Title:	**360Lib Software Manual**		
Status:	Software AHG working document		
Purpose:	Information		
Author(s) or Contact(s):	Yuwen He Kiho Choi Jian-Liang Lin Yule Sun Muhammed Coban Yao Lu Adeel Abbas Minhua Zhou Zhipin Deng Hyun-Mook Oh	Tel: Email:	yuwen.he@bytedance.com kiho14.choi@samsung.com jl.lin@mediatek.com sunyule@zju.edu.cn mcoban@qti.qualcomm.com yao.lu@owlreality.com minhua.zhou@broadcom.com zhipin.deng@intel.com hyunmook.oh@lge.com
Source:	InterDigital Communications Inc. Samsung Electronics Co,. Ltd MediaTek Inc. Zhejiang University Qualcomm Inc. OwlReality GoPro Broadcom Limited Intel Corp. LG Electronics Inc. Bytedance Inc.		

Abstract

This document describes the usage of 360Lib reference software for JVET experimentation.

1 Introduction

360Lib is developed based on projection format conversion tool for 360 video provided by InterDigital. The tool is added as a separate library to the HM, JEM, VTM, and BMS software package, which can be used together with encoder. A standalone application is also provided. The following list summarizes features and functionalities currently offered by 360Lib:

- Currently, 360Lib supports 14 different projection formats including equirectangular projection (ERP), adjusted equal-area projection (AEP), cubemap projection (CMP) / hemisphere CMP, octahedron projection (OHP), icosahedron projection (ISP), truncated square pyramid projection (TSP), segmented sphere projection (SSP), adjusted cubemap (ACP), rotated sphere projection (RSP), equatorial cylindrical projection (ECP), equi-angular cubemap (EAC) / hemisphere EAC,

hybrid equi-angular cubemap (HEC), fisheye, and generalized cubemap (GCMP). The tool supports forward and backward conversion between any two of these formats.

- Further, 360Lib also supports conversion between the same projection format but with different parameters. For example, it supports ERP to ERP with resolution change, CMP to CMP with different frame packing arrangements and/or with different face sizes, and so on. It also supports chroma format and bit-depth change between the input and output video (with same or different projection formats).

- For cubemap-like projection, such as ACP, RSP, ECP, EAC, HEC, and GCMP, 360Lib supports flexible frame packing arrangement. It allows any of the six cube faces to be rotated by 0, 90, 180 or 270 degrees. The user can further specify where to place that face (with desired rotation) in the 2D projected image. As examples, cubemap 4x3 and cubemap 3x2 are supported, as well as other packing arrangements specified by the user.

- 360Lib can work with video input and/or output bit-depths of ≥ 8 bits/sample, and chroma formats of 4:2:0, 4:4:4.

- 360Lib can render video inside a field of view (FOV) specified by the user with rectilinear projection.

- 360Lib supports objective quality evaluation metrics including S-PSNR [8], WS-PSNR [9], CPP-PSNR, static and dynamic viewport based PSNR. 360Lib also supports end-to-end and cross-format quality evaluation.

- 360Lib is integrated with VTM (360Lib-6.0 or later version) and JEM/HM (360Lib-5.1 or ealier version) encoder. The encoder can directly encode the output video after projection format conversion, which eliminates the need to always store the intermediate video as YUV files. The encoder also reports all those spherical metrics in the encoding stage as well as PSNR.

- Truncated square pyramid projection (TSP) included in 360Lib is for viewport-aware bitstream switching schemes. The included integration utilizes the cube geometry and warps the six cube faces into a compact frame with 25% resolution of the original equirectangular input frame.

- 360Lib also supports fisheye projection format. For this projection format, forward and backward conversions to and from ERP are supported.

2 Detailed description of main features

This section describes the main features of the 360Lib software package, including projection formats, frame packaging configurations for cubemap, octahedron, icosahedron, truncated square pyramid projection and segmented sphere projection, bit-depth and chroma formats supported, and quality evaluation metrics supported. We also describe interpolation filters supported by the 360Lib software.

