|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11**  28th Meeting: Torino, IT, 15–21 July 2017 | Document: JCTVC-AB0038 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **Omnidirectional fisheye video SEI message** | | |
| *Status:* | Input document to JCT-VC | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Hyun-Mook Oh Sejin Oh | Email: | [hyunmook.oh@lge.com](mailto:hyunmook.oh@lge.com) |
| *Source:* | LG Electronics | | |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Abstract

This contribution proposes to define omnidirectional fisheye video SEI message to carry fish eye video parameters that can be used in the post decoding process. The aim of the SEI message is to provide information that enables to construct a spherical video in the receiver. Based on the description of fish eye video in OMAF DIS text, three categories of information are proposed in the SEI message: circular image description, spherical domain mapping parameter, and fisheye camera parameters.

# Problem Statement

In OMAF DIS text [1], two types of video formats are considered as input to video encoding: projected picture and fisheye circular picture. The fisheye circular picture conveys direct camera outputs in a rectangular format while the projected picture is generated by arranging an image on a sphere onto a two-dimensional picture. Comparing to the conventional omnidirectional video generation process, two main process steps are assumed to be omitted for fisheye video: stitching process which maps the camera outputs into a spherical surface and the projection process which generates a projected rectangular picture from the mapped image on the spherical surface. Due to the omission of these processes, the pre-encoding burden is reduced while the corresponding work is necessary in the receiver.

When the projected picture is delivered, the picture is decoded and projected onto the screen of head-mounted display or any other display device according to the user viewport. Unlike this, when the fisheye circular picture is delivered, stitching of circular images and projecting it onto the sphere needs to be performed after video decoding. To generate an omnidirectional video from the raw circular data, information about the camera outputs and the camera parameters, which affect to the image quality of stitching needs to be signaled. The following fisheye parameters are the necessary for receivers to generate high quality of the omnidirectional video from the fisheye circular picture, as specified in the section 6 of OMAF DIS text [1].

* Indication that the decoded picture is fisheye video, and the post-processing is needed to construct the omnidirectional video.
* Information on circular images conveyed in the decoded picture: the number of circular images, location, size, and transformation, etc.
* Spherical domain mapping information: spherical regions corresponding to the circular images (a point that the center pixel of the circular image and corresponding field of views.
* Fisheye camera parameters: the position of the fisheye camera center, lens characteristics, etc.

# Proposal

In this contribution, we propose to define an SEI message to carry on information to indicate that the decoded picture is fisheye video and to inform receiver that the post-processing is needed to construct the spherical video. According to the number of circular image produced from fisheye cameras, the detailed information with regard to decoded picture, the projection on the spherical surface, and fisheye camera parameters could be delivered as defined in OMAF DIS text. An example of the detailed syntax elements of those parameters is provided in Appendix.

## Syntax

|  |  |
| --- | --- |
| omnidirectional\_fisheye\_video( payloadSize ) { | **Descriptor** |
| fisheye\_video ( ) |  |
| } |  |

## Semantics

The presence of the omnidirectional fisheye video SEI message in a CLVS indicates that each coded video picture in the CLVS is an omnidirectional fisheye video picture containing a number of (usually two) circular images captured by fisheye camera lens. The information of the omnidirectional fisheye video carried in the omnidirectional fisheye video SEI message can be used by a receiver to properly render the omnidirectional fisheye video.

The omnidirectional fisheye video SEI message applies to the CLVS that contains the SEI message, also referred to as the current CLVS. When present in a CVLS the omnidirectional fisheye video SEI message shall be present in the first access unit of the CLVS and may be present in other access units of the CLVS.

fisheye\_video ( ) contains the elements of information on circular images conveyed in the decoded picture, spherical domain mapping information, and fisheye camera parameters for omnidirectional video application using fisheye video.

1. **References**
2. W16824, “Text of ISO/IEC DIS 23090-2 Omnidirectional Media Format”, B. Choi, Y.-K. Wang, M. M. Hannuksela, Y. Lim, April 2017, Hobart.

# Patent rights declaration(s)

**LG Electronics may have current or pending patent rights relating to the technology described in this contribution and, conditioned on reciprocity, is prepared to grant licenses under reasonable and non-discriminatory terms as necessary for implementation of the resulting ITU-T Recommendation | ISO/IEC International Standard (per box 2 of the ITU-T/ITU-R/ISO/IEC patent statement and licensing declaration form).**

