ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/326504559

JVET-H1004: Algorithm descriptions of projection format conversion and video quality metrics in 360Lib

Technical Report · July 2018

CITATIONS
51

READS
3,062

3 authors, including:

Yan Ye
Assumption University of Thailand
96 PUBLICATIONS **2,335** CITATIONS
SEE PROFILE

Jill M Boyce
Nokia
241 PUBLICATIONS **3,821** CITATIONS
SEE PROFILE

All content following this page was uploaded by Jill M Boyce on 20 July 2018.

The user has requested enhancement of the downloaded file.

Joint Video Exploration Team (JVET)
of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11
8th Meeting: Macao, CN, 18–24 Oct. 2017

Document: JVET-H1004

Title:	**Algorithm descriptions of projection format conversion and video quality metrics in 360Lib Version 5**
Status:	Output Document of JVET
Purpose:	360Lib algorithm description

Author(s) or Contact(s):	Yan Ye	Tel:
	InterDigital Communications	Email: yan.ye@interdigital.com
	Elena Alshina	elena_a.alshina@samsung.com
	Samsung Electronics	jill.boyce@intel.com
	Jill Boyce	
	Intel	
Source:	Editors	

Abstract

The Joint Video Exploration Team (JVET) of ITU-T VCEG (Q6/16) and ISO/IEC MPEG (JTC 1/SC 29/WG 11) is studying the potential need to include 360-degree video coding technologies in a future video coding standard. The JVET has established a 360Lib software package that can perform projection format conversion between various projection formats as a standalone conversion tool, or in combination with encoding and decoding using HM or JEM reference software. Among the projection formats 360Lib supports, viewport generation using rectilinear projection is also included. Further, 360Lib supports a number of objective 360-degree video quality metrics. This document describes the algorithms used for projection format conversion and video quality metrics in the 360Lib-5.0 software release.

Compared to JVET-G1003, the following changes are included: -----------Release 1-----------

- (Ed. YY) JVET-H0056, AHG8: An Update on RSP Projection
- (Ed. YY) fix for WS-PSNR weight calculation for the ECP projection format

Compared to JVET-F1003, the following agreed changes in 360Lib software are included: -----------Release 1-----------

- (Ed. YY) JVET-G0051, adjusted equal-area projection (AEP)
- (Ed. YY) JVET-G0056, equi-angular cubemap projection (EAC)
- (Ed. YY) JVET-G0074, equatorial cylindrical projection (ECP) with padding
- (Ed. YY) JVET-G0088, WS-PSNR calculation for ACP
- (Ed. YY) JVET-G0097, EAP-based segmented sphere projection (SSP) with padding
- (Ed. JB/YY) JVET-G0098, Padded ERP (PERP) projection format
- (Ed. YY) general editorial improvements (restructuring of sections, etc)
-

Compared to JVET-E1003, the following changes are included: -----------Release 1-----------

- (Ed. YY) JVET-F0025, adjusted cubemap projection (ACP)
- (Ed. YY) JVET-F0036, rotated sphere projection (RSP)
- (Ed. YY) JVET-F0042, cross format S-PSNR-NN fix
- (Ed. YY) JVET-F0053, compact octahedron projection with padding improvement
- (Ed. YY) JVET-F0065, modifications for spherical rotation
- (Ed. YY) general editorial improvements
- (Ed. EA) updated CISP padding

1 Overview

The Joint Video Exploration Team (JVET) of ITU-T VCEG (Q6/16) and ISO/IEC MPEG (JTC 1/SC 29/WG 11) is working on 360-degree video coding as part of the explorations being conducted for developing coding technologies for future video coding standards. The JVET has established a 360Lib software package for 360-degree video coding and processing [1], and defined the common test condition (CTC) for 360-degree video coding [2]. This document describes the algorithms implemented in 360Lib for projection format conversion and 360-degree video quality evaluation.

The 360Lib software is written in C++, following similar programming styles used by the HM and JEM reference software. 360Lib can be used as standalone application to perform projection format conversion and quality metric calculations. It is also integrated with the latest versions of HM and JEM, and can be used in combination with HM or JEM to perform projection format conversion (before and/or after coding), compression, and quality metric calculation altogether, without the need to store intermediate YUV sequences at intermediate steps. Example configuration files according to the common test conditions for 360 video [2], and a software usage manual are included in the 360Lib package. 360Lib reference software was agreed to be established at the 5th JVET meeting in Oct 2016. The first version, 360Lib v1.0, was officially released in Dec 2016. The latest 360Lib v5.0 was recently released in Nov 2017.

The projection formats supported in 360Lib, as well as their corresponding indices, are listed in Table 1. 360Lib supports projection format conversion between any pair of these projection formats, except for indices 4 and 6, which are used for viewport generation and CPP-PSNR calculation, respectively. More details about each of the projection formats will be presented in Section 2, followed by the detailed discussion of the conversion process in Section 3.

Table 1. Projection formats supported in 360Lib

Index	Projection format
0	Equirectangular [3] (ERP)
1	Cubemap [4] (CMP)
2	Adjusted Equal-area [5][20] (AEP)
3	Octahedron [6] (OHP)
4	Viewport generation using rectilinear projection [7]
5	Icosahedron [8] (ISP)
6	Crasters Parabolic Projection for CPP-PSNR calculation [9]
7	Truncated Square Pyramid [10] (TSP)
8	Segmented Sphere Projection [11] (SSP)
9	Adjusted Cubemap Projection [15] (ACP)
10	Rotated Sphere Projection [17][28][28] (RSP)
11	Equi-angular Cubemap Projection [21] (EAC)
12	Equatorial Cylindrical Projection [22] (ECP)

Depending on the specific projection format used to represent the spherical video, different areas of the sphere are sampled at different densities on the 2D plane. For example, the commonly used ERP format oversamples the sphere at the poles, resulting in over-stretched top and bottom areas on the ERP picture. When using CMP format, spherical positions corresponding to the center of a CMP face are sampled more sparsely compared to those corresponding to the sides of the face. Because projecting a spherical 360-degree video onto 2D planes is a non-linear process, the conventional PSNR, which weighs the sample errors at each 2D position equally, is not a suitable quality metric for 360-degree video. 360Lib supports a number of 360-degree video quality metrics to address the uneven sampling problem in the 2D projected video. Further details about these metrics are provided in Section 4.

The rest of this document is organized as follows. Section 2 provides detailed description of each projection format supported in 360Lib. The viewport generation process using rectilinear projection is also discussed. Section 3 describes the conversion process between any two projection formats. Section 4 describes the objective quality metrics supported in 360Lib, as well as how objective quality metrics are measured in the 360 video CTC. Throughout this document, equations are defined using general trigonometry functions. The corresponding standard C functions used to implement these trigonometry functions in 360Lib software are provided in Section 5.

2 Detailed descriptions of projection formats supported in 360Lib

The 3D XYZ coordinate system as shown in Figure 1 is used in 360Lib to describe the 3D geometry of each projection format representation. Starting from the center of the sphere, X axis points toward the front of the sphere, Y axis points toward the top of the sphere, and Z axis points toward the right of the sphere.

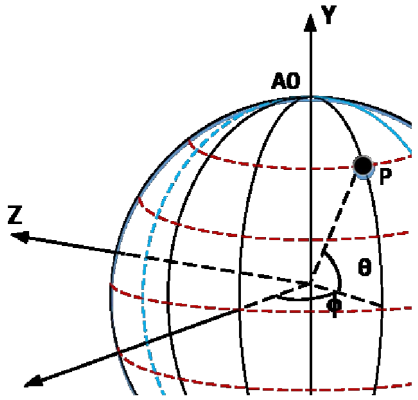


Figure 1. 3D XYZ coordinate definition used in 360Lib, A3 is the equator

Figure 1 shows the internal (X, Y, Z) coordinate system based on the right-hand coordinate system used in 360Lib. Rotation of this coordinate system is supported, and will be described in Section 3. The sphere can be sampled with longitude (ϕ) and latitude (θ). The longitude ϕ is in the range [−π, π], and latitude θ is in the range [−π/2, π/2], where π is the ratio of a circle's circumference to its diameter. In 360Lib, the longitude ϕ is defined by the angle starting from X axis in counter-clockwise direction as shown in Figure 1. The latitude θ is defined by the angle from the equator toward Y axis as shown in Figure 1. The (X, Y, Z) coordinates on the unit sphere can be evaluated from (ϕ, θ) using (1) (2) (3).

$$X = \cos(\theta)\cos(\varphi) \tag{1}$$

$$Y = \sin(\theta) \tag{2}$$

$$Z = -\cos(\theta)\sin(\varphi) \tag{3}$$

Inversely, the longitude and latitude (ϕ, θ) can be evaluated from (X, Y, Z) coordinates using (4)(5).

$$\varphi = \tan^{-1}(-Z/X) \tag{4}$$

$$\theta = \sin^{-1}(Y/(X^2+Y^2+Z^2)^{1/2}) \tag{5}$$

A 2D plane coordinate system is defined for each face in the 2D projection plane. Whereas some of the projection formats in 360Lib have only one face (such as ERP and EAP), other projection formats have multiple faces. In order to generalize the 2D coordinate system, a face index is defined for each face in the 2D projection plane. Each face is mapped to a 2D plane, referred as the uv plane, associated with one face index. The 2D image sampling grid is defined in the uv plane. We refer to the sampling point position as (m, n), where m and n are the column and row coordinates of the sampling position. Figure 2 shows the sampling coordinates defined in the uv plane for ERP projection. The orange circles are the sampling points (m, n). In 360Lib, in order to arrange all sampling points in a symmetric manner in both directions, there is a shift between the origin of (u, v) coordinates and the origin of (m, n) coordinates, as shown in Figure 2.

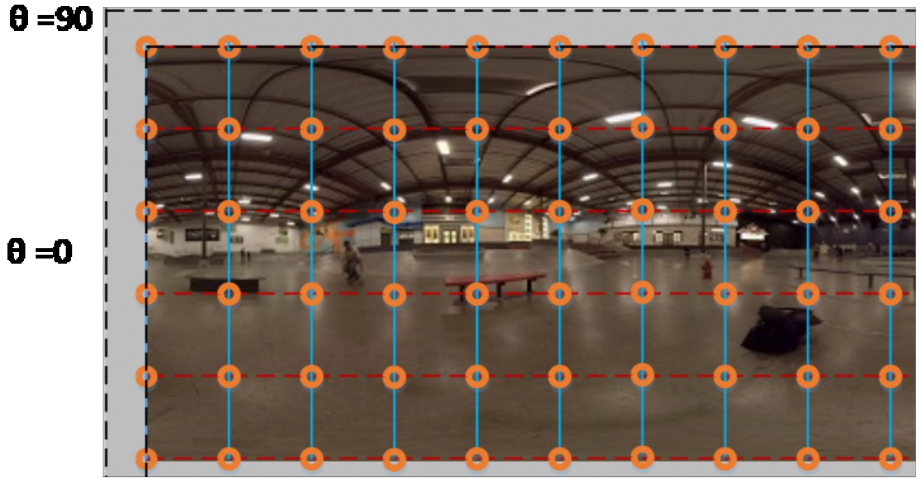


Figure 2. Sampling coordinate definition in (u, v) plane in 360Lib

Finally, we assume W and H are the width and height of a face, respectively. With these notations and coordinate systems, this section describes the conversion between sampling point (f, m, n) (f is the face index) and 3D point position (X, Y, Z) for each projection format supported in 360Lib. In the 360Lib software package, the conversion from (f, m, n) to (X, Y, Z) is implemented by the function map2Dto3D(), and the conversion from (X, Y, Z) to (f, m, n) is implemented by the function map3Dto2D().

2.1 Equi-rectangular projection format (ERP) with optional padding

The ERP projection format is the most widely used projection format for representing 360-degree video on a 2D plane. It is the default projection format in 360Lib.

ERP has only one face. Therefore, the face index f for ERP is always set to 0. In the uv plane, u and v are in the range [0, 1].

For 2D-to-3D coordinate conversion, we first start from a given sampling position (m, n), and calculate (u, v) using (6)(7).

$$u = (m + 0.5) / W, 0 \leq m < W \tag{6}$$

$$v = (n + 0.5) / H, 0 \leq n < H \tag{7}$$

Then, the longitude and latitude (ϕ, θ) in the sphere can be calculated from (u, v) using (8)(9):

$$\varphi = (u - 0.5) * (2 * \pi) \tag{8}$$

$$\theta = (0.5 - v) * \pi \tag{9}$$

Finally, (X, Y, Z) can be calculated from Equation (1)(2)(3).

For 3D-to-2D coordinate conversion starting from (X, Y, Z), (ϕ, θ) is first calculated using (4)(5). Then, (u, v) is calculated by solving Equations (8)(9). Finally, (m, n) is calculated by solving Equation (6)(7).

In order to reduce the seam artifacts in reconstructed viewports that encompass the left and right boundaries of the ERP picture, padding of 8 luma samples is applied on each of the left and the right sides of the picture. Configuration option is provided to turn ERP padding on/off. By default, padding is turned on for the ERP format in 360 CTC.

When ERP with padding is turned on, the padded ERP picture is encoded. After decoding, the reconstructed ERP with padding is converted back to reconstructed ERP by blending the duplicated samples or cropping the padded areas (blending or cropping is denoted as post-processing in the 360 video testing workflow depicted in Figure 19). Configuration option is provided in 360Lib software to choose blending or cropping. By default, blending is turned on for reconstructed ERP with padding.

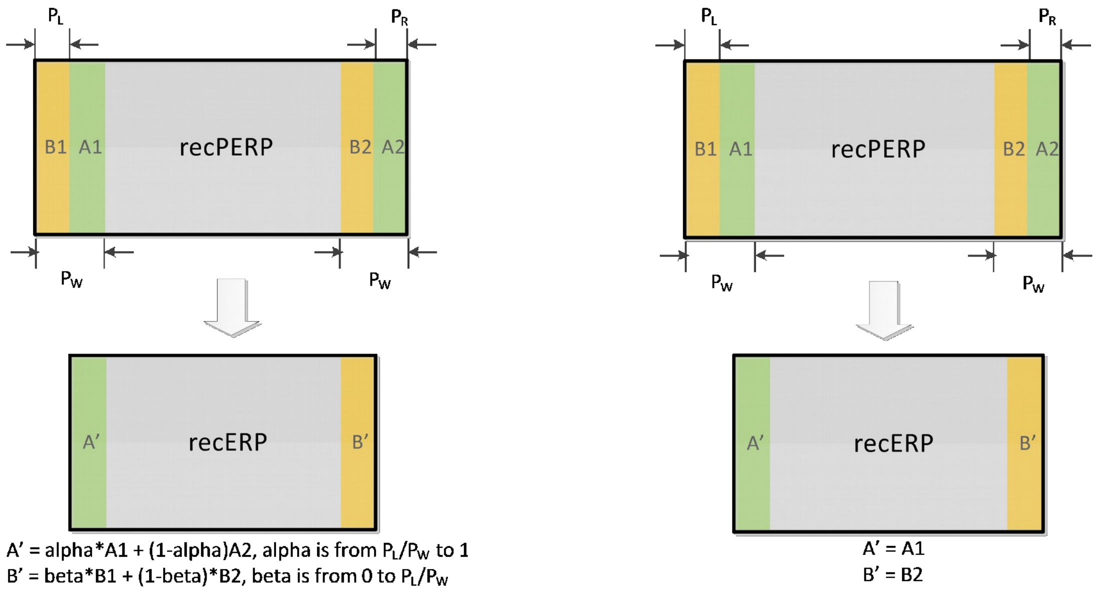

P_L
P_R
B1
A1
recPERP
B2
A2
P_W
P_W
A'
recERP
B'
A' = alpha*A1 + (1-alpha)A2, alpha is from P_L/P_W to 1
B' = beta*B1 + (1-beta)*B2, beta is from 0 to P_L/P_W
P_L
P_R
B1
A1
recPERP
B2
A2
P_W
P_W
A'
recERP
B'
A' = A1
B' = B2

Figure 3. Post-processing of ERP with padding: convert padded ERP (recPERP) to unpadded ERP (recERP) by (a) blending or (b) cropping.