2.1 Projection formats and examples

Table 1 lists the projection formats supported by 360Lib, their respective values of the GeometryType parameter in the configuration file, and respective example images.
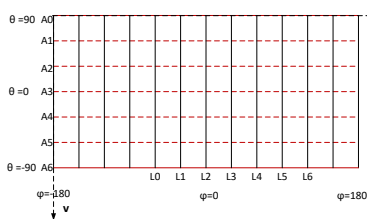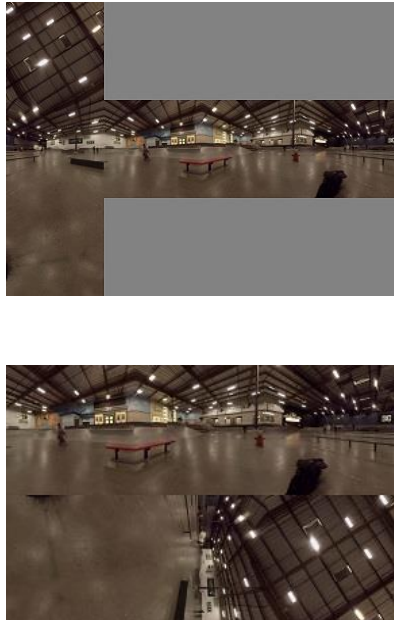
In the configuration file, two GeometryType parameters are used: InputGeometryType specifies the input video geometry, and CodingGeometryType specifies the output video geometry. For example, to convert from equirectangular to cubemap, these parameters should be set as follows:
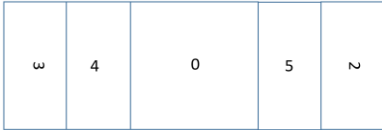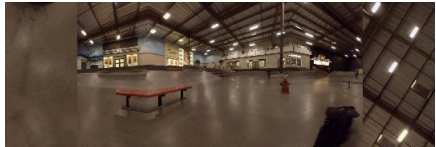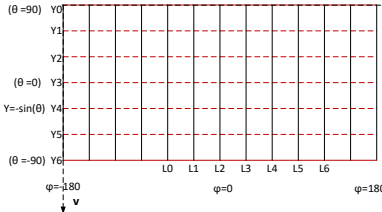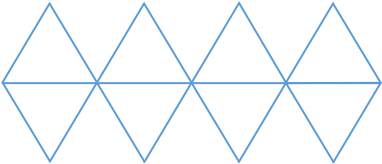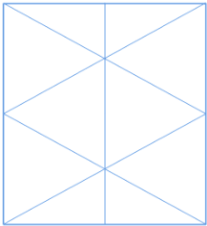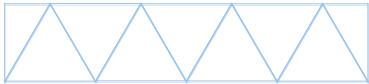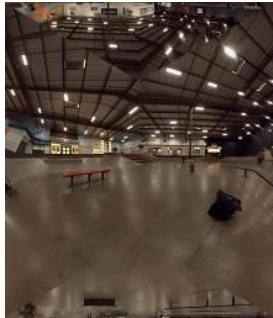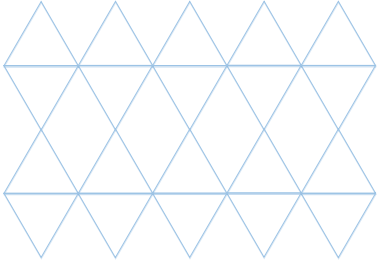
InputGeometryType : 0