## Appendix Example of fisheye video element syntax

|  |  |
| --- | --- |
| fisheye\_video () { | **Descriptor** |
| **num\_circular\_images** | u(8) |
| for(i=0; i< num\_circular\_images; i++) { |  |
| **image\_center\_x**[ i ] | u(32) |
| **image\_center\_y**[ i ] | u(32) |
| **full\_radius**[ i ] | u(32) |
| **picture\_radius**[ i ] | u(32) |
| **scene\_radius**[ i ] | u(32) |
| **image\_rotation**[ i ] | u(32) |
| **image\_flip**[ i ] | u(2) |
| **image\_scale\_axis\_angle**[ i ] | u(32) |
| **image\_scale\_x**[ i ] | u(32) |
| **image\_scale\_y**[ i ] | u(32) |
| **field\_of\_view**[ i ] | u(32) |
| **num\_angle\_for\_displaying\_fov**[ i ] | u(16) |
| for(j=0; j<num\_angle\_for\_displaying\_fov[ i ]; j++) { |  |
| **displayed\_fov**[ i ][ j ] | u(32) |
| **overlapped\_fov**[ i ][ j ] | u(32) |
| } |  |
| **camera\_center\_yaw**[ i ] | i(32) |
| **camera\_center\_pitch**[ i ] | i(32) |
| **camera\_center\_roll**[ i ] | i(32) |
| **camera\_center\_offset\_x**[ i ] | u(32) |
| **camera\_center\_offset\_y**[ i ] | u(32) |
| **camera\_center\_offset\_z**[ i ] | u(32) |
| **num\_polynomial\_coefficeients** [ i ] | u(16) |
| for(j=0; j< num\_polynomial\_coefficeients[ i ]; j++) |  |
| **polynomial\_coefficient\_K**[ i ][ j ] | u(32) |
| **num\_local\_fov\_region**[ i ] | u(16) |
| for(j=0; j<num\_local\_fov\_region[ i ]; j++) { |  |
| **start\_radius**[ i ][ j ] | u(32) |
| **end\_radius**[ i ][ j ] | u(32) |
| **start\_angle**[ i ][ j ] | i(32) |
| **end\_angle**[ i ][ j ] | i(32) |
| **radius\_delta**[ i ][ j ] | u(32) |
| **angle\_delta**[ i ][ j ] | i(32) |
| for(k=0; k<= numRadius; k++) |  |
| for(l=0; l<= numAngle; l++) |  |
| **local\_fov\_weight**[ i ][ j ][ k ][ l ] | u(32) |
| } |  |
| **num\_polynomial\_coefficients\_lsc**[ i ] | u(16) |
| for(j=0; j< num\_polynomial\_coefficients\_lsc[ i ]; j++) { |  |
| **polynomial\_coefficient\_K\_lsc\_R**[ i ][ j ] | u(32) |
| **polynomial\_coefficient\_K\_lsc\_G[** i ][ j ] | u(32) |
| **polynomial\_coefficient\_K\_lsc\_B[** i ][ j ] | u(32) |
| } |  |
| } |  |
| **num\_deadzones** | u(8) |
| for(i=0; i< num\_deadzones; i++) { |  |
| **deadzone\_left\_horizontal\_offset**[ i ] | u(16) |
| **deadzone\_top\_vertical\_offset**[ i ] | u(16) |
| **deadzone\_width**[ i ] | u(16) |
| **deadzone\_height**[ i ] | u(16) |
| } |  |
| } |  |

The semantics of num\_circular\_images\_minus1, image\_center\_x[ i ], image\_center\_y[ i ], full\_radius[ i ], picture\_radius[ i ], scene\_radius[ i ], image\_rotation[ i ], image\_flip[ i ], image\_scale\_axis\_angle[ i ], image\_scale\_x[ i ], image\_scale\_y[ i ], field\_of\_view[ i ], num\_angle\_for\_displaying\_fov[ i ], displayed\_fov[ i ][ j ], overlapped\_fov[ i ][ j ], camera\_center\_yaw[ i ], camera\_center\_pitch[ i ], camera\_center\_roll[ i ], camera\_center\_offset\_x[ i ], camera\_center\_offset\_y[ i ], camera\_center\_offset\_z[ i ], num\_polynomial\_coefficeients [ i ], polynomial\_coefficient\_K[ i ][ j ], num\_local\_fov\_region[ i ], start\_radius[ i ][ j ], end\_radius[ i ][ j ], start\_angle[ i ][ j ], end\_angle[ i ][ j ], radius\_delta[ i ][ j ], angle\_delta[ i ][ j ], local\_fov\_weight[ i ][ j ][ k ][ l ], num\_polynomial\_coefficients\_lsc[ i ],polynomial\_coefficient\_K\_lsc\_R[ i ][ j ], polynomial\_coefficient\_K\_lsc\_G[ i ][ j ], polynomial\_coefficient\_K\_lsc\_B[ i ][ j ], num\_deadzones, deadzone\_left\_horizontal\_offset[ i ], deadzone\_top\_vertical\_offset[ i ], deadzone\_width[ i ], deadzone\_height[ i ] follows the semantics defined in OMAF DIS.

numRadius and numAngle are defined as follows.

numRadius = (end\_radius[i][j]-start\_radius[i][j])/radius\_delta[i][j]

numAngle = (end\_angle[i][j]-start\_angle[i][j])/angle\_delta[i][j]