Figure 3 illustrates the blending and cropping operations. Let recPERP be the reconstructed ERP with padding before post-processing, recERP be the reconstructed ERP without padding after post-processing. As shown in Figure 3 (a), the duplicated samples of the recPERP are blended by applying a distance-based weighted averaging, specifically, the region A' is generated by blending $A1$ with $A2$, and the region B' is generated by blending $B1$ with $B2$ using the following procedure.

Denote the width and height of unpadded recERP as W and H, and denote the left and right padding widths as P_L and P_R, respectively. The total padding width is set to $P_W = P_L + P_R$. The following blending operation applies to convert recPERP to recERP:

For a sample $recERP(j,i)$ in A', $i=[0,P_R-1], j=[0,H-1]$

$$recERP(j,i)=(recPERP(j,i+P_L)*(i+P_L)+recPERP(j,i+P_L+W)*(P_R-i)+(P_W \gg 1))/P_W$$

For a sample $recERP(j,i)$ in B', $i=[W-P_L,W-1], j=[0,H-1]$

$$recERP(j,i)=(recPERP(j,i+P_L)*(P_R-i+W)+recPERP(j,i+P_L-W)*(i-W+P_L)+(P_W \gg 1))/P_W$$

As shown in right picture of Figure 3, cropping directly discards the padded samples in recPERP to obtain recERP.

2.2 *Cubemap projection format (CMP)*

The CMP projection has 6 square faces in total, labelled as PX, PY, PZ, NX, NY, NZ (with “P” standing for “positive” and “N” standing for “negative”), in Figure 4.

specifies the face index values corresponding to each of the six CMP faces in 360Lib.

The uv plane definition for each CMP face in 360Lib is also shown in Figure 4. Each face in the uv plane is a 2x2 square, with u and v being defined in the range of [−1, 1].

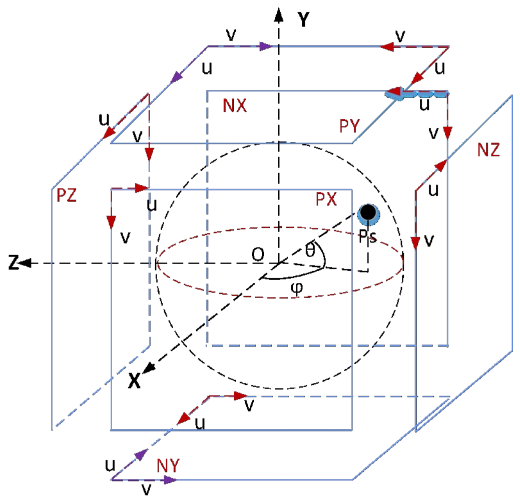


Figure 4. Coordinates definition for CMP

Table 2. Face index of CMP

Face index	Face label	Notes
0	PX	Front face with positive X axis value
1	NX	Back face with negative X axis value
2	PY	Top face with positive Y axis value
3	NY	Bottom face with negative Y axis value
4	PZ	Right face with positive Z axis value
5	NZ	Left face with negative Z axis value

Denote the dimension of any square face as $A \times A$. For 2D-to-3D coordinate conversion, given the position (m, n) on a given face f, (u, v) is first calculated as:

$$u = (m + 0.5) * 2 / A - 1, \quad 0 \leq m < A \tag{10}$$

$$v = (n + 0.5) * 2 / A - 1, \quad 0 \leq n < A \tag{11}$$

Then, the 3D coordinates (X, Y, Z) are derived using Table 3 given the position (u, v) and the face index f.

Table 3. (X, Y, Z) derivation given (u, v) and the face index f

f	X	Y	Z
0	1.0	−v	−u
1	−1.0	−v	u
2	u	1.0	v
3	u	−1.0	−v
4	u	−v	1.0

5	−u	−v	−1.0

For 3D-to-2D coordinate conversion, given (X, Y, Z), the (u, v) and face index f is calculated according to Table 4. Then, (m, n) on the face f is calculated by solving the Equation (10)(11).

Table 4. Derivation of (u, v) and the face index f given (X, Y, Z)

Condition	f	u	v
$\lvert X\rvert \geq \lvert Y\rvert$ and $\lvert X\rvert \geq \lvert Z\rvert$ and $X > 0$	0	$-Z/\lvert X\rvert$	$-Y/\lvert X\rvert$
$\lvert X\rvert \geq \lvert Y\rvert$ and $\lvert X\rvert \geq \lvert Z\rvert$ and $X < 0$	1	$Z/\lvert X\rvert$	$-Y/\lvert X\rvert$
$\lvert Y\rvert \geq \lvert X\rvert$ and $\lvert Y\rvert \geq \lvert Z\rvert$ and $Y > 0$	2	$X/\lvert Y\rvert$	$Z/\lvert Y\rvert$
$\lvert Y\rvert \geq \lvert X\rvert$ and $\lvert Y\rvert \geq \lvert Z\rvert$ and $Y < 0$	3	$X/\lvert Y\rvert$	$-Z/\lvert Y\rvert$
$\lvert Z\rvert \geq \lvert X\rvert$ and $\lvert Z\rvert \geq \lvert X\rvert$ and $Z > 0$	4	$X/\lvert Z\rvert$	$-Y/\lvert Z\rvert$
$\lvert Z\rvert \geq \lvert X\rvert$ and $\lvert Z\rvert \geq \lvert Y\rvert$ and $Z < 0$	5	$-X/\lvert Z\rvert$	$-Y/\lvert Z\rvert$

For projection formats that have multiple faces, there are different ways to arrange the faces onto one 2D picture. 360Lib allows the faces to be arranged in a flexible manner. Parameters are provided in the configuration files to allow the user to specify whether to rotate a given face by 0, 90, 180 or 270 degrees, and where to place that given face on the 2D picture. In the default CMP configuration files included in 360Lib software, the top and bottom faces (PZ and NZ) are rotated such that the effective uv coordinates are depicted by the purple axes instead of the red axes in Figure 4. For further details on how to use the configuration parameters, readers are referred to the user guide in 360Lib at [1].

The CMP 3x2 face packing arrangement in the 360 CTC [2] is shown in Figure 5, with face indices specified according to Table 2. The faces in the bottom row are rotated by 90 degrees.

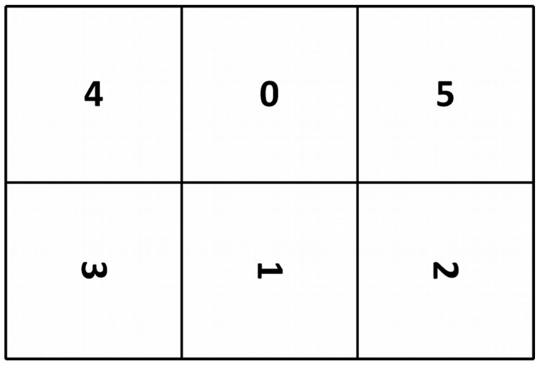


Figure 5. CMP3x2 frame packing

2.3 Adjusted cubemap projection format (ACP)

The ACP format shares the same face definition as the CMP format shown in Figure 4 and Table 2. The uv plane definition for each ACP face is also the same as CMP, where each face in the uv plane is a 2x2 square, with u and v being defined in the range of [−1, 1].

Denote the dimension of any square face as $A \times A$. For 2D-to-3D coordinate conversion, given the position (m, n) on a given face f, (u’, v’) is first calculated using equations (10) and (11). Then, (u, v) is calculated by adjusting (u’, v’) according to the following:

$$u = sgn(u')\frac{0.34-\sqrt{0.34^2-0.09\lvert u'\rvert}}{0.18} \tag{12}$$

$$v = sgn(v')\frac{0.34-\sqrt{0.34^2-0.09\lvert v'\rvert}}{0.18} \tag{13}$$

Finally, the 3D coordinates (X, Y, Z) are derived using Table 3 given the position (u, v) and the face index f.

For 3D-to-2D coordinate conversion, given (X, Y, Z), the (u', v') and face index f is calculated according to Table 4. Then, (u, v) is calculated by adjusting the (u', v') using the following:

$$u = sgn(u')(-0.36u'^2 + 1.36|u'|) \tag{14}$$

$$v = sgn(v')(-0.36v'^2 + 1.36|v'|) \tag{15}$$

Finally, (m, n) on the face f is calculated by solving equations (10) and (11).

Denote the forward adjustment functions in (12)(13) as f(x) and the backward adjustment functions in (14)(15) as g(x) (x may be u or v), Figure 6 depicts the shapes of f(x) and g(x).

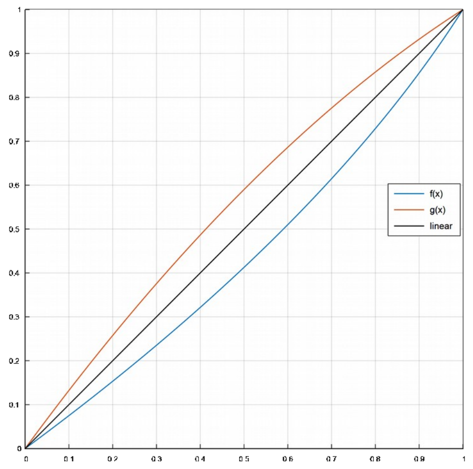


Figure 6. Forward and backward adjustment functions used in ACP

2.4 Equi-angular cubemap projection (EAC)

Description of the ACP format in 2.3 applies to the EAC format except for the forward and backward adjustment functions (f(x) in (12)(13) and g(x) in (14)(15)), which are replaced by the following equations.

For the forward adjustment function f(x), with x being u or v, the EAC format uses the following:

$$u = \tan(u' * \pi / 4.0) \tag{16}$$

$$v = \tan(v' * \pi / 4.0) \tag{17}$$

For the backward adjustment function g(x), with x being u or v, the EAC format uses the following:

$$u = \frac{4.0}{\pi} * \tan^{-1}(u') \tag{18}$$

$$v = \frac{4.0}{\pi} * \tan^{-1}(v') \tag{19}$$

2.5 Adjusted equal-area projection format (AEP)

The conventional equal-area projection format (EAP) [5] had been supported in earlier 360Lib software releases (up to 360Lib-3.x). Starting from 360Lib-4.0, it was replaced by the AEP format to improve compression performance and to avoid the visual quality problem around the Poles that could be caused by the conventional EAP format [20]. Same as ERP, there is only one face for AEP, and the face index for AEP is always set to 0. In the uv plane, u and v are in the range [0, 1].

For 2D-to-3D coordinate conversion, given the sampling position (m, n), we can first calculate (u, v) with Equations (6)(7). Then, the longitude and latitude (ϕ, θ) on the sphere can be calculated from (u, v) as:

$$\varphi = (u - 0.5) * (2 * \pi) \tag{20}$$

$$\theta = \frac{1}{\beta} * \sin^{-1}((1.0 - 2 * v) * \sin(0.5 * \pi * \beta)) \tag{21}$$

Finally, (X, Y, Z) can be calculated using Equations (1)(2)(3).

For 3D-to-2D coordinate conversion, given (X, Y, Z), first (ϕ, θ) is calculated using Equations (4)(5). Then, (u, v) is calculated using the following equations:

$$u = \varphi / (2 * \pi) + 0.5 \tag{22}$$

$$v = -0.5 * \sin(\beta * \theta) / \sin(0.5 * \beta * \pi) + 0.5 \tag{23}$$

Finally, (m, n) is calculated by solving Equations (6)(7). The value of β is selected by experiment, and is set to $\beta = 1/1.4$ in the 360Lib software.

2.6 Octahedron projection format (OHP)

The OHP projection format has 8 triangle faces and 6 vertices, labelled as {F0, F1, F2, F3, F4, F5, F6, F7} and {V0, V1, V2, V3, V4, V5}, respectively, as shown in Figure 7. The XYZ coordinates of the 6 vertices are defined in Table 5 for both non-compact and compact OHP.

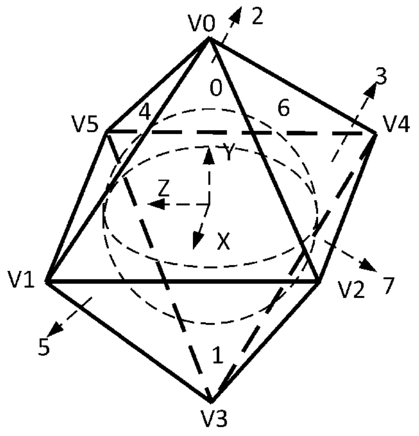


Figure 7. Definitions of faces and vertices for OHP in 360Lib

Table 5. Definition of the XYZ coordinates of OHP's vertices

Vertex	(X, Y, Z) definition in non-compact OHP	(X, Y, Z) definition in compact OHP
V0	$(0, 2^{0.5}, 0)$	$(0, 0, 2^{0.5})$
V1	(1, 0, 1)	(1, −1, 0)
V2	(1, 0, −1)	(1, 1, 0)
V3	$(0, -2^{0.5}, 0)$	$(0, 0, -2^{0.5})$
V4	(−1, 0, −1)	(−1, 1, 0)

V5	(−1, 0, 1)	(−1, −1, 0)

Unlike the CMP and ACP projection formats, the OHP projection (and ISP in the next section) contains triangle faces. The triangle faces need to be packed carefully, in order to minimize discontinuity between neighboring faces and to improve coding efficiency. Further, for better coding efficiency, it is also desirable to minimize the number of inactive samples in the packed 2D picture. "Inactive samples" are defined as samples on the 2D picture that do not correspond to any sample positions on the sphere; they are filled with default gray colour in 360Lib. 360Lib supports two type of packing for OHP: the non-compact OHP packing and the compact OHP (COHP) packing [12]. Table 6 shows how the 6 vertices are used to define the 8 triangle faces in the non-compact OHP and in the compact OHP in 360Lib.