CodingGeometryType : 1
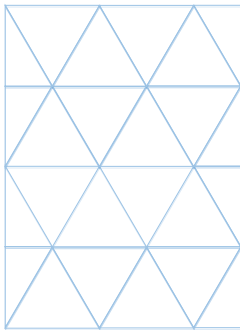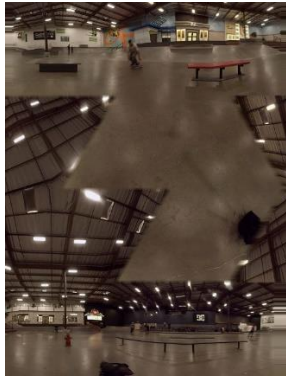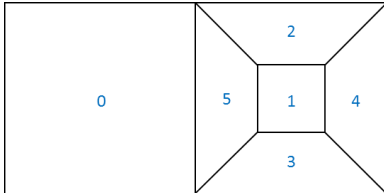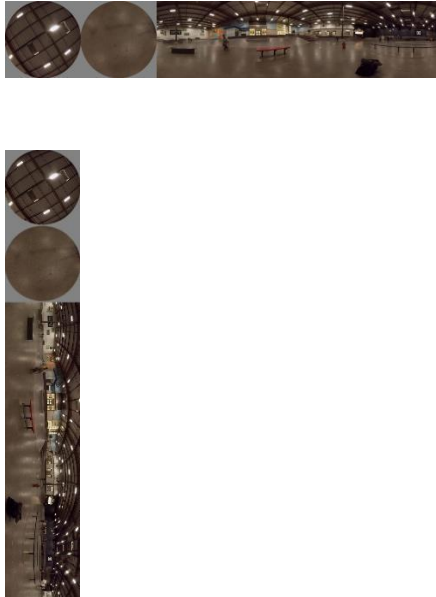
If the geometry type is equirectangular projection, two extra parameters are used to specify padding: InputPERP specifies whether the input equirectangular is padded equirectangular or not, CodingPERP specifies whether the output equirectangular is padded equirectangular or not. For example, to convert from equirectangular to padded equirectangular, the parameters should be set as follows:
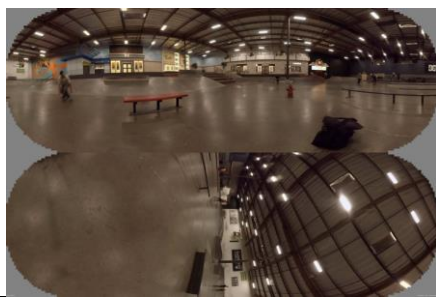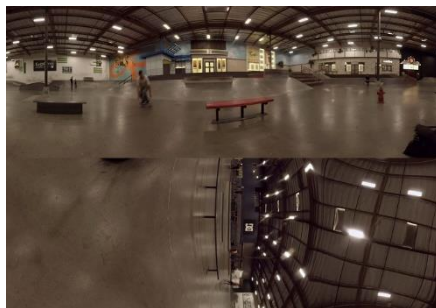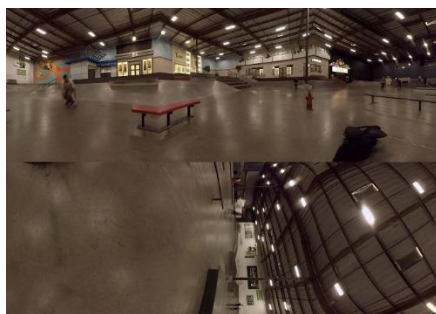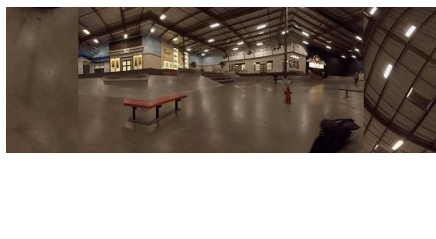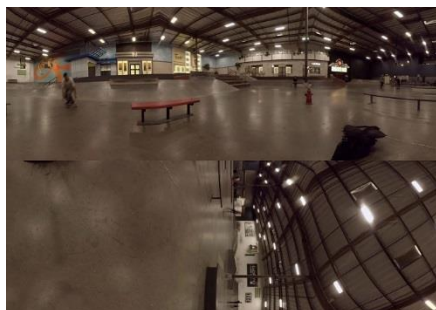
InputGeometryType	**: 0**
CodingGeometryType	**: 0**
InputPERP	**: 0**
CodingPERP	**: 1**
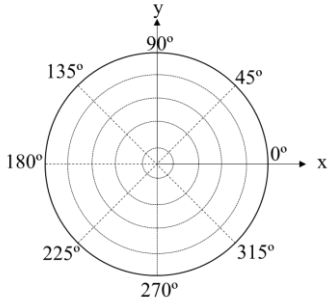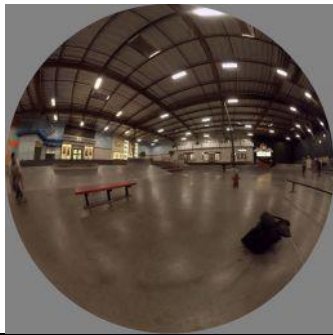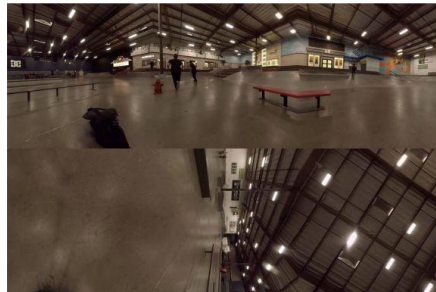
Table 1. Projection formats

