|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11**  37th Meeting: Geneva, CH, 4–10 October 2019 | Document: JCTVC-AK0031 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **SW implementation of fisheye projection format** | | |
| *Status:* | Input document to JCT-VC | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Hyun-Mook Oh | Email: | [hyunmook.oh@lge.com](mailto:hyunmook.oh@lge.com) |
| *Source:* | LG Electronics, Inc. | | |

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

# Abstract

In this contribution, SW implementation of fisheye projection format is described to study 2D to 3D conversion equations defined for fisheye video information SEI message. The mapping function is implemented on 360Lib v9.1 which supports conversion with and without encoder as well as E2E PSNR calculation. To verify the implementation, a picture of the ERP format is converted to the fisheye projection format and then converted back to the ERP format. Based on the objective and subjective results of the conversion, we propose to include the fisheye video information SEI message with the mapping equation in the amendment to HEVC specification.

# SW implementation of fisheye projection format

Fisheye video information SEI message was adopted to the amendment of HEVC [1] where the mapping equation was asked to be studied by software implementation. To use the fisheye projection format in 360Lib v9.1, a new projection format is defined and conversion between 2D and 3D coordinate is implemented. The implementation is examined by converting fisheye projection format to ERP where the input fisheye picture is obtained from the ERP format. In the following, the source code and the implementation result are discussed.

|  |  |  |
| --- | --- | --- |
| **Projection** | **Frame packing** | **Example Image** |
| Fisheye |  | **cid:image006.png@01D564CD.7A14BB70** |

## Fisheye to ERP conversion on 360Lib v9.1

A point on a circular image of the fisheye circular projection could be matched with a point on a unit sphere where the coverage on a sphere is determined by the field of view (FoV) of the fisheye camera lens, e.g., the coverage on a sphere for 180 degrees of FoV is hemi-sphere. In Figure 1, the relationship between a point on a unit fisheye circular projection image, a corresponding point on a unit sphere, and a point on ERP is described. In the conversion from fisheye circular projection to the sphere coordinate, the circular image is assumed to be on the YZ plane where the angle φ from the plane is estimated as a linear function of the radius. After the fisheye circular projection format to sphere coordinate conversion, the point could be mapped to ERP format by calculating longitude and latitude of the point.

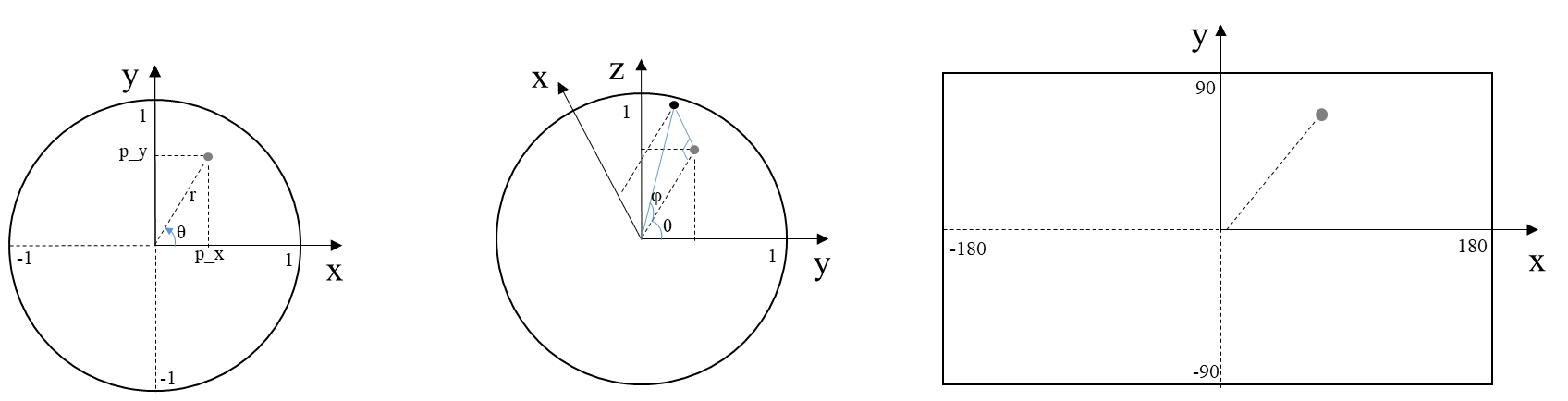


Figure 1. Correspondence of a point on (a) fisheye circular projection, (b) sphere coordinate and (c) ERP

In the implementation, fisheye projection to sphere coordinate, and its inverse was implemented based on the relationship described above. The source code of 2D to 3D conversion is shown below.