Table 6. Definition of the vertices of each of the triangle faces for non-compact OHP and compact OHP in 360Lib

Face index	Vertex definition in non-compact OHP	Vertex definition in compact OHP
0	{V0, V1, V2}	{V0, V1, V2}
1	{V3, V2, V1}	{V3, V2, V1}
2	{V0, V4, V5}	{V0, V4, V5}
3	{V3, V5, V4}	{V3, V5, V4}
4	{V0, V5, V1}	{V1, V0, V5}
5	{V3, V1, V5}	{V1, V5, V3}
6	{V0, V2, V4}	{V2, V4, V0}
7	{V3, V4, V2}	{V2, V3, V4}

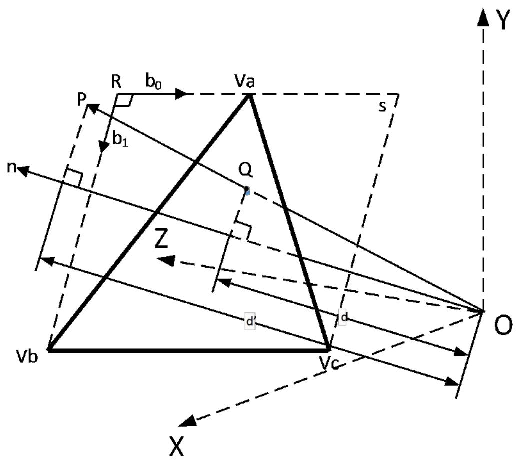


Figure 8. Projection of a point to one face of octahedron

Before we define the 2D-to-3D and 3D-to-2D coordinate mapping processes, we define a few notations in the 3D geometry. Figure 8 shows a side view of a triangle face in OHP in 3D space. The top-left point of the rectangle containing the triangle is denoted as the point R. Denote the XYZ coordinates of R as (X_R, Y_R, Z_R), and denote R's position in vector form as $\vec{R}$. Denote the vertices of the triangle face as $\vec{V}_a$, $\vec{V}_b$, and $\vec{V}_c$ (the positions of $\vec{V}_a$, $\vec{V}_b$, and $\vec{V}_c$ on a given face are provided in Table 6). $\vec{R}$ is derived according to Equation (24).

$$\vec{R}=\vec{V}_a+(\vec{V}_b-\vec{V}_c)/2 \tag{24}$$

Two base vectors $\acute{b}_0$, $\acute{b}_1$ are defined for the two orthogonal directions on the face, and are derived according to Equations (25)(26).

$$\acute{b}_0=(\acute{V}_c-\acute{V}_b)/¿\vee\acute{V}_c-\acute{V}_b\vee¿ \tag{25}$$

$$\acute{b}_1=(\acute{V}_b+\acute{V}_c-2\acute{V}_a)/¿\vee\acute{V}_b+\acute{V}_c-2\acute{V}_a\vee¿ \tag{26}$$

The norm vector of the face $\acute{n}$ is derived as:

$$\acute{n}=(\acute{b}_1\times\acute{b}_0)/¿\vee\acute{b}_1\times\acute{b}_0\vee¿ \tag{27}$$

With these notations, the following steps are applied in 2D-to-3D coordinate mapping. First, given the sampling positon (m, n), (u, v) is calculated as:

$$u = (m+0.5)*2/W \tag{28}$$

$$v = (n+0.5)* \sqrt{3} /H \tag{29}$$

Then, the 3D coordinates (X, Y, Z) are calculated with Equations (30) (31) (32), where ($\acute{V}$)$_X$, ($\acute{V}$)$_Y$ and ($\acute{V}$)$_Z$ denote the X, Y, Z coordinates of vector $\acute{V}$, respectively.

$$X = X_R + (u* \acute{b}_0)_X +(v* \acute{b}_1)_X \tag{30}$$

$$Y = Y_R + (u* \acute{b}_0)_Y +(v* \acute{b}_1)_Y \tag{31}$$

$$Z = Z_R + (u* \acute{b}_0)_Z +(v* \acute{b}_1)_Z \tag{32}$$

For 3D-to-2D coordinate mapping, the following is applied. Denote a point P with coordinates (X, Y, Z) in 3D space as $\acute{P}$ in the vector form, we first determine the face index to which it belongs using Equation (33).

$$f=\underset{i,i\in[0,7]}{argmax}\left(\acute{P}\cdot\acute{n}_i\right) \tag{33}$$

Then, the projection point Q on the face f is calculated as follows, where $\acute{n}_f$ is the norm vector of the face f:

$$\acute{Q}=\frac{\acute{R}\cdot\acute{n}_f}{\acute{P}\cdot\acute{n}_f}\acute{P} \tag{34}$$

Then, the 2D point (u, v) is calculated as follows using the base vectors $\acute{b}_0$, $\acute{b}_1$ of the face f (face index f is ignored in the base vector notations):

$$u = \left(\acute{Q}-\acute{R}\right)\cdot\acute{b}_0 \tag{35}$$

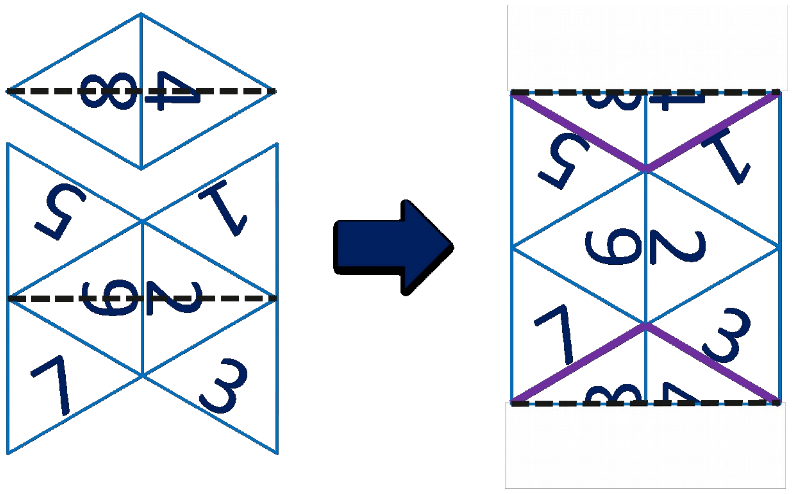
$$v = \left(\acute{Q}-\acute{R}\right)\cdot\acute{b}_1 \tag{36}$$

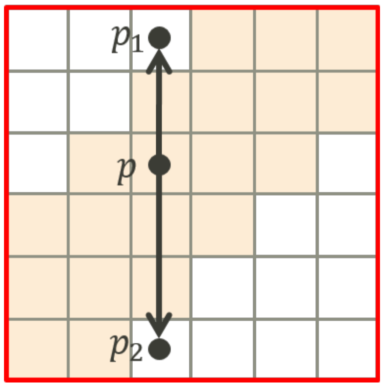
Finally, the position (m, n) in the sampling grid is calculated as:

$$m = u*(W/2) -0.5 \tag{37}$$

$$n = v*(H/ \sqrt{3}) -0.5 \tag{38}$$

When COHP is used [12], the eight triangle faces are re-arranged and packed into a rectangular picture as shown in Figure 9 (a) (the numeric face indexes are as labelled in Table 6). The thick purple lines in Figure 9 (a) represent discontinuous edges in the packed picture. Such discontinuous edges can be difficult to code and could cause visual artifacts. Therefore, padding is applied to smooth these discontinuous edges [16]. As shown in Figure 9 (b), a band of 16 padded samples are added between the discontinuous edges; the 16 padded samples (p) are vertically interpolated using distance-based linear blending of the neighboring samples (p1 and p2).


(a) Packing of eight triangle faces into one rectangular picture in COHP



(b) Padding along discontinuous edges in COHP

Figure 9. Face packing and padding used in COHP in 360Lib

2.7 Icosahedron projection format (ISP)

The ISP projection format has 12 vertices and 20 faces. Each face is a triangle in the same shape as the face of an octahedron. 360Lib also supports two types of packing for ISP: the non-compact packing and the compact packing. Figure 10 shows the definitions of those vertices and faces for non-compact ISP (left) and compact ISP (right). Table 7 defines the XYZ coordinates of those vertices, and Table 8 defines the vertices of each of the 20 faces in 360Lib, for both non-compact and compact ISP.

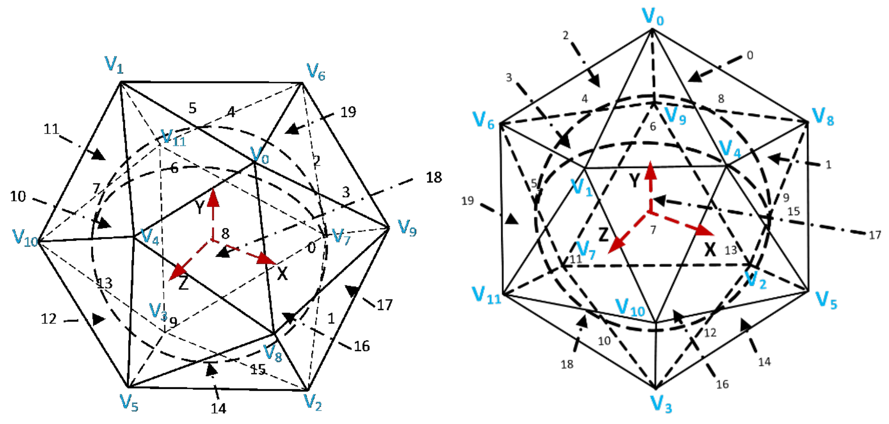


Figure 10. Definition of ISP vertices in 360Lib. Left: vertices used in non-compact ISP; right: vertices used in compact ISP

Table 7. Coordinates definition of ISP vertices, $c=(\sqrt{5}+1)/2$

Vertex	Vertices for non-compact ISP	Vertices for compact ISP
V0	(1, c, 0)	(0,1.902, 0)
V1	(−1, c, 0)	(0.526,0.851, 1.618)
V2	(1, −c, 0}	(−0.526, −0.851, −1.618)
V3	(−1, −c, 0)	(0, −1.902, 0)
V4	(0, 1, c)	(1.701, 0.851, 0)
V5	(0, −1, c)	(1.376, −0.851, −1)
V6	(0, 1, −c)	(−1.376, 0.851, 1)
V7	(0, −1, −c)	(−1.701, −0.851, 0)
V8	(c, 0, 1}	(0.526, 0.851, −1.618)
V9	(c, 0, −1)	(−1.376, 0.851, −1)
V10	(−c, 0, 1)	(1.376, −0.851, 1)
V11	(−c, 0, −1)	(−0.526, 0.851, 1.618)

Table 8. Definition of triangle faces of icosahedron

Face index	F for non-compact ISP	F for compact ISP
0	{V0, V8, V9}	{V8, V9, V0}
1	{V2, V9, V8}	{V2, V9, V8}
2	{V0, V9, V6}	{V0, V9, V6}
3	{V7, V6, V9}	{V7, V6, V9}
4	{V0, V6, V1}	{V6, V1, V0}
5	{V11, V1, V6}	{V11, V1, V6}

6	{V0, V1, V4}	{V4, V0, V1}
7	{V10, V4, V1}	{V10, V4, V1}
8	{V0, V4, V8}	{V0, V4, V8}
9	{V5, V8, V4}	{V5, V8, V4}
10	{V3, V10, V11}	{V3, V10, V11}
11	{V1, V11, V10}	{V1, V11, V10}
12	{V3, V5, V10}	{V10, V3, V5}
13	{V4, V10, V5}	{V4, V10, V5}
14	{V3, V2, V5}	{V2, V5, V3}
15	{V8, V5, V2}	{V8, V5, V2}
16	{V3, V7, V2}	{V3, V7, V2}
17	{V9, V2, V7}	{V9, V2, V7}
18	{V3, V11, V7}	{V11, V7, V3}
19	{V6, V7, V11}	{V6, V7, V11}

For non-compact ISP, the same geometry conversion functions used for OHP are shared by ISP. Non-compact ISP simply replaces the definition for vertex coordinates with those defined in Table 8. For non-compact ISP, the 2D-to-3D and 3D-to-2D coordinate mapping processes are exactly the same as those already described for OHP in section 2.6.

For compact ISP (CISP), all 20 triangle faces are compacted to the rectangular frame as shown on Figure 11. While compacting rectangular frame some triangles are split into the 2 parts vertically, some are flipped vertically or horizontally. 4 samples margin horizontal per boundary between 2 triangles are introduced in order to ensure always usage of proper chroma sample for each luma sample in non 4:4:4 formats and keeping resulting frame size multiple of 8. Those extra samples are not used in CISP to sphere (or viewport) projection, so they are just padded using nearest sample from triangle face.

If two triangle faces are next to each other both on icosahedron and on rectangular coding projection then there is no discontinuity. But if two triangle faces are not neighboring on icosahedron but put next to each other on rectangular frame then margin is introduced in order to resolve discontinuity, simplify encoding and reduce artifacts which might appear on triangles boundaries during encoding. Again extra samples from this margin are not used in CISP to sphere (or viewport) projection; so they are just filled using bilinear combination of nearest available samples from opposite edges of adjacent faces (or just copied if only one side neighbor is available). Size of the margin resolving discontinuity is 64 samples in horizontal and 32 samples in vertical directions correspondently.

As shown on Figure 11 (c), CISP has three continuous sets of polygons representing different parts of an icosahedron. Their triangle indexes are:

Set A: 2; 4-1; 6-2; 8; 13-2; 9; 15; 1; 17; 3-2.

Set B: 3-1; 19; 5; 11; 7; 13-1; 18; 10; 12; 14; 16.

Set C: 6-1; 4-2.

There is a vertical discontinuity between faces: 19 and 2; 18 and 0; 16 and 1; 13-1 and 6-1

There is a horizontal discontinuity between faces: 1 and 18; 0 and 16; 14 and 4-2; 12 and 6-1

Displacement for set of triangle faces are summarized in Table 9.

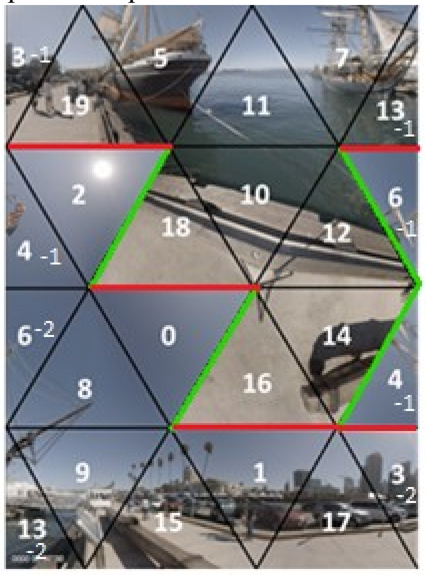
Table 9. Displacement for faces set in CISP

	Set A	Set B	Set C
Vertical Shift	p_v	0	$p_v/2$
Horizontal Shift	0	p_h	$2p_h$

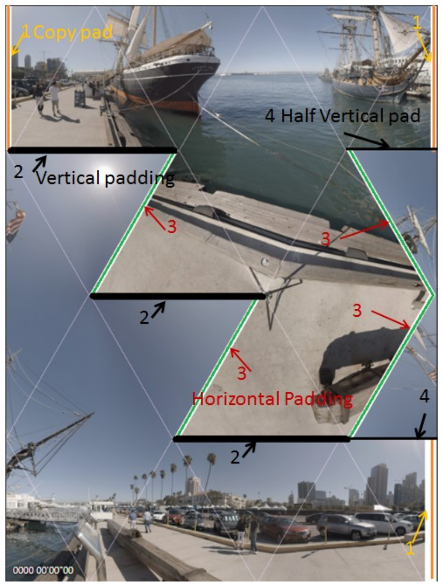
In 360Lib reference software, the padding size is set to p_h=48, p_v=24.

