be to be carried in a metadata item of the timed metadata track that corresponds to time-aligned video samples. Having no child boxes is valid but likely not useful. Having only child FreeSpaceBoxes is appropriate if wanting to reserve space in the VideoContourCollectionBox that might be populated later in some production process.

A VideoContourCollectionBox will typically hold one or more VideoContourMapBoxes (see below). Readers should be prepared to find other Box types and either ignore them or process them if they understand them.

New Boxes should not be introduced into the VideoContourCollectionBox unless documented in this specification or a successor version of this specification. FreeSpaceBoxes can appear.

### 1.2. Syntax

```
aligned(8) class VideoContourCollectionBox extends Box('ctrs') {
   VideoContourMapBox()[];
   Box()[];             // other optional boxes with FreeSpaceBox()
                        //   reserved for its expected use
}
```

### 1.3. Semantics

VideoContourCollectionBox contains zero or more child boxes that signal a specific video contour of an associated video frame or an associated sequence of frames. Child boxes are defined in this specification now or may be added in the future. Child boxes might be defined in external specifications but the boxtype used there should be registered not to collide with boxes introduced in this or related specifications. The order of child boxes in the `VideoContourCollectionBox` and in all contained boxes recursively is not prescribed. A reader should be prepared to find boxes in any order.

**Note:** As `FreeSpaceBox` ('free') has a very common meaning in ISOBMFF and QTFF, one or more `FreeSpaceBoxes` may occur among the child boxes and should be interpreted as having no other meaning than taking up space. There is no guarantee that the payload of a FreeSpaceBox will contain exclusively zero (0) bytes but that is encouraged. It is recommended that any FreeSpaceBox be filled with bytes having the value zero (0).

**Note:** An empty `VideoContourCollectionBox` (i.e., no child boxes) is allowed but should generally not be used. It may however be useful to reserve space by including a `'ctrs'` in concert with a contained `FreeSpaceBox` ('`free`').

## 1.4. Video contour map

The `VideoContourMapBox` is a `FullBox` carrying a mapping of areas in a frame or image to values of a signaled uniform type. `VideoContourMapBox` uses a `boxtype` of '`ctrm`'.

Each video contour map signals these categories of information:

- The kind of geometry the mapping describes (e.g., a row by column tiling of values, a single value across the entire associated frame, or an array of rectangles)
- The type and size of value the mapping provides for each area (e.g., a parallax measurement in a specific form)
- An indication of how the values were determined for the mapped area (e.g., a minimum, a maximum)
- Any necessary parameters the geometry kind requires (e.g., row and column count for tiling)
- An optional value that can be used for an area's mapping to signal the value is unknown or indeterminant
- The values associated with the areas that the geometry indicates

The purpose of a video contour map is to characterize the video according to some condition or predicate. One case is a mapping of the minimum parallax (a.k.a., horizontal disparity) across all measured parallax values for each area. Another case might use parallax values but instead be the maximum parallax. In these two examples, the value type is parallax but the "operator" differs (i.e., minimum vs. maximum).

Video contour maps for the same "operator" and value type can also differ in geometry or detail level of the same kind of geometry. For the tiled geometry kind, the `VideoContourMapCollectionBox` might contain one `VideoContourMapBox` with 3 x 3 tiles and another much more detailed with 40 x 30 tiles.

**Note:** A video contour box should not be used where coded video of a depth or disparity map is more appropriate. Coded video benefits from compression technology including differencing of frames. Video contour maps are each held in a sync frame.

## 1.5.Contour map geometry

Two kinds of geometries are currently defined. The following describes each and the number of values that each implies:

1. A *tiled geometry* where the entirety of the dimensions of the video frame are tiled by N rows of M columns and a value is associated with each tile. There are N x M values.
2. A *rectangle array geometry* where an array of rectangle descriptions cover the video frame dimensions either partially or wholly. Each rectangle has one associated value. If *count* is the number of rectangles in this array, there are *count* values.

New kinds of geometries may be introduced in future versions of this specification. Readers should confirm they understand the signaled geometry and ignore the contour map if they do not understand it.

## 1.6.Contour map operator

The *operator* indicates how the values were derived. As the initial focus is summarizing values across areas the geometry specified, these will tend to be operations that summarize. Two kinds of operators are currently defined:

- `Minimum`: An area's value is the minimum of the values measured across the source area the corresponding geometry specifies.
- `Maximum`: An area's value is the maximum of the values measured across the source area the corresponding geometry specifies.

New kinds of operators may be introduced in future versions of this specification. Readers should confirm they understand the signaled operator and ignore the contour map if they do not understand it.

The operators are meant to be agnostic to the specific kind of value the contour map carries. So, if there was also an element format type for luminance, a minimum operator could also be applied to determine the minimum luminance.