*Void TFisheye::map2DTo3D(SPos& IPosIn, SPos \*pSPosOut)*

*{*

*POSType cx, cy;*

*cx = (IPosIn.x - m\_FisheyeInfo.fCircularRegionCentre\_x)*

*/ (m\_FisheyeInfo.fCircularRegionRadius);*

*cy = (m\_FisheyeInfo.fCircularRegionCentre\_y - IPosIn.y)*

*/ (m\_FisheyeInfo.fCircularRegionRadius);*

*Double FOVrad = S\_PI \* m\_FisheyeInfo.fFOV / 180.0; // FOV of the fisheye*

*Double r\_dist = ssqrt(cx\*cx + cy\*cy);*

*Double phi = r\_dist \* FOVrad / 2;*

*Double theta = atan2(cy, cx);*

*pSPosOut->x = (POSType)(scos(phi));*

*pSPosOut->y = (POSType)(ssin(phi)\*ssin(theta));*

*pSPosOut->z = -(POSType)(ssin(phi)\*scos(theta));*

*rotate3D(\*pSPosOut, (Int)round(m\_FisheyeInfo.fCentreTilt),*

*(Int)round(m\_FisheyeInfo.fCentreElevation),*

*(Int)round(m\_FisheyeInfo.fCentreAzimuth));*

*}*

## Experimental results

To verify the implementation, a test sequence in ERP format is converted to fisheye projection format where the parameters are set as show below. Note that, the same parameters are used for both forward and backward conversion, i.e., from ERP to Fisheye and from Fisheye to ERP, respectively, to show that the pixel values are restored in the same position.

FisheyeCircularRegionCentreX : 511.5  
FisheyeCircularRegionCentreY : 511.5  
FisheyeCircularRegionRadius : 512  
FisheyeFieldOfView : 210  
FisheyeCenterAzimuth : 0  
FisheyeCenterElevation : 0  
FisheyeCenterTilt : 0  
FisheyeRectRegionTop : 0  
FisheyeRectRegionLeft : 0  
FisheyeRectRegionWidth : 1024  
FisheyeRectRegionHeight : 1024

In Figure 2, the subjective result of the conversion is described, where the gray represents the empty area. In Figure 2 (b), the result of the forward conversion is shown where the information from -105 degrees to 105 degrees in both longitude and latitude are contained in circular region. When the fisheye format is converted back to the ERP format, the pixel values in the active area of the fisheye format are mapped onto the ERP format where the pixels not covered by the fisheye format are filled with gray. For the objective verification, WS-PSNR value for hemi-sphere region, i.e., -180 to 180 for both of the longitude and the latitude, are calculated where the results are 35.4721, 40.0865, 42.4418 for each YUV channels.



1. Input video (PoleVault\_le\_3840\_1920\_30fps\_8bit\_420\_erp.yuv)



1. ERP to fisheye conversion (FoV=210, rotation = (0,0,0))



1. Reconstructed ERP from fisheye format

Figure 2. Result of conversion between fisheye circular projection and ERP

In Figure 3, the results of different parameters in terms of FoV are described.

Figure 3. Result of conversion between fisheye circular projection and ERP with different FoV. The first row shows the fisheye projection and corresponding ERP for FoV = 180, the bottom row shows that for FoV = 150.

# Proposal

Based on the study of the conversion equation with the software implementation, we propose to add the following to the amendment of HEVC [1].