Figure 11.b specifies methods of padding on each boundary:

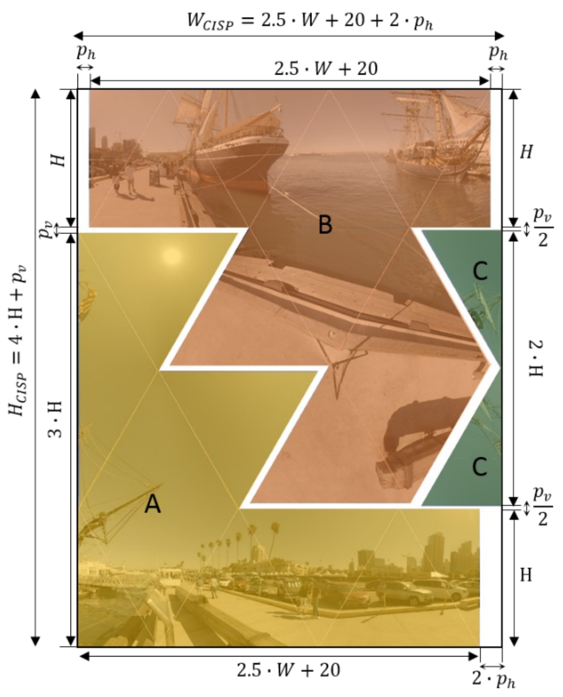
1. for filling p_h pixels on both sides of top row of triangles and $2p_h$ pixels on a right from the bottom row of triangles, copy padding method is used;
2. p_v pixels are padded with bilinear filter;
3. p_h pixels are padded with bilinear filter;
4. $p_v/2$ pixels are padded with bilinear filter.



a)



b)



c)

Figure 11. Compact icosahedral projection layout: a – face indexes; b – padding description; c – face sets and shift description.

The size of CISP rectangular frame $W_{CISP} \times H_{CISP}$ is calculated based on width (W) and height (H) of rectangular faces and horizontal (p_h¿ and vertical (p_v¿ padding sizes as follows:

$$W_{CISP} = 5 \cdot (W/2+4) + 2 \cdot p_h$$

$$H_{CISP} = 4 \cdot H + p_v$$

For W calculation there is multiple 5 because CISP has 5 triangles boundaries in each horizontal line. For H calculation there is multiple 4 because CISP has 4 triangles boundaries in each vertical line.

2.8 Segmented sphere projection format (SSP)

SSP segments the sphere into 3 segments: the north pole, the equator, and the south pole. Figure 12 shows SSP projection and the default SSP frame packing structure, as well as the definition of (u, v) coordinates for each face. The boundaries of the 3 segments are at 45°N and 45°S. The north and south poles are mapped into 2 circles, labelled “0” and “1” in Figure 12. The corners of the two pole segments are inactive samples and filled with the default gray colour. The equatorial segment uses the same projection as ERP. The equatorial segment is split into 4 squares in order to get “faces” of the same size, labelled “2” to “5” in Figure 12. The diameter of the circle is equal to the face size of the equatorial segments because they all have a 90° latitude span.

Denote the dimension of one face as $A \times A$. In 2D-to-3D conversion, a point (m, n) on the face is mapped to a point on the sphere (ϕ, θ) according to Equations (39) to (44), depending on the face it belongs to.

For the circle face f = 0, the following applies:

$$\phi = \arctan 2\left(m + 0.5 - \frac{A}{2}, n + 0.5 - \frac{A}{2}\right) \tag{39}$$

$$\theta=\frac{\pi}{2}\left(1-\frac{r}{A}\right) \tag{40}$$

For the circle face f = 1, the following applies:

$$\phi=\arctan 2\left(m+0.5-\frac{A}{2},\frac{A}{2}-n-0.5\right) \tag{41}$$

$$\theta=\frac{\pi}{2}\left(\frac{r}{A}-1\right) \tag{42}$$

Where $r=\sqrt{\left(m+0.5-\frac{A}{2}\right)^2+\left(n+0.5-\frac{A}{2}\right)^2}$, and $\arctan 2(x,y)$ calculate the inverse tangent of y/x.

For the square faces with face index f, f = 2…5, the following applies:

$$\phi=\left(\frac{m+0.5}{A}+f-4\right)\cdot\frac{\pi}{2} \tag{43}$$

$$\theta=\sin^{-1}\left(\left(0.5-\frac{n+0.5}{A}\right)\cdot 2\sin\left(\frac{\pi}{4}\right)\right) \tag{44}$$

In 3D-to-2D conversion, a point on the sphere (ϕ,θ) is mapped to a point (m,n) on the face according to Equations (45) to (50), depending on the face index f.

For the circle face f = 0, $\theta\in\left(\frac{\pi}{4},\frac{\pi}{2}\right]$, $\phi\in(-\pi,\pi]$, the following applies:

$$m=\frac{A}{2}\left(1+\frac{\left(\frac{\pi}{2}-\theta\right)\sin\phi}{\frac{\pi}{4}}\right)-0.5 \tag{45}$$

$$n=\frac{A}{2}\left(1+\frac{\left(\frac{\pi}{2}-\theta\right)\cos\phi}{\frac{\pi}{4}}\right)-0.5 \tag{46}$$

For the circle face f = 1, $\theta\in\left(\frac{-\pi}{2},-\frac{\pi}{4}\right]$, $\phi\in(-\pi,\pi]$, the following applies:

$$m=\frac{A}{2}\left(1+\frac{\left(\frac{\pi}{2}+\theta\right)\sin\phi}{\frac{\pi}{4}}\right)-0.5 \tag{47}$$

$$n=\frac{A}{2}\left(1-\frac{\left(\frac{\pi}{2}+\theta\right)\cos\phi}{\frac{\pi}{4}}\right)-0.5 \tag{48}$$

For the square faces with face index f, f = 2…5, longitude in the range $\phi \in \left(-\pi+(f-2)\frac{\pi}{2}, -\frac{\pi}{2}+(f-2)\frac{\pi}{2}\right]$ depending on f, and latitude in the range $\theta \in \left[\frac{-\pi}{4}, \frac{\pi}{4}\right]$, the following applies:

$$m=\frac{2\phi}{\pi}A+(4-f)A-0.5 \tag{49}$$

$$n=\frac{A}{2}\left(1-\frac{\sin\theta}{\sin(\frac{\pi}{4})}\right)-0.5 \tag{50}$$

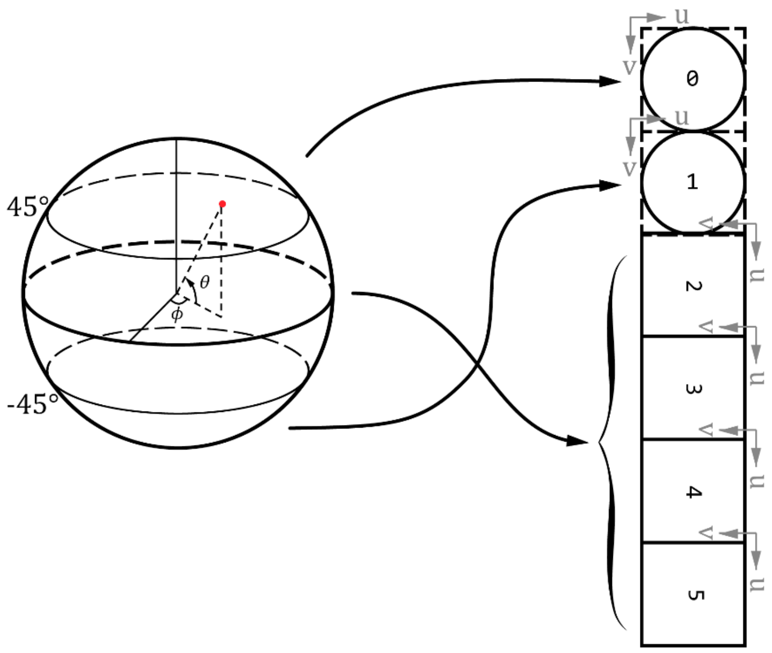


Figure 12. Segmented sphere projection faces specification

In order to improve visual quality of the reconstructed viewports, padding is applied between the SSP segments that represent discontinuous edges [25]. Specifically, padding is applied to the boundary between segments 0 and 1 (that is, between the two pole segments) and the boundary between segments 1 and 2 (that is, between the south pole segment and the equatorial segment). Padding size is set to 8 luma samples on each side; equivalently, a total of 16 luma samples is padded between segments 0 and 1 and between segments 1 and 2.

2.9 Rotated sphere projection format (RSP)

Rotated Sphere Projection (RSP) [17] partitions the sphere into two equal-sized segments and places them on the 2D projection plane in two rows, as shown in Figure 13. On a sphere, the boundary between the two RSP segments looks similar to the stitch line on a tennis ball. Since RSP has a 3:2 aspect ratio like CMP3x2 in Figure 5, in 360Lib, RSP is implemented using 6 faces shown in Figure 13. The face indices are the same as specified Table 2. Faces 4, 0 and 5 represent the top segment; samples on these faces are derived using ERP. Faces 3, 1 and 2 represent the bottom segment; samples on these faces are also derived using ERP projection, but after the sphere is rotated around Y and X axes, as will be described below. In terms of the field of view (FOV) span on the sphere, faces 4, 0 and 5 together span a 270° × 90° FOV, and faces 3, 1 and 2 together span a 90° × 270° FOV.

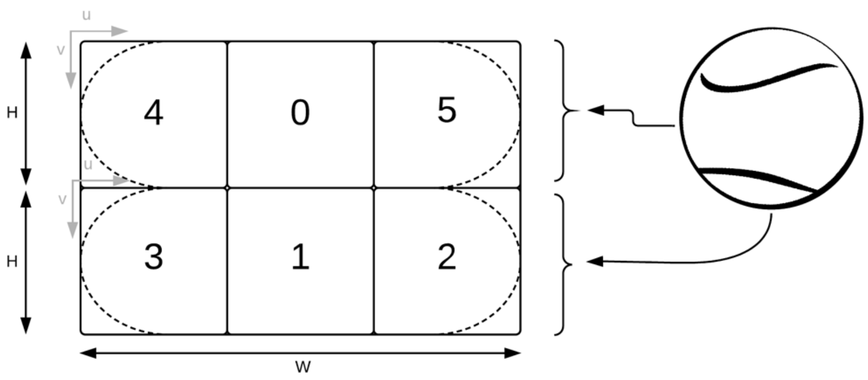


Figure 13. Rotated sphere conversion from sphere to horizontal frame packing

The corner samples of each segment (those that lie outside of the dashed arcs in Figure 13) overlap with samples from the other segment and as a result, they can be greyed out and marked as inactive. The determination of whether a point (m, n) lies inside face *f* is made following the steps below:

- If *f* is 0 or 1, then (m, n) is marked as being inside face.
- Otherwise (if f is 2, 3, 4, or 5), whether (m, n) is inside face *f* is determined as follows:
  1. Clip (m, n) to (m_0, n_0) as follows, with (m_0, n_0) being on a 16x16 sample grid and H being as shown in Figure 13:

$$\square_{\square}\left(\frac{\square}{\square}\right)$$

$$\square_{\square}\left(\frac{\square}{\square}\right)$$

$$\square_{\square}\left(\frac{\square}{\square}\square_{\square}\right)\square_{\square}\square_{\square}\frac{\square}{\square}$$

  2. Given (m_0, n_0) and face index *f*, perform 2D-to-3D conversion followed by 3D-to-2D conversion to obtain (m', n') and face index *f'* to which (m', n') belongs, using the 2D-to-3D and 3D-to-2D conversion process for RSP to be described below.
  3. If *f'* is equal to *f*, then it is determined that the original point (m, n) is inside face *f*. Otherwise, it is determined that (m, n) is outside of face *f* and (m, n) is marked as inactive.

The resulting RSP arc from this operation is aligned on 16x16 luma sample grid and shown inFigure 14 Figure 14.

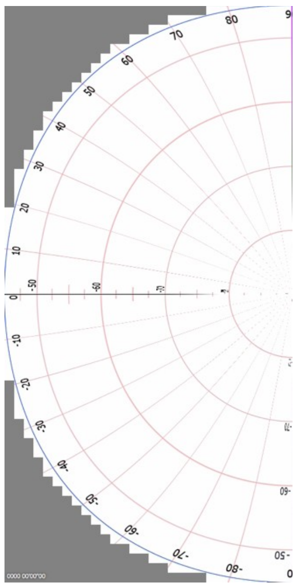


Figure 14. RSP arc on a 16x16 grid

Figure 132D-to-3D and 3D-to-2D conversion equations for faces 4, 0, and 5 are trivial, since they can be directly obtained from ERP. For faces 3, 1, and 2, the following rotation is first performed on the sphere during 2D-to-3D and 3D-to-2D conversions:

- 180° rotation along Y-axis (to bring back side to the front)
- 90° rotation along X-axis (to bring polar area to the equator).

Let (X', Y', Z') be the 3D coordinates on the sphere after performing the above rotation to the 3D point (X, Y, Z):

$$X' = -X \tag{51}$$

$$Y' = -Z \tag{52}$$

$$Z' = -Y \tag{53}$$

Denote the dimension of each of the RSP segments as W×H. The normalized (u,v) coordinates of the faces 4, 0, 5 can be derived from (m,n) as follows:

$$u = \left(\frac{m+0.5}{W}\right) \tag{54}$$

$$v = \left(\frac{n+0.5}{H}\right) \tag{55}$$

Then, the latitude and longitude (ϕ,θ) on the sphere can be derived as follows:

$$\phi = \frac{3\pi}{2}(u-0.5) \tag{56}$$

$$\theta = \frac{\pi}{2}(0.5-v) \tag{57}$$

For 3D-to-2D conversion, a point (m,n) of the top segment can be expressed in terms of (ϕ,θ) as follows:

$$m = W\left(\frac{2\phi}{3\pi}+0.5\right)-0.5 \tag{58}$$

$$n = H\left(\frac{-2\theta}{\pi} + 0.5\right) - 0.5 \qquad (59)$$

Another pair of latitude and longitude angles (ϕ', θ') can be derived from (ϕ, θ) according to the following:

$$\theta \sin \phi$$
$$\cos ¿$$
$$¿$$
$$¿$$
$$\theta' = \sin^{-1} ¿ \qquad (60)$$

$$\phi' = \tan^{-1} \frac{\tan \theta}{-\cos \phi} \qquad (61)$$

For the bottom segment, a point (m, n) on the 2D plane is also derived from Equations (58) and (59) except that instead of (ϕ, θ), the latitude and longitude angles (ϕ', θ') are used.

Given a 3D point (X, Y, Z) on the sphere, the face index f of RSP is determined with two steps:

1. Given (X, Y, Z), an initial face index f' is determined in the same way as CMP, according to Table 4.

2. If the initial face index f' is 0 or 1, then the final face index f is set equal to f'. Otherwise, if the initial face index f' is 2, 3, 4, or 5, then the final face index f is set by adjusting f' according to the conditions listed in Table 10. If the condition in the center column in Table 10 is true, then the adjustment in the right column will be performed. Otherwise, f is set equal to f' directly. The adjustment is needed because a point on face 2, 3, 4, or 5 may end up in an inactive region in Figure 13.

Table 10. Face index update in RSP

Initial index f'	Condition	Final index f

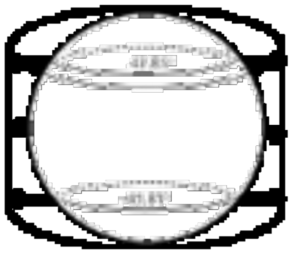
f'=2 or 3	X > 0 and $\lVert (X, Y, Z) - P \rVert > T$, where $P = (0, 1, 0)$ if f' = 2; $P = (0, -1, 0)$ if f' = 3	f = 0 *if* $\frac{-\pi}{4} < \phi < \frac{\pi}{4}$
		f = 4 *if* $\phi \leftarrow \frac{\pi}{4}$
		f = 5 *if* $\frac{\pi}{4} < \phi$

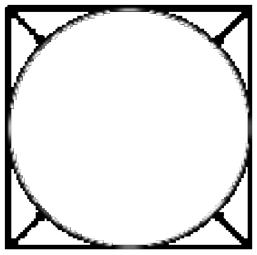
f'=4 or 5	X < 0 and $\lVert (X, Y, Z) - P \rVert > T$, where $P = (0, 0, 1)$ if f' = 4; $P = (0, 0, -1)$ if f' = 5	f = 1 *if* $\theta < 0$ and $\phi' \geq -\frac{\pi}{4}$ or $\theta > 0$ and $\phi' \leq \frac{\pi}{4}$
		f = 2 *if* $\theta > 0$ and $\phi' > \frac{\pi}{4}$
		f = 3 *if* $\theta < 0$ and $\phi' \leftarrow \frac{\pi}{4}$

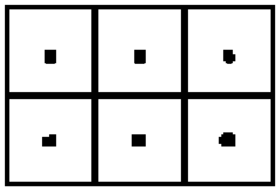
In Table 10, the value of T is a pre-determined constant set to 0.59. $\lVert (X, Y, Z) - P \rVert$ computes the squared distance between (X, Y, Z) and P. Denote P as (X', Y', Z'), $\lVert (X, Y, Z) - P \rVert = \lVert (X, Y, Z) - (X', Y', Z') \rVert = (X - X')^2 + (Y - Y')^2 + (Z - Z')^2$. The value of ϕ' is derived from (ϕ, θ) according to (61).