Geometry Type	Projection	Frame packing	Example Image
0	Equirectangular [2]		
	Padded equirectangular [16]	Padding 8 pixels at both left and right sides	
1	Cubemap [3]	4x3: 3x2: 	

Date Saved: 2020-08-25

	Hemisphere cubemap		
2	Adjusted equal-area [4]		
3	Octahedron [5]	Non-compact [1]: Compact option 1 [10]: Compact option 2 [1]: 	
5	Icosahedron [6]	Non-compact [1]: Compact:	

7	Truncated Square Pyramid (TSP) [11]		
8	Segmented Sphere Projection (SSP) [12]	Horizontal frame packing: Vertical frame packing: 	
9	Adjusted cubemap projection [14]		

Date Saved: 2020-08-25

10	Rotated sphere projection [15]		
11	Equatorial cylindrical projection [17]		
12	Equi-angular cubemap [18]		
	Hemisphere equi-angular cubemap		
13	Hybrid equi-angular cubemap [20]		

Date Saved: 2020-08-25

14	Fisheye [21]		
15	Generalized cubemap [22][24]	3x2: <table><tr><td>4</td><td>0</td><td>5</td></tr><tr><td>3.</td><td>1.</td><td>2.</td></tr></table>	3x2:
		3x2 (viewing from spherical center): <table><tr><td>5</td><td>0</td><td>4</td></tr><tr><td>3</td><td>1</td><td>2</td></tr></table>	3x2:
		5x1 (hemisphere cubemap): <table><tr><td>3</td><td>4</td><td>0</td><td>5</td><td>2.</td></tr></table>	5x1 (hemisphere cubemap):

*GeometryType = 4 is used for viewport projection, GeometryType = 6 is used for Crasters Parabolic Projection

2.2 Frame packing for cubemap projection

When the input or output geometry is the cubemap format (that is, InputGeometryType = 1 or CodingGeometryType = 1), the respective configuration parameters SourceFPStructure and CodingFPStructure are used to specify the frame packing arrangement of the input and/or cubemap faces. The six faces of a cubemap are illustrated in Figure 1(a), and Figure 1(b) shows the 2D uv coordinates system used to store the cubemap faces: u is the horizontal axis and v is the vertical axis in the frame memory. As will be discussed below, each of the cubemap faces may be rotated by a multiple of 90^0 before being stored in the output frame packed image. Figure 1(b) illustrates how each of the six cubemap faces will be stored in the output frame packed image if no rotation is applied (that is, 0^0 rotation). When 90^0, 180^0, or 270^0 rotation is applied to a given face, that face will be rotated counter-clockwise based on the

orientation of the uv axes. The tool supports cubemap conversion from one frame packing arrangement to another frame packing arrangement.