*Add clause D.3.41.1.7, as follows:*

***D.3.41.1.7 Conversion from a sample location of an active area to sphere coordinates relative to the global coordinate axes***

Inputs to this process are:

– the sample location (x, y) in units of luma samples,

– the centre location (xc, yc) and the radius (rc) of the circular region that contains the i-th active area, given by fisheye\_circular\_region\_centre\_x[ i ], fisheye\_circular\_region\_centre\_y[ i ], and fisheye\_circular\_region\_radius[ i ], respectively, all in units of 2−16 luma samples,

– the field of view (θv) of the lens corresponding to the i-th active area, given by fisheye\_field\_of\_view[ i ], in units of 2−16 degrees,

– the rotation parameters (αc, βc, γc), given by fisheye\_camera\_centre\_azimuth[ i ], fisheye\_camera\_centre\_elevation[ i ], and fisheye\_camera\_centre\_tilt[ i ], respectively, all in units of 2−16 degrees, and

– the number of polynomial coefficients numCoeffs and the polynomial coefficients coeffVal[ j ] (for j ranging from 0 to numCoeffs − 1, inclusive) of the i-th active area, given by fisheye\_num\_polynomial\_coeffs[ i ] and fisheye\_polynomial\_coeff[ i ][ j ] (for j ranging from 0 to fisheye\_num\_polynomial\_coeffs[ i ] − 1, inclusive), respectively.

Outputs of this process are:

– sphere coordinates (ϕ, θ) relative to the global coordinate axes.

The method of converting a sample location of an active area to sphere coordinates is determined as follows:

– If numCoeffs is equal to 0, there is only one method of converting a sample location of an active area to sphere coordinates that is specified, which is to not use polynomial coefficients.

– Otherwise (numCoeffs is not equal to 0), there are two methods of converting a sample location of an active area to sphere coordinates that are specified, which are to not use polynomial coefficients or to use polynomial coefficients. The method using polynomial coefficients is preferred, as this method is intended to provide a more precise model of the fisheye characteristics. However, the other method may also be appropriate for some uses, as it provides a single conversion process that can be used regardless of whether numCoeffs is equal to 0 or not. This Specification does not prescribe which of the two methods is to be used in this case.

The outputs are derived as follows:

– If polynomial coefficients are not used, the angle ϕ′ is derived by

ϕ′ = ( Sqrt( ( x − xc ÷ 216 )2 + ( y − yc ÷ 216 )2 ) ÷ ( rc ÷ 216 ) ) \* ( θv ÷ 216 \* π ÷ 180 ) ÷ 2 (D‑XX)

– Otherwise (polynomial coefficients are used), the angle ϕ′ is derived by

ϕ′ = ( ( coeffVal[ j ] \* 2−24 ) \* ( Sqrt( ( x – xc \* 2−16 )2 + ( y – yc \* 2−16 )2 ) ÷ ( rc \* 2−16 ) )j )  
 (D‑XX)

The outputs are then derived as follows:

θ′ = Atan2( y − yc ÷ 216, x − xc ÷ 216 )  
x1 = Cos( ϕ′ )  
y1 = Sin( ϕ′ ) \* Cos( θ′ )  
z1 = Sin( ϕ′ ) \* Sin( θ′ )  
α = ( αc ÷ 216 ) \* π ÷ 180  
β = ( βc ÷ 216 ) \* π ÷ 180  
γ = ( γc ÷ 216 ) \* π ÷ 180  
x2 = Cos( β ) \* Cos ( γ ) \* x1 − Cos( β ) \* Sin( γ ) \* y1 + Sin( β ) \* z1 (D‑XX)y2 = ( Cos( α ) \* Sin( γ ) + Sin( α ) \* Sin( β ) \* Cos( γ ) ) \* x1 +  
 ( Cos( α ) \* Cos( γ ) − Sin( α ) \* Sin( β ) \* Sin( γ ) ) \* y1 −  
 Sin( α ) \* Cos( β ) \* z1z2 = ( Sin( α ) \* Sin( γ ) − Cos( α ) \* Sin( β ) \* Cos( γ ) ) \* x1 +  
 ( Sin( α ) \* Cos( γ ) + Cos( α ) \* Sin( β ) \* Sin( γ ) ) \* y1 +  
 Cos( α ) \* Cos( β ) \* z1ϕ = Atan2( y2, x2 ) \* 180 ÷ π  
θ = Asin( z2 ) \* 180 ÷ π

# References

[1] JCTVC-AH1012, “Annotated regions and fisheye video information SEI messages for HEVC (Draft 2)”, J. Boyce, Y.-K. Wang, G. J. Sullivan, Jan., 2019, Marrakech

# Patent rights declaration(s)

**LG Electronics, Inc. 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).**