2.10 Equatorial cylindrical projection format (ECP)

The ECP format partitions the sphere into three regions, the equatorial region, the North pole and the South pole regions. ECP applies Lambert cylindrical equal-area projection to the equatorial sphere region as illustrated in Figure 15 (a). The latitude interval of the equatorial region is $\pm\sin^{-1}(2/3)\approx\pm41.81°$, which is chosen such that the equatorial region is 2/3 of the total sphere area, and each pole area is 1/6 of the sphere area. The pole discs are mapped into square faces as shown in Figure 15 (b). ECP uses the frame packing arrangement as shown in Figure 15 (c). Figure 15 (d) shows the corresponding faces in the ERP format. The pole faces are numbered 0 and 1, while the equatorial faces are numbered 2 to 5. In the frame packed image, the top row consists of the first three equatorial faces (face indices 2-4) and the bottom row contains the rotated 5th equatorial face in between the rotated square pole faces.


(a) Sphere partitions


(b) mapping pole discs to square faces


(c) ECP frame packing

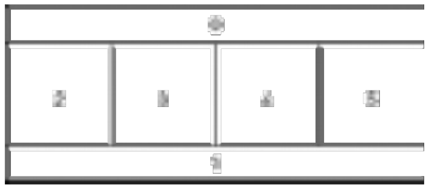

(d) corresponding faces in ERP

Figure 15. Partitions of the sphere and frame packing in ECP

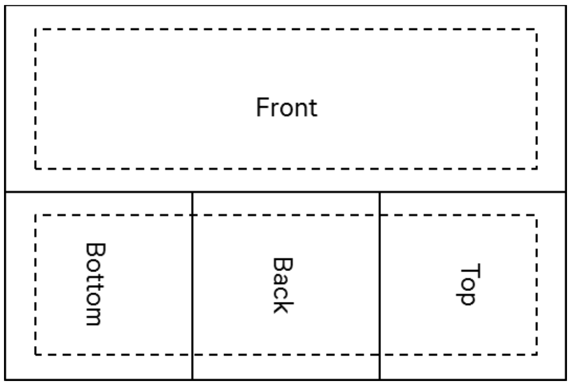


Figure 16. Padding in ECP

Padding is applied to the ECP projection as shown in Figure 16, where the regions inside the dashed boxes contain active samples, and the regions outside the dashed boxes contain the padded samples. Similar to padding applied to other projection formats, padding in ECP reduces visual artifacts such as seams in the rendered viewport. In 360Lib, the ECP padding size is set to 4 luma samples between a dashed line and its respective neighboring solid line. The padded samples are obtained using the ECP projection process using the neighboring active samples.

Denote the dimension of one face as $A\times A$. In 2D-to-3D conversion, a point (m,n) on the face is mapped to a point on the sphere (ϕ,θ) according to equations below, depending on the face to which it belongs (Figure 15 (c)).

First, the point (x,y) is determined as follows:

$$x=\frac{2}{A}\left(m+\frac{1}{2}\right)-1 \qquad (62)$$

$$y=\frac{2}{A}\left(n+\frac{1}{2}\right)-1 \qquad (63)$$

Two scaling factors are defined as follows based on the padding size:

$$p\,adfactor=1+\frac{2*padmargin}{A-2*padmargin} \qquad (64)$$

$$apadfactor=\frac{A}{A-padmargin} \qquad (65)$$

with *padmargin* specifying the padding width in luma sample units, which is set to 4 in the 360Lib software. Then, depending on the face index f, scaling as specified in Table 11 is applied to (x, y) :

Table 11. Scaling of (x, y) for face *f (f = 0 … 5)* in 2D-to-3D conversion in ECP

Index *f*	**2D-to-3D Conversion**
0	$x^{'}=x*padfactor$ $y^{'}=(y+1)*apadfactor-1$
1	$x^{'}=x*padfactor$ $y^{'}=(y-1)*apadfactor+1$
2	$x^{'}=(x-1)*apadfactor+1$ $y^{'}=y*padfactor$
3	$x'=x$ $y^{'}=y*padfactor$
4	$x^{'}=(x+1)*apadfactor-1$ $y^{'}=y*padfactor$
5	$x^{'}=x*padfactor$ $y'=y$

If face index *f* = 0 or *f* = 1, the gradual sampling transitioning is applied as follows:

$$x^{''}=\begin{cases}1+\tanh\left(\frac{-y'-1}{b}\right), if\,f=0\\ 1+\tanh\left(\frac{y'-1}{b}\right), if\,f=1\end{cases} \qquad (66)$$

$(x^{''})\cdot x'$

$\tan^{-1}¿/x''$ (67)

$¿$

$¿$

$x=\tan ¿$

with $b=0.2$ and $y=y'$.

If face index *f* = 0 or 1, square-to-disc mapping [27] is applied:

$$s=\frac{\sqrt{x^2+y^2-x^2y^2}}{\sqrt{x^2+y^2}} \tag{68}$$

$$u=\frac{A}{2}\cdot s\cdot x \tag{69}$$

$$v=\frac{A}{2}\cdot s\cdot y \tag{70}$$

For the face *f* = *0*, the following applies:

$$\phi=\tan^{-1}\left(\frac{u}{v}\right)\frac{\pi}{4} \tag{71}$$

$$\theta=\frac{\pi}{2}-\frac{2r}{A}\left(\frac{\pi}{2}-\sin^{-1}\left(\frac{2}{3}\right)\right) \tag{72}$$

where $r=\sqrt{u^2+v^2}$, $\theta\in\left(\sin^{-1}\left(\frac{2}{3}\right),\frac{\pi}{2}\right]$, $\phi\in(-\pi,\pi]$.

For the face *f* = *1*, the following applies:

$$\phi=\tan^{-1}\left(\frac{u}{v}\right)+\frac{\pi}{4} \tag{73}$$

$$\theta=\frac{2r}{A}\left(\frac{\pi}{2}-\sin^{-1}\left(\frac{2}{3}\right)\right)-\frac{\pi}{2} \tag{74}$$

where $r=\sqrt{u^2+v^2}$ and $\theta\in\left[\frac{-\pi}{2},-\sin^{-1}\left(\frac{2}{3}\right)\right)$, $\phi\in(-\pi,\pi]$.

For the square faces with face index *f, f = 2…5,* (u,v) is set to (x',y') after scaling is applied according to Table 11. Then, the following applies:

$$\phi=\frac{\pi}{4}\cdot(u+2f-7) \tag{75}$$

$$\theta=\sin^{-1}(-2v/3) \tag{76}$$

with yaw in the range $\phi\in\left(-\pi+(f-2)\frac{\pi}{2},-\frac{\pi}{2}+(f-2)\frac{\pi}{2}\right]$ depending on f, and pitch in the range $\theta\in\left[-\sin^{-1}\left(\frac{2}{3}\right),\sin^{-1}\left(\frac{2}{3}\right)\right]$.

For 3D-to-2D conversion, a point on the sphere (ϕ,θ) is mapped to a point (m,n) on the face with dimension $A\times A$ depending on the face index f as described below.

For the face *f* = *0*, $\theta\in\left(\sin^{-1}\left(\frac{2}{3}\right),\frac{\pi}{2}\right]$, $\phi\in(-\pi,\pi]$, the following applies:

$$u = \frac{\left(\frac{\pi}{2} - \theta\right) \sin\left(\phi + \frac{\pi}{4}\right)}{\left(\frac{\pi}{2} - \sin^{-1}\left(\frac{2}{3}\right)\right)} \tag{77}$$

$$v = \frac{\left(\frac{\pi}{2} - \theta\right) \cos\left(\phi + \frac{\pi}{4}\right)}{\left(\frac{\pi}{2} - \sin^{-1}\left(\frac{2}{3}\right)\right)} \tag{78}$$

Followed by disc-to-square mapping [27]:

$$(x', y') = \begin{cases} \left(\frac{w}{v}, \frac{w}{u}\right) if\, |w| > 0 \\ (u, v), otherwise \end{cases} \tag{79}$$

Where $w = \frac{sgn(uv)}{\sqrt{2}} \sqrt{u^2 + v^2 - \sqrt{(u^2+v^2)(u^2+v^2-4u^2v^2)}}$ with *sgn()* being the sign function. Then, the gradual sampling transitioning is applied:

$$x'' = 1 + \tanh\left(\frac{-y' - 1}{b}\right) \tag{80}$$

$$x = \tan^{-1}(x'' x') / \tan^{-1}(x'') \tag{81}$$

with $b = 0.2$ and $y = y'$.

For the face *f = 1*, $\theta \in \left[\frac{-\pi}{2}, -\sin^{-1}\left(\frac{2}{3}\right)\right)$, $\phi \in (-\pi, \pi]$, the following applies:

$$u = \frac{\left(\frac{\pi}{2} + \theta\right) \sin\left(\phi + \frac{\pi}{4}\right)}{\left(\frac{\pi}{2} - \sin^{-1}\left(\frac{2}{3}\right)\right)} \tag{82}$$

$$v = \frac{-\left(\frac{\pi}{2} + \theta\right) \cos\left(\phi + \frac{\pi}{4}\right)}{\left(\frac{\pi}{2} - \sin^{-1}\left(\frac{2}{3}\right)\right)} \tag{83}$$

Followed by disc-to-square mapping:

$$(x', y') = \begin{cases} \left(\frac{w}{v}, \frac{w}{u}\right) if\, |w| > 0 \\ (u, v), otherwise \end{cases} \tag{84}$$

Where $w = \frac{sgn(uv)}{\sqrt{2}} \sqrt{u^2 + v^2 - \sqrt{(u^2+v^2)(u^2+v^2-4u^2v^2)}}$.

Then, the gradual sampling transitioning is applied:

$$x'' = 1 + \tanh\left(\frac{y'-1}{b}\right) \quad (85)$$

$$x = \tan^{-1}(x'' x') / \tan^{-1}(x'') \quad (86)$$

with $b = 0.2$ and $y = y'$.

For the square faces with face index *f* = 2…5, yaw in the range $\phi \in \left(-\pi + (f-2)\frac{\pi}{2}, -\frac{\pi}{2} + (f-2)\frac{\pi}{2}\right]$ depending on f, and pitch in the range $\theta \in \left[-\sin^{-1}\left(\frac{2}{3}\right), \sin^{-1}\left(\frac{2}{3}\right)\right]$, the following applies:

$$x = \frac{4}{\pi}\phi + 7 - 2f \quad (87)$$

$$y = \frac{-3}{2}\sin(\theta) \quad (88)$$

Depending on the face index *f*, scaling as given in Table 12 is applied to (x, y) to derive (x', y') , where *padfactor* and *apadfactor* are derived according to Equations (64) and (65):

Table 12. Scaling of (x, y) for face *f (f = 0 … 5)* in 3D-to-2D conversion in ECP

Index *f*	**3D-to-2D Conversion**
0	$x' = x / padfactor$ $y' = (y+1)/apadfactor - 1$
1	$x' = x / padfactor$ $y' = (y-1)/apadfactor + 1$
2	$x' = (x-1)/apadfactor + 1$ $y' = y / padfactor$
3	$x' = x$ $y' = y / padfactor$
4	$x' = (x+1)/apadfactor - 1$ $y' = y / padfactor$
5	$x' = x / padfactor$ $y' = y$

Finally, (m, n) are given by:

$$m = \frac{A}{2}(x'+1) - \frac{1}{2} \quad (89)$$

$$n = \frac{A}{2}(y'+1) - \frac{1}{2} \quad (90)$$

2.11 *Truncated square pyramid projection format (TSP)*

360Lib includes the truncated square pyramid (TSP) projection, which utilizes the cube geometry and warps the six cube faces into a compact frame. Figure 17 defines the (x, y) coordinates inside the packed TSP frame. (x, y) have normalized ranges (0.0, 1.0). Given the sampling position (m, n) on cube face f, (x, y) are calculated with following equations:

$$x = 0.5*(m + 0.5)/W + 0.5,\ 0 \leq m < W \tag{91}$$

$$y = (n + 0.5)/H,\ 0 \leq n < H \tag{92}$$

In this implementation, the front TSP face corresponds with the front cube face, the back cube face is subsampled by 4 horizontally and vertically, while the side cube faces are warped to trapezoidal regions. Table 13 specifies the forward and inverse mapping equations. The coordinates (u, v) can be obtained from (x′, y′) as follows:

$$u = 2x' - 1.0 \tag{93}$$

$$v = 2y' - 1.0 \tag{94}$$

where (x’, y’) is defined in Table 13, and (x′, y′) are in the range (0.0, 1.0). Subsequently, (X, Y, Z) are derived with Table 3 given the position (u, v) on cube face f.