(a)   (b)
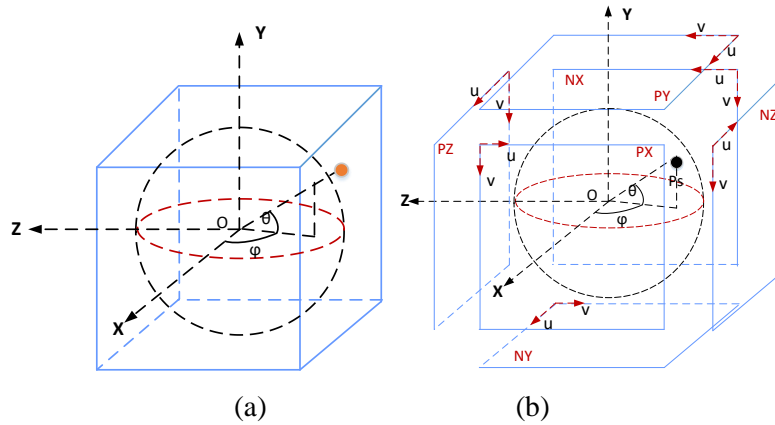
Figure 1. Cubemap format

Table 2 specifies the face index values corresponding to each of the six cubemap faces.

Table 2. Face index of cubemap

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
>5	"Null" face values used for frame packing only	To fill any remaining "holes" (i.e. invalid faces) in the frame packing arrangement. See example in Figure 3 below.

The frame packing arrangement parameters InputFPStructure and CodingFPStructure are specified as a set of integer values. The first two integer values specify the numbers of rows and the number of columns of faces, respectively. Then, a string of integer pairs follow; each of the pairs is arranged as the face index followed by degrees of rotation of that face. Note that all rotations are performed counter clockwise. The face index values are specified by scanning the packed frame in raster scan order, until all (# row × # columns) of them have all been specified. If (# row × # columns) is greater than 6 (for example, if cubemap4x3 is used), then some of the faces will be invalid faces. These invalid faces can be specified by a face index greater than 5 (see Table 2), and will be filled with default gray color, i.e., (Y,U,V) set to $(2^{(OutputBitDepth-1)}, 2^{(OutputBitDepth-1)}, 2^{(OutputBitDepth-1)})$, in the final frame packed output image.

To help users understand how to specify frame packing parameters, we provide two examples using an input image in Figure 2 in equirectangular projection (InputGeometryType=0). To obtain a corresponding cubemap 4x3 projection with the frame packing arrangement shown in Figure 3, CodingGeometryType should be set to 1, and the output frame packing parameter CodingFPStructure should be set as follows:

CodingFPStructure   : 3 4   2 90 6 0 7 0 8 0   1 0 4 0 0 0 5 0   3 270 9 0 10 0 11 0

Where "3" and "4" specify the number of face rows and the number of face columns, respectively. Then, three strings of integers follow to specify how to pack the faces. In these strings, the face index values are marked in red, and the rotation degrees in black. The first string of integers "2 90 6 0 7 0 8 0" specifies the face placement configuration for the first face row. "2" is the index of the TOP face as in Table 2, "90" means to rotate the TOP face by 90 degrees counter clockwise before putting it in the output image. "6", "7" and "8" are virtual face index specifying an empty (i.e., null) face, and "0" means that no rotation (0-degree rotation) is applied. The second string of integers "1 0 4 0 0 0 5 0" specifies the face placement configuration for the second face row. "1" is the index of the BACK face as in Table 2; "0" means that the BACK face is not rotated. Similarly for face "0" (the FRONT face)' and face "5" (the LEFT face). Note that all faces in the second face row are valid face with face index values no greater than 5. And the third string of integers "3 270 9 0 10 0 11 0" specifies the face placement configuration for the third face row. "3" is the index of the BOTTOM face as Table 2; "270" means that it is rotated by 270 degrees counter clockwise before being stored in the output image. The remaining faces "9" "10" and "11" are all null faces. With this specification for CodingFPStructure, the six faces will be packed as shown in Figure 3.