While it is possible to introduce new operators that are specific to the value type, it is useful to use the same operator across disparate value types.

Other kinds of operators might include the mean (average) or the median value. These are not currently defined.

## 1.7. Contour map element values and element format types

A video contour map carries a single type of value. The type of value has a particular semantic and encoding. The semantic defines how the value should be interpreted. The encoding is the machine representation and includes its size.

The element format type documented here is a parallax value that is represented on the signed integer range -100000 to +100000 inclusive (negative one hundred thousand to positive one hundred thousand) that can be interpreted either as from -1.0 to +1.0 inclusive or as -100.0% to +100.0% inclusive. The element format type embodies a few things:

- that the format is related to parallax as opposed to say luminance
- that the format value is a signed integer as opposed to a floating point number
- that the format value defines a particular interpretation

The encoding however is the detail of how that semantic is represented in the file format. This parallax value might be stored as a signed, big endian 32-bit integer.

As the initial focus is summarizing minimum and maximum parallax across areas the geometry specifies, only one kind of value is currently defined:

- `parallax_value_type`: A signed, 32-bit integer defining a parallax (horizontal disparity) measurement expressed as the range -100000 to +100000. This can be interpreted as mapping onto the range -1.0 to +1.0 or -100.0% to +100.0%. As a 32-bit integer can carry values outside this (-100000,+100000) range, a contour map can use excursions beyond this range to signal a special undefined value. Analogous to a floating-point NaN, the special integer serves to indicate that no parallax value was produced and allowing the reader to ignore that part of the geometry.

New kinds of contour map element format types may be introduced in future versions of this specification. Readers should confirm they understand the format type and ignore the contour map if they do not understand it.

## 1.8. Contour map geometry-associated values

A contour map carries one or more values of the signaled element format type. The number of values depends upon the kind of geometry and the parameters, if any, for that geometry within the contour map.

In general, all values carried in the map are valid or known. As there are cases where a value cannot be measured or calculated reliably, video contour maps can signal a unique value that can be interpreted as unknown. This is analogous to a floating point NaN (i.e., not a number).

This contour map value is termed the "unknown value". It has no other interpretation than signaling the value in that part of the geometry (e.g., a particular area) could not be determined. An unknown value should be specially handled and should not participate in operations such as summing or minimum or mean calculations across area values.

Because handling contour maps with unknown values may be more computationally burdensome than contour maps without unknown values, the contour map signals (1) if it has an unknown value and (2) the value used. It is hoped that this allows the reader to detect the difference and perhaps apply different optimizations between contour maps with and without unknown values.

## 3. Video Contour Map (‘ctrm’) box

This section describes the Video Contour Map box structure or `VideoContourMapBox`.

### 1.1. Definition

Box Type: `‘ctrm’`
Container: Video Contour Collection Box (`‘ctrs’`)
Mandatory: Yes
Quantity: Zero or one

### 1.2. Syntax

```
aligned(8) class VideoContourMapBox extends FullBox(‘ctrm’, 0, 0) {
   // inline flags separate from FullBox ‘flags’ specific to this contour
map
   unsigned int(8)  operator;     // processing done to produce map (e.g.,
minimum, maximum) or 0 if entire map should be ignored
   unsigned int(5)  reserved = 0;
   unsigned int(1)  integrates_over_extended_time_window;
   unsigned int(1)  uses_forward_time_window;
   unsigned int(1)  uses_unknown_element_values;
   unsigned int(8)  geometry_type; // 1 (tiles), 2 (rect array)
   unsigned int(8)  element_field_size;  // size of an element in bits (8,
16, 32)
   unsigned int(32) element_format_type; // interpretation of the value and
analogous to a pixel format (e.g., a parallax measurement)

   switch( geometry_type ) {
      case 1:     // tiled geometry type
         unsigned int(16) row_count;
         unsigned int(16) column_count;
         break;
      case 2:     // rect_array geometry type
         unsigned int(16) rect_array_count;

         int i;
         for (i=0; i < rect_array_count; i++) {
            unsigned int(32) x;     // top left x, normalized range [0-1.0]
            unsigned int(32) y;     // top left y, normalized range [0-1.0]

            unsigned int(32) width; // width of the rectangle
            unsigned int(32) height;// height of the rectangle
         }
         break;
   }

   // duration of forward time window over which values are integrated
   if ( uses_forward_time_window ) {
      int(32) forward_time_window_value;
      int(32) forward_time_window_timescale;
   }

   // optional special element value
   if ( uses_unknown_element_values ) {
      unsigned int(element_field_size) unknown_element_value;
   }

   // element values array
   int element_count;
   switch( geometry_type ) {
      case 1:     // tiled geometry type
         element_count = row_count * column_count; // one value for each
tile
         break;
      case 2:     // rect_array geometry type
         element_count = rect_array_count;     // one value per rectangle
         break;
   }
   unsigned int(element_field_size) element_values[ element_count ];
                     // other optional boxes
}
```

## 1.3. Semantics

The version of this structure is 0. Any reader seeing a later version should either understand that future version according to a future version of this specification or ignore the `VideoContourMapBox`. A `VideoContourCollectionBox` may contain a mix of `VideoContourMapBoxes` with differing versions.