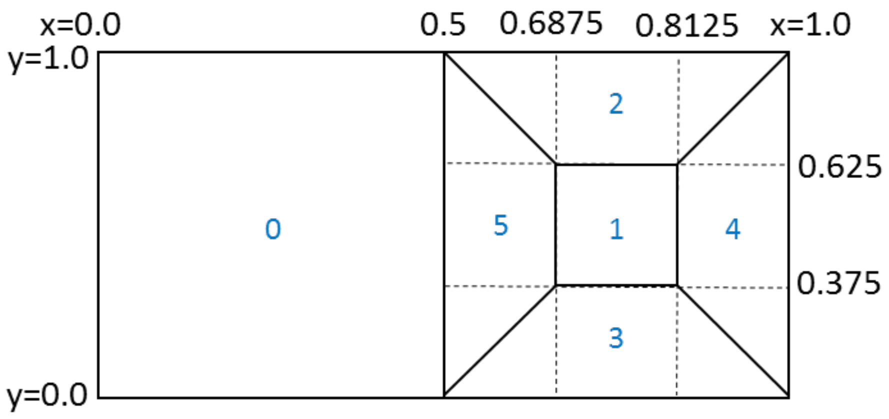


Figure 17. Truncated-square-pyramid frame packing format

Table 13. Forward and inverse equations between cube faces and TSP faces

Forward equations (TSP to cube faces)	**Inverse equations (cube faces to TSP)**
Right TSP trapezoid from right cube face: $x' = (x - 0.5) / 0.1875$ $y' = (y - 2.0x + 1.0) / (3.0 - 4.0x)$	Right cube face from right TSP trapezoid: $x = 0.1875x' + 0.5$ $y = 0.375x' - 0.75x'y' + y'$
Left TSP trapezoid from left cube face: $x' = (x - 0.8125) / 0.1875$ $y' = (y + 2.0x - 2.0) / (4.0x - 3.0)$	Left cube face from left TSP trapezoid: $x = 0.1875x' + 0.8125$ $y = 0.25y' + 0.75x'y' - 0.375x' + 0.375$
Bottom TSP trapezoid from bottom cube face: $x' = (1.0 - x - 0.5y) / (0.5 - y)$ $y' = (0.375 - y) / 0.375$	Bottom cube face from bottom TSP trapezoid: $x = 0.1875y' - 0.375x'y' - 0.125x' + 0.8125$ $y = 0.375 - 0.375y'$

Top TSP trapezoid from top cube face: $x' = (0.5 - x + 0.5y) / (y - 0.5)$ $y' = (1.0 - y) / 0.375$	Top cube face from top TSP trapezoid: $x = 1.0 - 0.1875y' - 0.5x' + 0.375x'y'$ $y = 1.0 - 0.375y'$
Back TSP face from back cube face: $x' = (x - 0.6875) / 0.125$ $y' = (y - 0.375) / 0.25$	Back cube face from back TSP face: $x = 0.125x' + 0.6875$ $y = 0.25y' + 0.375$

2.12 Viewport generation with rectilinear projection

In 360Lib, a viewport is generated by rectilinear projection, as illustrated in Figure 18. Viewport generation in 360Lib is performed assuming that the viewing angle is along the Z axis. When the viewport specified by the user is not along the Z axis, the sphere is rotated first to align the viewport with the Z axis, and then viewport generation is performed. Denote the center of the viewport to be at (ϕ_C, θ_C), the rotation matrix R is defined as:

$$R = \begin{bmatrix} \cos(\phi_C + \pi/2) & -\sin(\phi_C + \pi/2)\sin(\theta_C) & \sin(\phi_C + \pi/2)\cos \\ 0 & \cos(\theta_C) & \sin(\theta_C) \\ -\sin(\phi_C + \pi/2) & -\cos(\phi_C + \pi/2)\sin(\theta_C) & \cos(\phi_C + \pi/2)\cos \end{bmatrix} \quad (95)$$

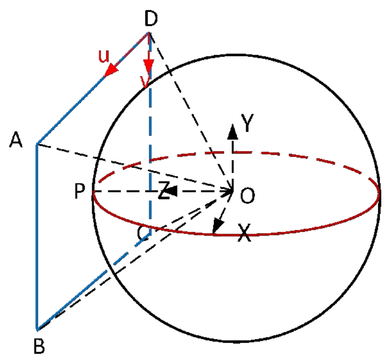


Figure 18. Viewport generation with rectilinear projection

Viewport generation starts from a sample position on the projected viewport, first finds the corresponding 3D (X, Y, Z) coordinates, then finds the corresponding 2D coordinates in the source projection plane, and finally takes the corresponding sample value at the corresponding position on the source 2D projection plane. In this section, we only describe the first step, that is, how to map a sample position to the 3D (X, Y, Z) coordinates based on rectilinear projection. The remaining steps are the same as the projection format conversion process to be described in the Section 3.

Denote the viewport picture ABCD's width as W_{VP} and its height as H_{VP}. Denote the FOV size of the viewport as (F_h x F_v), where F_h is the horizontal FOV angle and F_v is the vertical FOV angle. Given a sampling point (m, n) in the viewport picture ABCD shown in Figure 18, the (u, v) coordinates are calculated as:

$$u = (m+0.5)*2*\tan(F_h/2)/W_{VP} \quad (96)$$

$$v = (n+0.5)*2*\tan(F_v/2)/H_{VP} \quad (97)$$

Then, the 3D coordinates (x, y, z) are calculated as:

$$x = u - \tan(F_h/2) \quad (98)$$

$$y = -v + \tan(F_v/2) \quad (99)$$

$$z = 1.0 \tag{100}$$

Projecting the point (x, y, z) onto the point (x′, y′, z′) on the unit sphere, we have:

$$x' = x/ \quad x^2+y^2+z^2 \; ¿ \; \sqrt{¿} \tag{101}$$

$$y' = y/ \quad x^2+y^2+z^2 \; ¿ \; \sqrt{¿} \tag{102}$$

$$z' = 1.0 / \quad x^2+y^2+z^2 \; ¿ \; \sqrt{¿} \tag{103}$$

Finally, taking the rotation R into account, the 3D coordinates (X, Y, Z) on the sphere are calculated as:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = R \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix}$$

3 Conversion between two projection formats

In 360Lib, the picture of a given projection format is stored face by face. The conversion from the source projection format to the destination projection format is also performed face by face.

Denote (f_d, m_d, n_d) as a point (m_d, n_d) on face f_d in the destination projection format, and (f_s, m_s, n_s) as a point (m_s, n_s) on face f_s in the source projection format. Denote (X, Y, Z) as the corresponding coordinates in the 3D XYZ space. The conversion process starts from each sample position (f_d, m_d, n_d) on the destination projection plane, maps it to the corresponding (X, Y, Z) in 3D coordinate system, finds the corresponding sample position (f_s, m_s, n_s) on the source projection plane, and sets the sample value at (f_d, m_d, n_d) based on the sample value at (f_s, m_s, n_s). Additionally, instead of mandating the XYZ axes of the source and destination projections to be aligned, 360Lib allows the user to use a set of 3D rotation parameters (yaw, pitch, roll) in the configuration file to specify the relative rotation between the source and destination 3D coordinates, where yaw specifies the counterclockwise rotation in degrees along the Y axis, pitch specifies the counterclockwise rotation in degrees along −Z axis, and roll specifies the counterclockwise rotation in degrees along the X axis in the 360Lib coordinate system (Figure 1).

The projection format conversion process from source format to destination format is performed in the following steps:

(Step 1) Map the destination 2D sampling point (f_d, m_d, n_d) to 3D space coordinates (X, Y, Z) based on the destination projection format;

(Step 2) If needed, rotate the 3D point (X, Y, Z) along the three axes according to the three rotation angles (yaw, pitch, roll) to (X′, Y′, Z′) using the rotation matrix R_{XYZ} defined in Equation (104):

$$R_{XYZ} = R_Y(yaw) \cdot R_Z(-pitch) \cdot R_X(roll) \tag{104}$$

Where the three rotation matrices around the Z, Y, and X axes, R_Z , R_Y , R_X respectively, are defined as follows:

$$R_Z(pitch) = \begin{bmatrix} \cos \boxed{f0} (pitch) & -\sin \boxed{f0} (pitch) & 0 \\ \sin \boxed{f0} (pitch) & \cos \boxed{f0} (pitch) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_Y(yaw)=\begin{bmatrix} \cos\boxed{f0}\,(yaw) & 0 & \sin\boxed{f0}\,(yaw) \\ 0 & 1 & 0 \\ -\sin\boxed{f0}\,(yaw) & 0 & \cos\boxed{f0}\,(yaw) \end{bmatrix}$$

$$R_X(roll)=\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\boxed{f0}\,(roll) & -\sin\boxed{f0}\,(roll) \\ 0 & \sin\boxed{f0}\,(roll) & \cos\boxed{f0}\,(roll) \end{bmatrix}$$

(Step 3) Map (X′, Y′, Z′) to 2D sampling point (f_s, m_s, n_s) based to the source projection format;

(Step 4) Calculate the sample value at (f_s, m_s, n_s) by interpolating from neighboring samples at integer positions on face f_s, and the interpolated sample value is placed at (f_d, m_d, n_d) in the destination projection format.

The above steps are repeated until all sample positions (f_d, m_d, n_d) in the destination projection format are filled. In 360Lib software, (Step 1) is performed using the function map2Dto3D() defined in the class of destination projection geometry, and (Step 3) is performed using function map3Dto2D() defined in the class of source projection geometry. (Step 2) may be skipped, if the user has not specified non-zero (yaw, pitch, roll) parameters. Note that (Step 1), (Step 2) and (Step 3) can be pre-calculated at the sequence level and stored as a lookup table, and only (Step 4) needs to be performed per sample position for each picture in order to render the sample values.

If rotation in (Step 2) has been applied, then an inverse rotation is applied using the inverse rotation matrix R'_{XYZ} defined below, before all end-to-end quality metrics (see Section 4) are calculated:

$$R'_{XYZ}=R_X(-roll)\cdot R_Z(pitch)\cdot R_Y(-yaw) \qquad (105)$$

3.1 Interpolation filters

When the source sample position (f_s, m_s, n_s) is at factional sample position, interpolation needs to be performed to calculate the sample values at (f_s, m_s, n_s) from the neighboring sample values at integer positions. The set of interpolation filters supported by 360Lib is listed in Table 14. All of the interpolation filters are specified at 1/100-th sample precision in 360Lib.

For projection format conversion, the user can specify interpolation filters for the luma and chroma components separately. By default, Lanczos-3 is used for luma and Lanczos-2 is used for chorma in projection format conversion. For viewport generation, bilinear filters are used. When 360Lib is used in combination with the HM or JEM codec, only bilinear filters are supported for viewport generation. When 360Lib is used as a stand-alone application, the filters used for viewport generation are configurable.

Table 14. Interpolation methods supported

Index	Interpolation method	Notes
0	Reserved	
1	Nearest neighbor	
2	Bilinear	Used for viewport generation *
3	Bicubic	
4	Lanczos2	Default filter for the chroma components
5	Lanczos3	Default filter for the luma component

Note - Viewport generation inside encoder supports only bilinear filters, whereas viewport generation in standalone 360Lib application supports all filters listed.

3.2 Chroma format support

360Lib allows the user to specify an internal chroma format, in which the projection format conversion process is performed. The internal chroma format can be different from the input and/or output chroma formats. For example, if the input video is in 4:2:0 chroma format, the user may specify the internal chroma format to be 4:2:0 or 4:4:4. When the internal chroma format is not the same as the input chroma format, chroma upsampling or downsampling is performed on the input video before projection format conversion is performed. When the internal chroma format is not the same as the output chroma format, chroma upsampling or downsampling is performed after projection format conversion. The HEVC motion compensation filters are used to perform chroma upsampling or downsampling.

When the internal chroma format is in 4:4:4, the luma and chroma components have the same sampling grid. Therefore, the projection format conversion process in Section 3 can be directly applied to luma and chroma, as the luma and chroma coordinates will remain the same before and after projection conversion, maintaining the 4:4:4 chroma format without any issue.

However, when the internal chroma format is not 4:4:4, the luma and the chroma sampling grids are not alway exactly aligned. For example, it is well known that in chroma format 4:2:0, by default the chroma sample location is shifted by a ½ luma sample compared to the corresponding luma location in the vertical direction. Because the projection format conversion process is non-linear, care needs to be given in order to ensure that luma and chroma sampling grids are properly aligned after projection format conversion. In other words, for default 4:2:0 chroma location type, the ½ luma sample shift in the vertical direction needs to be maintained after projection format conversion. 360Lib uses a simple method to deal with this issue: because the luma and chroma sampling grids are aligned (except for a factor of 2 in terms of resolution) in the chroma sample location type 2, 360Lib uses the type-2 chroma sample location when internal chroma is set to 4:2:0. Then, when performing projection format conversion for the chroma components, the chroma sampling grid is first scaled up by a factor of 2 to align with the luma sampling grid before conversion, and is scaled down by a factor of 2 after conversion.

4 Spherical objective quality metrics

Four spherical quality metrics are implemented in 360Lib for 360 video quality evaluation: weighted to spherically uniform PSNR (WS-PSNR) [13], spherical PSNR without interpolation (S-PSNR-NN), spherical PSNR with interpolation (S-PSNR-I), PSNR in Crasters Parabolic Projection format (CPP-PSNR). S-PSNR was originally proposed in [9]. In order to evaluate viewport quality, viewport based PSNR is also supported in 360Lib.

Table 15. Quality metrics supported in 360Lib

Metric name	Notes
PSNR	Conventional PSNR calculation with equal weight for all samples
Weighted to Spherically uniform PSNR (WS-PSNR)	The distortion at each sample position is weighted by the area on the sphere covered by the given sample position. All samples on the 2D projection plane are used in WS-PSNR calculation. The two inputs to the metric calculation must have the same resolution and projection format.
Spherical PSNR w/o interpolation (S-PSNR-NN)	Calculate PSNR based on a set of points uniformly sampled on the sphere. To find the sample value at the corresponding position on the projection plane, nearest neighbor rounding is applied. The two inputs to the metric calculation can have different resolution and/or projection format.
Spherical PSNR with interpolation (S-PSNR-I)	Calculate PSNR based on a set of points uniformly sampled on the sphere. To find the sample value at the corresponding position on the projection plane, bicubic interpolation is applied. The two inputs to the metric calculation can have different resolution and/or projection format.

CPP-PSNR	Apply another projection format conversion to convert the two inputs into the Crasters Parabolic Projection (CPP) domain, and calculate PSNR in CPP domain. The two inputs to the metric calculation can have different resolution and/or projection format.

The JVET has established common test conditions for 360-degree video [2]. Figure 19 depicts the processing and coding pipeline. For a given input 360-degree video, projection format conversion is applied first to convert source projection format into coding projection format. In the CTC for 360-degree video, the coding projection format is in lower resolution than that of the source projection format. For example, 8K source video is coded in 4K resolution. After coding, the reconstructed signal in coding projection format is back converted to the source projection format at the source resolution. In the CTC, original video sequences are all provided in the ERP format in either 8K or 4K.

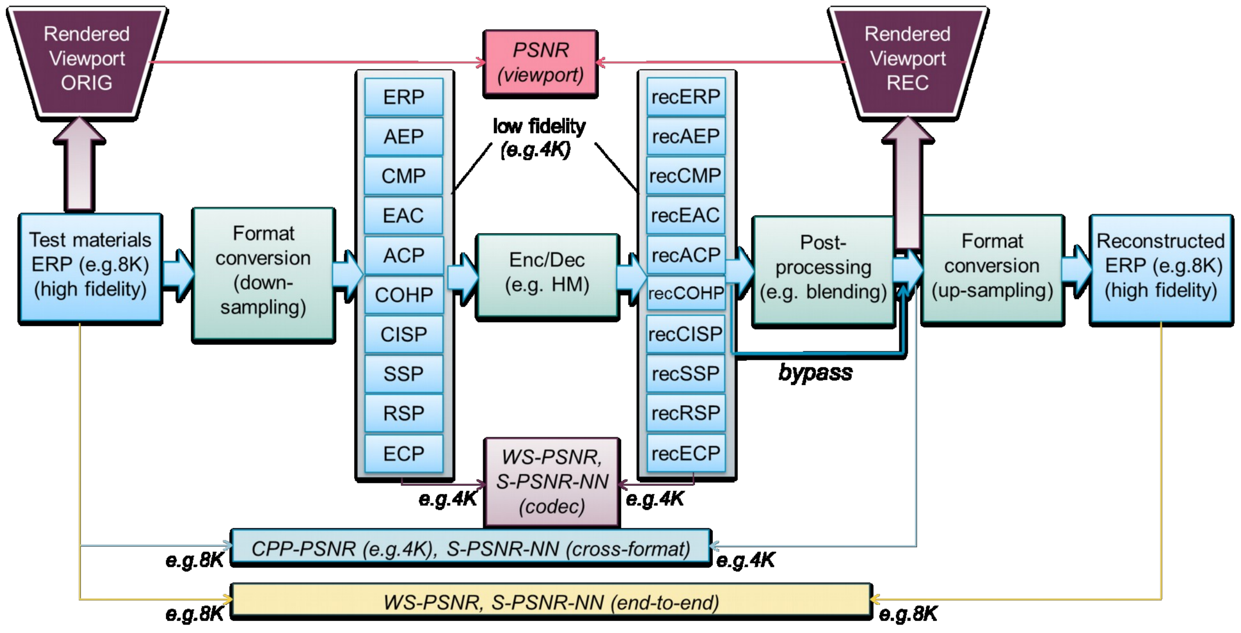

Rendered Viewport ORIG
PSNR (viewport)
Rendered Viewport REC
ERP
AEP
CMP
EAC
ACP
COHP
CISP
SSP
RSP
ECP
low fidelity (e.g.4K)
recERP
recAEP
recCMP
recEAC
recACP
recCOHP
recCISP
recSSP
recRSP
recECP
Test materials ERP (e.g.8K) (high fidelity)
Format conversion (down-sampling)
Enc/Dec (e.g. HM)
Post-processing (e.g. blending)
Format conversion (up-sampling)
Reconstructed ERP (e.g.8K) (high fidelity)
bypass
WS-PSNR, S-PSNR-NN (codec)
e.g.4K
e.g.4K
CPP-PSNR (e.g.4K), S-PSNR-NN (cross-format)
e.g.8K
e.g.4K
WS-PSNR, S-PSNR-NN (end-to-end)
e.g.8K
e.g.8K

Figure 19. 360 video testing procedure

Calculation of 360-video objective quality metrics is performed at different stages in the CTC coding pipeline, summarized as the following 3 categories:

1. End-to-end distortion measurement: WS-PSNR and S-PSNR-NN are calculated between the original signal in source projection format and the reconstructed signal in source projection format. The end-to-end distortion considers both projection format conversion errors (including forward and backward projection format conversion) and coding errors;
2. Cross-format distortion measurement: CPP-PSNR and S-PSNR-NN are measured between the original signal in source projection format and the reconstructed signal in coding projection format (hence the name “cross-format”). Partial (only forward) projection format conversion errors and coding errors are measured;

3. Coding distortion measurement: WS-PSNR and S-PSNR-NN are measured between the input to the codec and the output of the codec. Only coding errors are measured, and projection format conversion errors are not measured.