To convert the equirectangular projection in Figure 2 to a corresponding cubemap 3x2 projection shown in Figure 4, the configuration file parameters should be set as:

CodingFPStructure : 2 3   **4** 0 **0** 0 **5** 0   **3** 180 **1** 270 **2** 0

Where "2" and "3" specify the number of face rows and the number of face columns, respectively. Then, two strings of integers follow to specify how to pack the faces. The first string of integers "4 0 0 0 5 0" specifies the face placement configuration for the first face row. "4" "0" and "5" are the index values of the RIGHT, FRONT, and LEFT faces respectively; no rotation (0-degree rotation) is applied to these faces. The second string of integers "3 180 1 270 2 0" specifies the face placement configuration for the second face row, and can be interpreted in the same way as above. Note that all faces in cubemap3x2 packing arrangement are valid face with face index values no greater than 5. With this specification for CodingFPStructure, the six faces will be packed as shown in Figure 4.

Further, it is worth noting that the cubemap 4x3 and cubemap 3x2 examples in Figure 3 and Figure 4 are intentionally packed in a way that maximizes continuity along face boundaries in the output image.

In order to represent the picture viewing from the spherical center, horitontal flipping is also supported [24]. For example, if the rotation degree X is equal or greater than 360, it means horizontal flipping is applied first, then the face is rotated by (X-360) degrees in anti-clock wise. In this way, the face will be rendered as viewing from the spherical center toward outside. The frame packing parameter for cubemap 3x2 viewing from the sphere cener is as follows, and the packed picture is shown in Figure 5.