- `operator` is an enumerated value that indicates how element values were produced.

- `has_unknown_element_values` is an indication that the map values have a distinguished value that interpreted as an unknown or undefined value. This is analogous to a floating point `NaN`.

- `integrates_over_extended_time_window` is an indication that the element values are calculated based upon frames beyond the current frame.

- `uses_forward_time_window` is an indication that the field `forward_time_window` is present. If set, `integrates_over_extended_time_window` should also be set.

- `geometry_type` indicates the kind of geometry expressed in this contour map. There are three currently allowed values for `geometry_type`:
1 (or tiled geometry) : indicates a two-dimensional tiling across the entire extent of the associated video frame or image
2 (or rect array geometry) : indicates there is an array of rectangles that cover some or all of the associated video frame or image

**Note:** A 1 x 1 tiled geometry can be used for a single global value across the whole associated video frame.

- `element_field_size` is the number of bits needed to hold a single element value (one of 8, 16, 32).

- `element_format_type` indicates the type of value and its interpretation for each element held in `element_values` array and the optional `unknown_element_value`.

- `row_count` is the number of rows in the `element_values` array when the map uses the tiled geometry type.

- `column_count` is the number of columns in the `element_values` array when the map uses the tiled geometry type.

- `rect_array_count` is the number of entries in the rectangle array when the map uses the rectangle array geometry type.

- `x, y` are the left and top coordinates respectively of each element of the rectangle array when the map uses the rectangle array geometry type.  This is an unsigned 2.30 fixed point number and is normalized to a range of 0.0 to 1.0 mapped onto the overall dimension of the image it covers. The value 0x40000000 corresponds to 1.0.

- `width, height` are the width and height respectively of each element of the rectangle array when the map uses the rectangle array geometry type. This is an unsigned 2.30 fixed point number and is normalized to a range of 0.0 to 1.0 that is mapped onto the overall dimension of the image it covers.

- `unknown_element_value` is an optional value that can be used in the `element_values` array to indicate the element's value is unknown or undefined. This field is only present if the field `uses_unknown_element_values` is set to 1 in the contour map.

- `element_values` is an array of one or more element values associated with the geometry described by the map. The number of elements of `element_values` is determined by the kind of geometry indicated by `geometry_type` and the parameters associated with that geometry. If `has_unknown_element_values` is set, values of elements may have the value from `unknown_element_value`. If `has_unknown_element_values` is not set, values of elements should not have any unknown values. Nevertheless, it is okay to signal an unknown element value and not include it in `element_values`.

The `operator` field indicates the operator used to produce the video contour map.  Currently defined operator values are:

- 0 : reserved, can be used to mark the contour map invalid or unused as a whole
- 1 : a *minimum* or *min()* operation was applied to produce all element values, taking into consideration any potentially signaled unknown element values
- 2 : a *maximum* or *max()* operation was applied to produce all element values, taking into consideration any potentially signaled unknown element values

All other `operator` values are reserved.

The `geometry_type` field defines the kind of geometry the contour map uses to map areas to particular element values.  Currently defined geometry types are:

- 0 : reserved
- 1 : This is the “tiled” geometry and specifies a two-dimensional tiling of N rows x M columns is applied across the extent of the associated video frame or image. The order of values is in row major order so the values correspond to values for row 0 with columns 0 to M-1 followed by row 1’s columns 0 to M-1 followed by row 2 and so on up to row N-1. This can be thought of as

"raster order". A 1 row by 1 column can be used for a single value across the extent of the associated video frame or image.

- 2 : This is the "rectangle array" or "rect array" geometry and specifies an array of one or more rectangles where each covers some or all of the associated video frame or image. The order of values corresponds to the order of the rectangles.

  **Note:** The behavior of processing of rectangle arrays with spatially overlapping rectangles is undefined so overlapping is discouraged.

All other `geometry_type` values are reserved.

The `element_format_type` field defines the type of value the contour map stores in its `element_values` array. This is a 32-bit integer holding a FourCC. There is one currently defined element type:

- '`prlx`' : This indicates the value is a parallax value. Parallax—or horizontal disparity—using the '`prlx`' element format type describes a normalized value allowing both negative and positive parallax. A value of zero should be interpreted as at the screen plane. Negative values are nearer to the viewer than the screen plane. Positive values are farther from the viewer and beyond the screen plane. The value is a signed 32-bit integer that is interpreted as a uniform number over the range [-1.0…0.0…+1.0]. The valid range of the integer is from -100000 to +100000 which maps from -1.0 to +1.0. The interval of 0.0 to 1.0 and 0.0 to -1.0 are each mapped over the width of a view's image. Half of this value is applied to each stereo eye view.