Additionally, the CTC includes viewport quality evaluation using viewports generated from the original signal in the source projection format and the reconstructed signal in the coding projection format. For more details on the CTC, readers are referred to [2].
In 360 CTC, ERP with padding is designated as the anchor projection format. By default, blending is used as the post-processing step in Figure 19 for ERP with padding. The reconstructed ERP with blended samples (i.e., recERP described in 2.1) will be used for end-to-end distortion measurement, cross-format distortion measurement, and viewport distortion measurement. When measuring coding distortion, the active samples in padded ERP (i.e., recPERP described in 2.1) are used.

4.1 S-PSNR

S-PSNR was originally proposed in [9]. It uses a set of uniformly sampled positions on the sphere to measure 360-degree video quality. S-PSNR is calculated using the following steps:

1. For a point "s" on the sphere, apply 3D-to-2D coordinate mapping to find the corresponding positions "r" and "t" in the two inputs to metric calculation (that is, reference signal and test signal in Figure 20);
2. Calculate the distortion between the samples values at "r" and "t";

3. Calculate the overall distortion by accumulating the distortion at each "s" in the set of points, and calculate PSNR from distortion

360Lib uses the same set of 655,362 points as provided in [9].

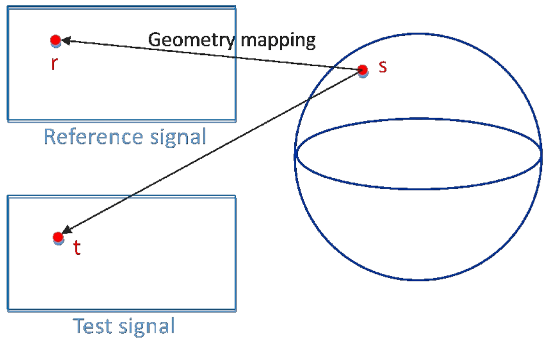


Figure 20. S-PSNR calculation

Two variants of the S-PSNR metric are supported in 360Lib. The first one uses nearest neighbor rounding when "r" and/or "t" are at fractional sample positions in order to avoid introducing impact due to interpolation filters when calculating distortion. It is referred to as S-PSNR-NN. The second variant uses interpolation filters to calculate sample values at fractional positions. It is referred to as S-PSNR-I. At the 7th JVET meeting, it was agreed to only report S-PSNR-NN in the 360 video CTC [26].

4.1.1 Cross-format S-PSNR-NN

As depicted in Figure 19, in 360-degree video CTC [2], the cross-format quality metrics S-PSNR-NN, S-PSNR-I, and CPP-PSNR measure the quality difference between the original 360-degree video and the reconstructed video output from the decoder, which may be in a projection format and/or resolution that is different from the original video. For S-PSNR-NN, special care needs to be taken to ensure that the sampling points in the original and the reconstruction video between which errors are calculated always correspond to the same spherical point after nearest rounding of sampling position is applied.

The cross-format S-PSNR-NN in 360Lib-3.0 is calculated following the steps below (as shown in Figure 21):

1) Map point S (S is from the set of 655,362 point) to point g in the original video projection plane;

2) Round point g to nearest neighbor point g’ at integer sampling position in the original video projection plane;

3) Perform inverse coordinate mapping of the point g’ back onto the sphere at S(g’);

4) Perform coordinate mapping from the spherical coordinate S(g’) to the reconstructed video projection plane at the position r;

5) Apply interpolation to derive the sample value at point r using its neighboring sample values at integer sampling positions (if r is not located at integer sampling position);

6) Calculate error between the sample value at point g’ in the original video and the interpolated sample value at point r in the reconstructed video.

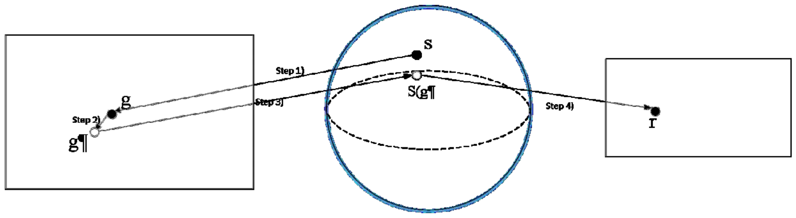


Figure 21. Cross-format S-PSNR-NN calculation

This cross-format S-PSNR-NN calculation method ensures that interpolation is not applied to the original video; therefore, the characteristics of the original video are not changed during metric calculation.

4.2 *WS-PSNR*

Another quality metric supported in 360Lib is the Weighted to Spherically uniform PSNR (WS-PSNR) [14]. WS-PSNR calculates PSNR using all image samples on the 2D projection plane. The distortion at each position (i,j) is weighted by the spherical area covered by that sample position. For each position (i,j) of an $M \times N$ image on the 2D projection plane, denote the sample values on the reference and test images as $y(i,j)$ and $y'(i,j)$, respectively, and denote the spherical area covered by the sample as $w(i,j)$. The weighted mean squared error (WMSE) is first calculated as:

$$WMSE = \frac{1}{\sum_{i=0}^{M-1}\sum_{j=0}^{N-1} w(i,j)} \sum_{i=0}^{M-1}\sum_{j=0}^{N-1} \left(y(i,j) - y'(i,j)\right)^2 * w(i,j) \tag{106}$$

The WS-PSNR is then calculated as:

$$WS_{PSNR} = 10\log\left(\frac{MAX_I^{\,2}}{WMSE}\right) \tag{107}$$

where MAX_I is the maximum intensity level of the images.

Since WS-PSNR is calculated based on the projection plane, different weights are derived for different projection formats. The WS-PSNR metric implementation in 360Lib supports all projection formats except TSP. Weight derivation for each projection format is discussed below. Figure 22 to Figure 29 illustrate the weight distributions for different projection formats using weight maps, where locations with larger weights are shown with higher intensity levels (i.e., lighter shades of gray).

4.2.1 WS-PSNR calculation for ERP with optional padding

For an $M \times N$ image in the ERP format, the weight $w(i,j)$ at position (i,j) is calculated as:

$$w(i,j)_{ERP} = \cos \boxed{f()} \frac{(j+0.5-N/2)\pi}{N} \tag{108}$$

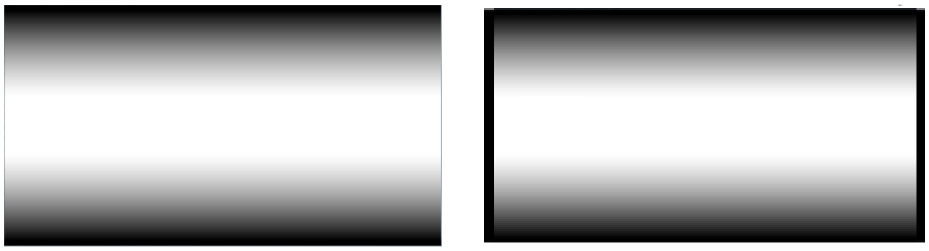

(a) (b)

Figure 22. Weight map for ERP: (a) without padding; (b) with padding

The ERP with padding contains padded samples at the left and right picture boundaries. For these positions, the WS-PSNR weights are set to 0, as shown in the dark areas in Figure 22 (b).

4.2.2 WS-PSNR calculation for CMP

For an image in CMP format, weight distributions on all faces are the same. Therefore, only weights in one face are derived. For position (i,j) on a CMP face with resolution $A \times A$, the weight $w(i,j)$ is calculated as:

$$w(i,j) = \left(1 + \frac{d^2(i,j)}{r^2}\right)^{-3/2} \tag{109}$$

where $r = \frac{A}{2}$ is the radius, and $d^2(i,j) = (i+0.5-A/2)^2 + (j+0.5-A/2)^2$ is the squared distance between (i,j) and the center of the face. Figure 23 shows the weight map for CMP with the faces arranged in the compact CMP3x2 manner.

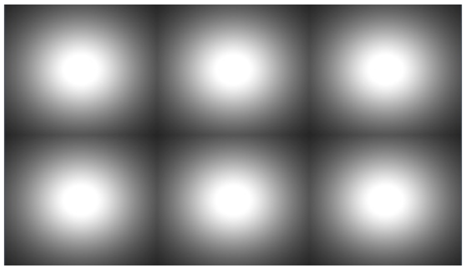


Figure 23. Weight map for CMP 3x2

4.2.3 WS-PSNR calculation for ACP

For an image in ACP format, weight distributions on all faces are the same. Therefore, only weights in one face are derived. Because the ACP format is derived using the CMP format as an intermediate step, the weights for ACP are also derived based on the weights for CMP [24]. For position (i,j) on an ACP face with resolution $A \times A$, the weight $w(i,j)$ is calculated as follows:

$$w(i,j) = \frac{1}{\left\{1 + 4 \cdot \left[\left(\left(i_{cmp} + \frac{1}{2} - \frac{A}{2}\right)/A\right)^2 + \left(\left(j_{cmp} + \frac{1}{2} - \frac{A}{2}\right)/A\right)^2\right]\right\}^{\frac{3}{2}} \cdot 16 \cdot t_i \cdot t_j} \tag{110}$$

where t_i and t_j are derived from (i,j) as follows:

$$t_i = \sqrt{0.34^2 - 0.09 * \left| 2(i + 0.5)/A - 1 \right|} \tag{111}$$

$$t_j=\sqrt{0.34^2-0.09*|2(j+0.5)/A-1|} \quad (112)$$

$$i_{cmp}=0.5\cdot A\cdot(1.0+(0.34-\sqrt{0.34^2-0.09*|2(i+0.5)/A-1|})/0.18) \quad (113)$$

$$j_{cmp}=0.5\cdot A\cdot(1.0+(0.34-\sqrt{0.34^2-0.09*|2(j+0.5)/A-1|})/0.18) \quad (114)$$

Error: Reference source not foundFigure 24 shows the weight map for ACP with the faces arranged in the compact ACP3x2 manner.

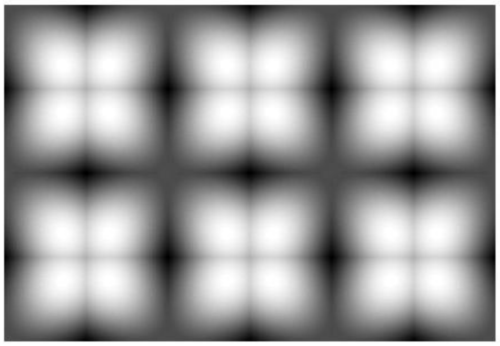


Figure 24. Weight map for ACP 3x2

4.2.4 WS-PSNR calculation for EAC

Similar to CMP and ACP, for the EAC format, weight distributions on all faces are the same. Therefore, only weights in one face are derived. For position (i,j) on an EAC face with resolution $A\times A$, the weight $w(i,j)$ is calculated as follows:

$$w(i,j)=\frac{\pi^2}{\left\{1+(\tan \boxed{f0}\,(t_i))^2+(\tan \boxed{f0}\,(t_j))^2\right\}^{\frac{3}{2}}\cdot 16\cdot(\cos \boxed{f0}\,(t_i))^2\cdot(\cos \boxed{f0}\,(} \quad (115)$$

where t_i and t_j are derived from (i,j) as follows:

$$t_i=\frac{\pi}{4}\cdot\left(\frac{2(i+0.5)}{A}-1\right) \quad (116)$$

$$t_j=\frac{\pi}{4}\cdot\left(\frac{2(j+0.5)}{A}-1\right) \quad (117)$$

Figure 25 shows the weight map for EAC with the faces arranged in the compact 3x2 layout format.

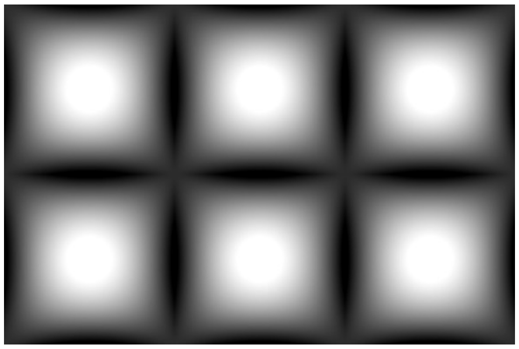


Figure 25. Weight map for EAC 3x2

4.2.5 WS-PSNR calculation for AEP

In the conventional equal-area projection format, weights are the same for all positions on an $M \times N$ image. In this case, WS-PSNR is the same as the conventional PSNR.

For the AEP format, the WS-PSNR weight $w(i,j)$ at sample position (i, j) on an $M \times N$ image is calculated as:

$$w(i,j) = \frac{4\pi * C * \cos\left(\frac{1}{\beta} * \sin^{-1}(C * (1-(2j+1)/N))\right)}{\beta * \sqrt{1-(C*(1-(2j+1)/N))^2}} \qquad (118)$$

Where C is a constant and $C = \sin(0.5 * \pi * \beta)$.

Figure 26 shows the weight map for AEP with $\beta = 1/1.4$.