CodingFPStructure : 2 3   **5** 360 **0** 360 **4** 360   **3** 360 **1** 630 **2** 540
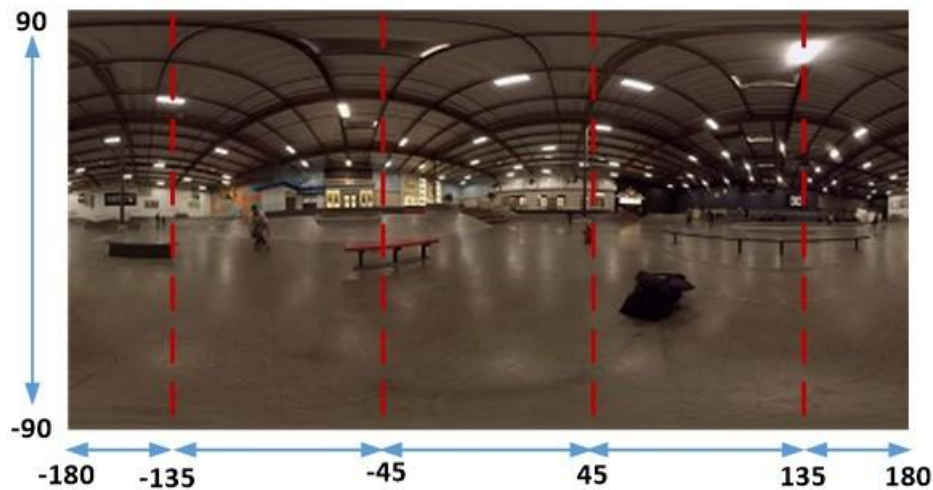
Figure 2. ERP picture as input: red dotted line indicates the longitude
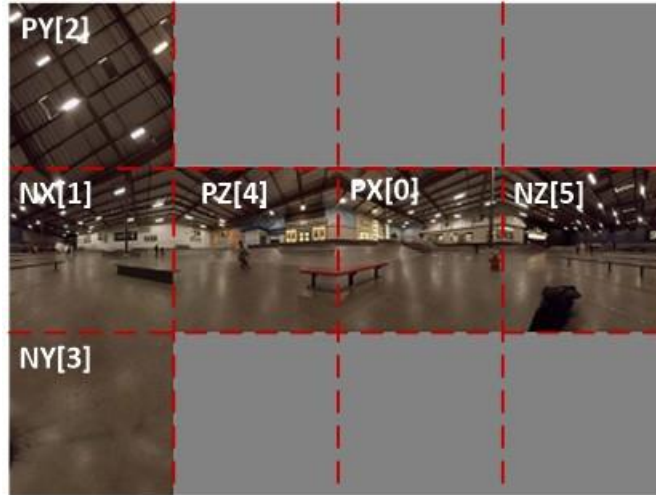
Figure 3. Cubemap 4x3 frame packing for the output: red dotted line indicates face boundary
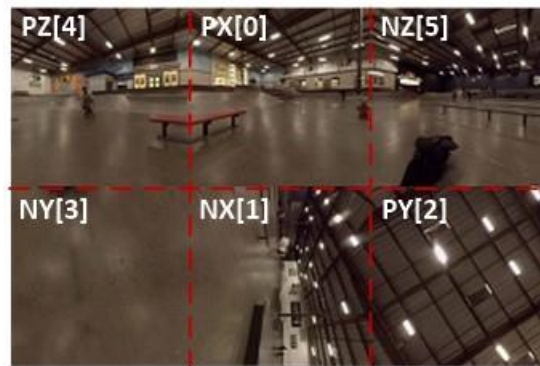
Figure 4. Cubemap 3x2 frame packing for the output: red dotted line indicates face boundary
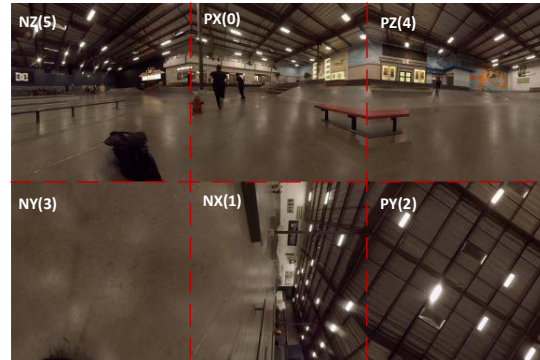
Figure 5. Cubemap 3x2 frame packing viewing from the spherical center toward outside

360Lib allows user to configure the face size in cubemap projection. As an example, the input picture resolution of 3840 x 1920 can be specified by the following parameters:

SourceWidth   **: 3840**

SourceHeight   **: 1920**

And the face size of the output cubemap projection of 960 x 960 can be specified by the following parameters:

CodingFaceWidth   **: 960**

CodingFaceHeight   **: 960**

The width and height of frame packed output picture will be calculated according to the numbers of face rows and face columns specified in CodingFPStructure, as discussed above.