All other `element_format_type` values are reserved.

More about the parallax ('prlx') element value type:

The encoded value is expressed as a percentage of the width of a video view (note: stereo left and stereo right views have the same resolution). An example might be 2.5%. Half is applied to each stereo eye but in opposite directions.

The value is encoded using a signed big endian 32-bit integer across a range of -100,000 to +100,000 which in turn maps from -1.0 to 1.0. The value 2.5% is 0.025, which when mapped onto the integer range, is 2500.

As the parallax value is signed, adding a negative value to the left stereo eye is equivalent to subtracting the absolute value of the value. The right stereo eye's subtraction of a negative value is likewise equivalent to adding the absolute value of the value. Another interpretation of the sign of the value is that positive denotes increased disparity with respect to the parallel view direction (e.g., horizontal) and negative denotes increased negative disparity with respect to the parallel view direction. Negative parallax is toward the viewer.

The minimum parallax corresponds to the "near" or "nearest" parallax. The maximum parallax corresponds to the "far" or "farthest" parallax. Having contour maps with both minimum and maximum will accommodate signaling of a depth budget for the corresponding frame(s).

The order of values in `element_values` array contents depend upon `geometry_type`.

# Timed Metadata, Captions and Stereoscopic Video

Spatial media tracks such as video may benefit from having associated timed metadata. This specification makes use of a parallel metadata track. This timed metadata track can use the ISOBMFF '`mebx`' format's ability to carry a number of metadata items for a time range. Media samples can carry a mix of items of different metadata item keys allowing a flexible way to signal a variety of structural or descriptive information.

We consider here one kind of timed metadata payload related to describing the parallax of decoded stereoscopic video frames. While general, its first expected use case is for caption placement in Z.

## Caption parallax timed metadata items

When carried in '`mebx`' timed metadata [QTFF, ISOBMFF], parallax metadata is carried as a `VideoContourCollectionBox` using the following key and value definition:

| Keyspace | Key | Datatype[1] | Well known data type[2] |
|---|---|---|---|
| mdta | com.apple.quicktime.video.parallax-coverage.measured | `VideoContourCollectionBox` | 0 |

**Note[1]:** BE is "Big Endian"
**Note[2]**: ISOBMFF does not currently support signaling datatypes as QTFF does.

**Value datatype**: A `VideoContourCollectionBox` with one or more child `VideoContourMapBoxes` using the parallax element format type (i.e., '`prlx`').

**Note:** Initially, the first contained `VideoContourMapBox` within the `VideoContourCollectionBox` should be one describing minimum parallax. Support for more than one `VideoContourMapBox` will likely be limited.

## Simplifying use of caption parallax timed metadata

There is a good amount of richness described in this specification. To maximize successful production of and client processing of contour map metadata a number of restrictions is described next.

1. Use a timed metadata track with a VideoContourCollectionBox payload item for each video frame. This is in preference to metadata items with durations longer than a video frame.
2. Use only one VideoContourMapBox in each VideoContourCollectionBox.
3. In the VideoContourMapBox,
    a. Use only the parallax `element_format_type` (value = '`prlx`')
    b. Use only the minimum `operator` (value = 1)
    c. Use only the tiled `geometry_type` (value = 1). A 1 x 1 tiling can be used for a single value characterizing the entirety of the associated video frame or image.
    d. Do not use `uses_unknown_element_values` or an associated value for `unknown_element_value`.

4. Provide valid parallax contours for each video frame.
5. Associate the timed metadata track with the video track it describes using a ‘cdsc’ track reference from the metadata track to the video track.

If any of the above restrictions are not satisfied, a reader can treat the contour as invalid and ignore its presence.

## Conclusion

The video contour collection defined in the `VideoContourCollectionBox` is an extensible construct that can be used to signal some number of contours describing something about the associated video frame(s) or image(s). The current `VideoContourMapBox` definition may be extended in future versions of this specification. Other kinds of `VideoContourCollectionBox` child boxes may also be introduced in the future. While initially intended to be carried as a metadata item with ‘`mebx`’ timed metadata track, the construct may find its way into other parts of the QuickTime File Format. The design should be applicable to ISOBMFF derived formats if that proves useful but there is no intention to replace any existing ISO signaling with this design.

## Document Revision History

| Date | Revision | Notes |
|---|---|---|
| 2023-06-21 | 0.9 | First version |