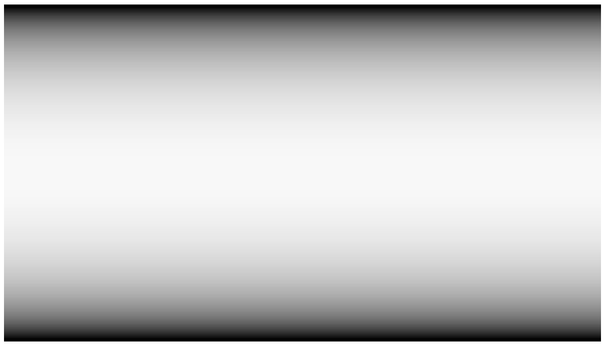


Figure 26. Weight map for AEP with $\beta = 1/1.4$

4.2.6 WS-PSNR calculation for OHP and ISP

The OHP and ISP projection formats have triangle faces. The weight distributions on all triangle faces are the same. Therefore, weights in only one triangle face need to be derived. To calculate weights for each position on a triangle face, the triangle is put into one rectangle of $FW \times FH$ resolution, as shown in the leftmost picture of Figure 27. Weights for positions outside of the triangle are set to 0 (the black area) and weights for positions inside the triangle face can be calculated using equation (109). The following is used to evaluate r and $d^2(i,j)$ in (109):

For OHP, $r = FW/\sqrt{6}$ and $d^2(i,j) = (i+0.5-FW/2)^2 + (j+0.5-2*FH/3)^2$;

For ISP, $r = FW/(4\sqrt{3}/(3+\sqrt{5}))$ and $d^2(i,j) = (i+0.5-FW/2)^2 + (j+0.5-2*FH/3)^2$.

The compact OHP [12] and compact ISP [8] in 360Lib contain some rotated and/or combined triangle faces. For these compact packing arrangements, the weights can be obtained by performing the same rotation and/or combination of the triangle faces based on the weights for one triangle face. The weight maps for compact OHP and compact ISP are illustrated in Figure 27. The compact ISP contains padded samples along some triangle boundaries. For these positions, the weights are set to 0, as shown in the dark areas in the rightmost picture of Figure 27.

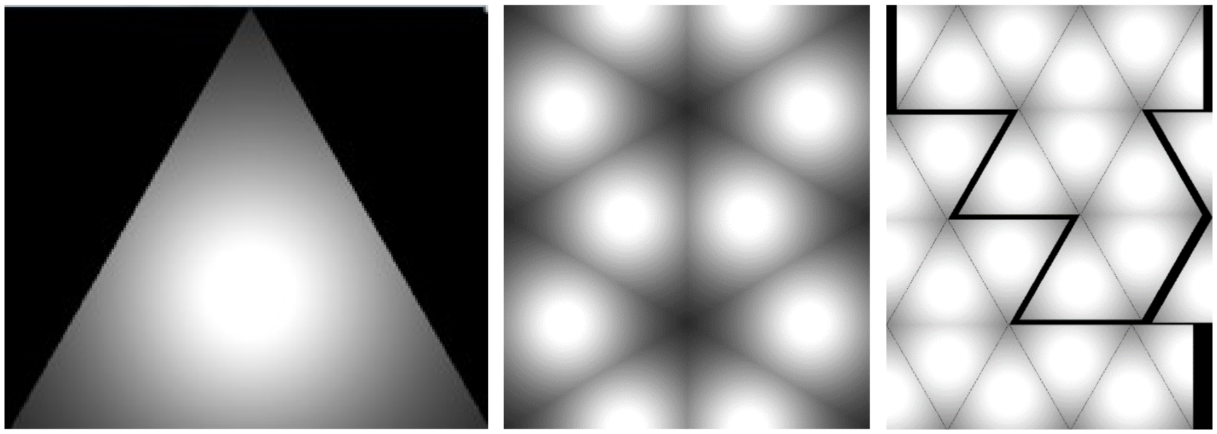


Figure 27. Weight maps for one triangle face (left), compact OHP (middle), and compact ISP (right)

4.2.7 WS-PSNR calculation for SSP

SSP contains the two circular areas and four square faces. Given a position (i, j), if it corresponds to an active sample position (i.e., located within a circle or a square face), then its corresponding latitude and longitude θ and ϕ, $\theta \in \left(\frac{-\pi}{2}, \frac{\pi}{2}\right), \phi \in (-\pi, \pi)$, are derived as specified in 2.8. Then, its weight $w(i, j)$ is calculated using equation (119). If the position (i, j) corresponds to an inactive sample position, its weight is set to 0, as shown in the black areas in Figure 28.

$$w(i,j)=\begin{cases}\frac{\pi}{2}-\text{¿}\theta\vee\text{¿},(i,j)\in f\overset{\circ}{a}ce \\ \frac{\cos\boxed{f_0}(\theta)}{\text{¿}}\frac{2\sqrt{2}}{\pi},(i,j)\in square\,face\end{cases} \qquad (119)$$

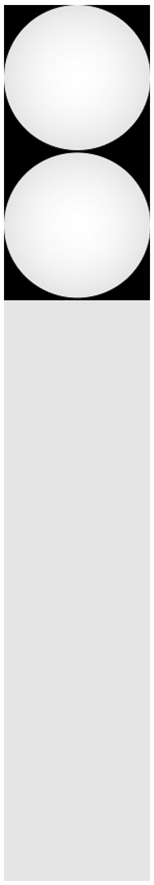


Figure 28. Weight map for SSP

4.2.8 WS-PSNR calculation for RSP

RSP contains two equal-sized faces with identical weight map function. Given a position (i, j), if a sample corresponds to an active sample position, its weight is set according to $\cos\theta$ function. If a sample corresponds to an inactive sample, it's weight is set to zero (and the sample is not considered in WS-PSNR computation). Weight map of RSP is shown in Figure 29.

$$w(i,j)=\begin{cases} 0 (i,j) \in \textit{inactive sample} \\ \cos(\theta), (i,j) \in \textit{active sample} \end{cases} \tag{120}$$

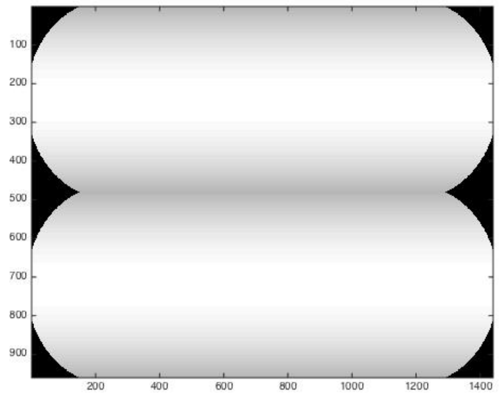


Figure 29. Weight map for RSP

4.2.9 WS-PSNR calculation for ECP

$(i, j), \quad 0 \le i < W, 0 \le j < H$

The WS-PSNR weight map for the ECP format is illustrated in 31. The weights are derived according to the WS-PSNR weights derivation guidance provided in [29]. Using the face index map shown in Figure 15 (c), the weights for faces $f = 3$ and 5 are set to $w = \pi \cdot padfactor / 6.0$, and the weights for faces $f = 2$ and 4 are set to $w = \pi \cdot padfactor \cdot apadfactor / 6.0$, where $padfactor$ and $apadfactor$ are as defined in eq. (64) and (65), respectively. Derivation of the weights for faces 0 and 1 is more complicated, and readers are referred to the 360Lib software [1] for detailed weight calculation.

The ECP format also contains padded regions four samples wide around the top and bottom halves of the packed picture. For these padded samples, the weights are set to zero.

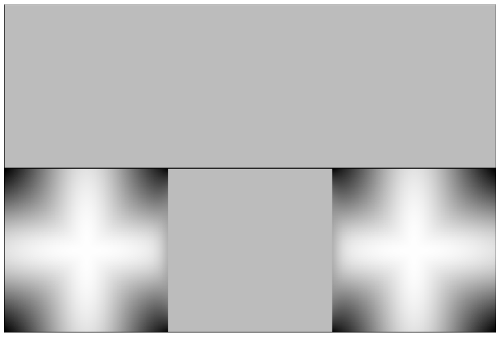


Figure 30. Weight map for ECP

31CPP-PSNR

Craster parabolic projection PNSR [9] is a generalization of omnidirectional image with a uniform samples distribution close to the distribution on the sample on a surface of unit sphere.

No additional weighting is applied to transformed samples.

Spherical coordinates are mapped on the 2D plane (Figure 18) using following equations:

$$m = R\theta\left[\left(2\cos\frac{2\phi}{3}\right) - 1\right]; n = \pi R \sin\frac{\phi}{3}; \tag{121}$$

where m and n are horizontal and vertical integer luma sample position in CPP projection respectively; θ and ϕ are longitude and latitude in spherical coordinate system (Figure 2); sphere has $R = 1$. Corresponding sampling position in origin projection may be derived using map3Dto2D() method of corresponding projection class.

CPP dimensions are derived from input projection parameters in a manner that no additional samples are introduced. Thus CPP width (w_{CPP}) and CPP height (h_{CPP}):

$$w_{CPP} = w_{ERP}^{¿},$$
$$h_{CPP} = \frac{w_{CPP}}{2}; \tag{122}$$

where $w_{ERP}^{¿}$ is coding ERP width for cross projection metrics and hi-fidelity ERP width for end-to-end metrics.

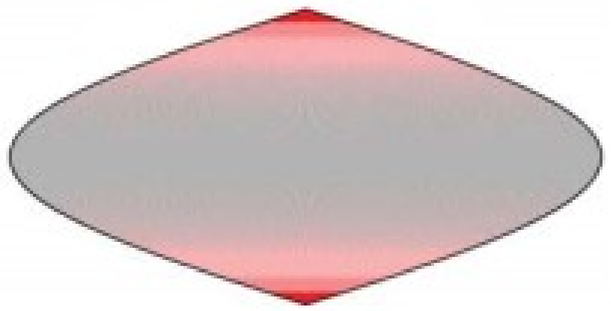


Figure 32. Craster parabolic projection image

Colour coding on Figure 32 represents a number of samples in ERP line used to represent single point on the sphere surface.

5 Implementation of trigonometry functions in 360Lib

The following table shows the corresponding standard C functions used in 360Lib to implement the trigonometry functions in this document.

Table 16. C functions used to implement trigonometry functions

Trigonometry function	C function
$\tan^{-1}(\frac{y}{x})$	$atan2(y,x)$
$\tan^{-1}(x)$	$atan(x)$
$\sin^{-1}(x)$	$asin(x)$
$\cos^{-1}(x)$	$acos(x)$
$\tanh(x)$	$\tanh(x)$

6 References

[1] https://jvet.hhi.fraunhofer.de/svn/svn_360Lib/trunk

[2] E. Alshina, J. Boyce, A. Abbas, Y. Ye, “JVET common test conditions and evaluation procedures for 360° video, JVET-H1030,” Oct 2017, Macao, China.

[3] Equirectangular projection, https://en.wikipedia.org/wiki/Equirectangular_projection

[4] Cubemap, https://en.wikipedia.org/wiki/Cube_mapping

[5] Lambert cylindrical equal-area projection, https://en.wikipedia.org/wiki/Lambert_cylindrical_equal-area_projection

[6] Octahedron, https://en.wikipedia.org/wiki/Octahedron

[7] Rectilinear projection, http://wiki.panotools.org/Rectilinear_Projection

[8] V. Zakharchenko, E. Alshina, K. P. Choi, A. Singh, A. Dsouza, “AhG8: Icosahedral projection for 360-degree video content,” Joint Video Exploration Team of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, JVET-D0028, Oct. 2016, Chengdu, China.

[9] V. Zakharchenko, E. Alshina, A. Singh, A. Dsouza, “AhG8: Suggested testing procedure for 360-degree video,” Joint Video Exploration Team of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, JVET-D0027, Oct. 2016, Chengdu, China.

[10] G. Van der Auwera, M. Coban, H. Fnu, M. Karczewicz, “AHG8: Truncated Square Pyramid Projection (TSP) For 360 Video,” Joint Video Exploration Team of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, JVET-D0071, Oct. 2016.

[11] C. Zhang, Y. Lu, J. Li, Z. Wen, “AhG8: segmented sphere projection (SSP) for 360-degree video content,” Joint Video Exploration Team of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, JVET-E0025, Jan. 2017, Geneva, Switzerland.

[12] H.-C. Lin, C.-C. Huang, C.-Y. Li, Y.-H. Lee, J.-L. Lin, S.-K. Chang, “AHG8: An improvement on the compact OHP layout,” Joint Video Exploration Team of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, JVET-E0056, Jan. 2017, Geneva, Switzerland.

[13] M. Yu, H. Lakshman, B. Girod, “A Framework to Evaluate Omnidirectional Video Coding Schemes”, IEEE International Symposium on Mixed and Augmented Reality, 2015.

[14] Y. Sun, A. Lu, L. Yu, “AHG8: WS-PSNR for 360 video objective quality evaluation,” Joint Video Exploration Team of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, JVET-D0040, Oct. 2016, Chengdu, China.

[15] M. Coban, G. Van der Auwera, M. Karczewicz, “AHG8: Adjusted cubemap projection for 360-degree video”, Joint Video Exploration Team of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, JVET-F0025, April 2017, Hobart, Australia.

[16] Y.-H. Lee, H.-C. Lin, J.-L. Lin, S.-K. Chang, C.-C. Ju, “AHG8: An improvement on compact octahedron projection with padding”, Joint Video Exploration Team of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, JVET-F0053, April 2017, Hobart, Australia.

[17] A. Abbas, D. Newman, “AHG8: Rotated Sphere Projection for 360 Video”, Joint Video Exploration Team of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, JVET-F0036, April 2017, Hobart, Australia.

[18] Q. Xu, J. Boyce, Y. He, Y. Ye, “360Lib modifications for spherical rotation”, Joint Video Exploration Team of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, JVET-F0065, April 2017, Hobart, Australia.

[19] Y. He, X. Xiu, Y. Ye, “AHG8: On cross-format S-PSNR-NN”, Joint Video Exploration Team of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, JVET-F0042, April 2017, Hobart, Australia.

[20] M. Zhou, “AHG8: A study on quality impact of line re-sampling rate in EAP,” Joint Video Exploration Team of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, JVET-G0051, July 2017, Torino, Italy.

[21] M. Zhou, “AHG8: A study on Equi-Angular Cubemap projection (EAC),” Joint Video Exploration Team of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, JVET-G0056, July 2017, Torino, Italy.

[22] G. Van der Auwera, M. Coban, M. Karczewicz, “AHG8: ECP with padding for 360-degree video,” Joint Video Exploration Team of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, JVET-G0074, July 2017, Torino, Italy.

[23] J. Boyce and Z. Deng, “EE4: Padded ERP (PERP) projection format,” Joint Video Exploration Team of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, JVET-G0098, July 2017, Torino, Italy.

[24] X. Xiu, Y. He, Y. Ye, “AHG8: On the derivation of weighted to spherically uniform PSNR (WS-PSNR) for adjusted cubemap projection (ACP) format,” Joint Video Exploration Team of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, JVET-G0088, July 2017, Torino, Italy.

[25] Y.-H. Lee, H.-C. Lin, J.-L. Lin, S.-K. Chang, C.-C. Ju, “EE4: ERP/EAP-based segmented sphere projection with different padding sizes,” Joint Video Exploration Team of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, JVET-G0097, July 2017, Torino, Italy.

[26] J. Boyce, “Report of BoG on 360 Video,” Joint Video Exploration Team of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, JVET-G0158, July 2017, Torino, Italy.

[27] M. Fernandez-Gausti, “Classroom notes: Analytic geometry of some rectilinear figures”, Int. J. Mathematical Education in Science and Technology, Vol. 23, No. 6, 1992, pp. 895–913.

[28] A. Abbas, D. Newman, “AHG8: An Update on RSP Projection,” Joint Video Exploration Team of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, JVET-H0056, October 2017, Macao, China.

[29] Y. Sun, L. Yu, “AHG8: The guidance of WS-PSNR weights derivation for different formats,” Joint Video Exploration Team of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, JVET-H0035, Oct. 2017, Macao, China.

View publication stats