2.2.1 Hemisphere cubemap projection

Hemisphere cubemap projection inherits Face indexes from Cubemap, however operates over front hemisphere (semi-ERP shown at Figure below) which is normally represented by 180x180 Equarectangular projection with paddings added.
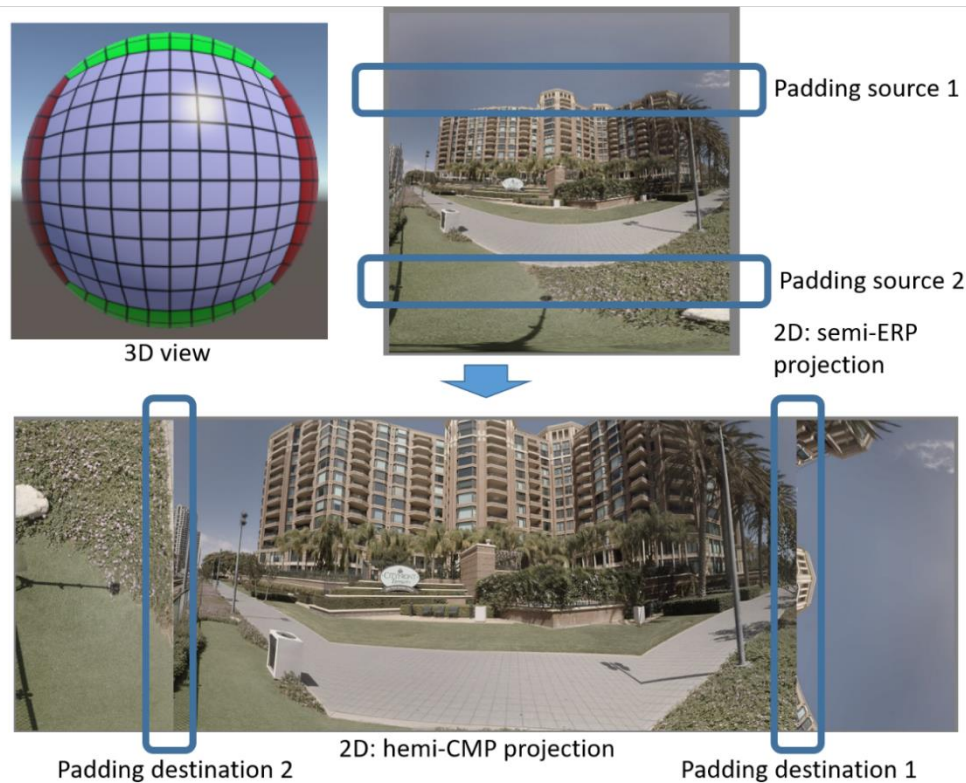
Figure 6. mapping of hemisphere into Hemisphere cubemap projection

Therefore Hemisphere flag (**InputGeometryHemiFlag** or **CodingGeometryHemiFlag**) is used to indicate the hemisphere cubemap. For example, the input is hemisphere CMP, so the following parameter InputGeometryHemiFlag is set to 1.

InputGeometryHemiFlag   : 1

Padding present or not for hemisphere cubemap is controlled by using parameters. **InputPCMP** is used to control padding for input format, and **CodingPCMP** is used to control padding for output format.

InputPCMP   : 1

CodingPCMP   : 1

When Hemisphere cubemap projection use as input – full 360x180 output in Equarectangular projection should be produced.

2.3 Frame packing for octahedron projection

Octahedron format projects the points on the unit sphere onto eight triangle faces as defined in Figure 7 (a) and Figure 7 (b). When the input/output geometry is octahedron (i.e., InputGeometryType = 3 or CodingGeometryType = 3), two frame packing arrangements are supported for the input/output faces, i.e., non-compact format and compact format. For non-compact format, the octahedron is defined in Figure 7 (a). In non-compact format, there are some invalid faces containing "null" values that are filled with gray samples. In compact format, the octahedron defined in Figure 7 (b) is used and the faces are arranged in such a way that no "null" values are present in the input/output picture. Figure 8 shows examples of non-compact format (Figure 8 (a)) and compact format (Figure 8 (b)) of the octahedron projection, respectively. In Figure 8 (b), in order to pack eight triangle faces into one single rectangle, Face 2 and 3 are vertically

split to two parts; "2-1" and "3-1" represent the first part of face 2 and 3 while "2-2" and "3-2" represent the second part. This compact format contains four discontinuous edges between face 2-1 and 6, face 3-1 and 7, face 2-2 and 4, and face 3-2 and 5. As shown in Figure 9, a band of 16 padded samples using vertical linear interpolation are added between the discontinuous edges.

In the configuration file, two frame packing parameters are used when the octahedron projection is applied to input/output: SourceCompactFPStructure specifies whether the input video is in non-compact format or compact format, and CodingCompactFPStructure specifies whether the output video is in non-compact format or compact format. For example, to convert from non-compact format to compact format, the two parameters should be set as follows:

SourceCompactFPStructure   **: 0**

CodingCompactFPStructure   **: 1**

(a)   (b)
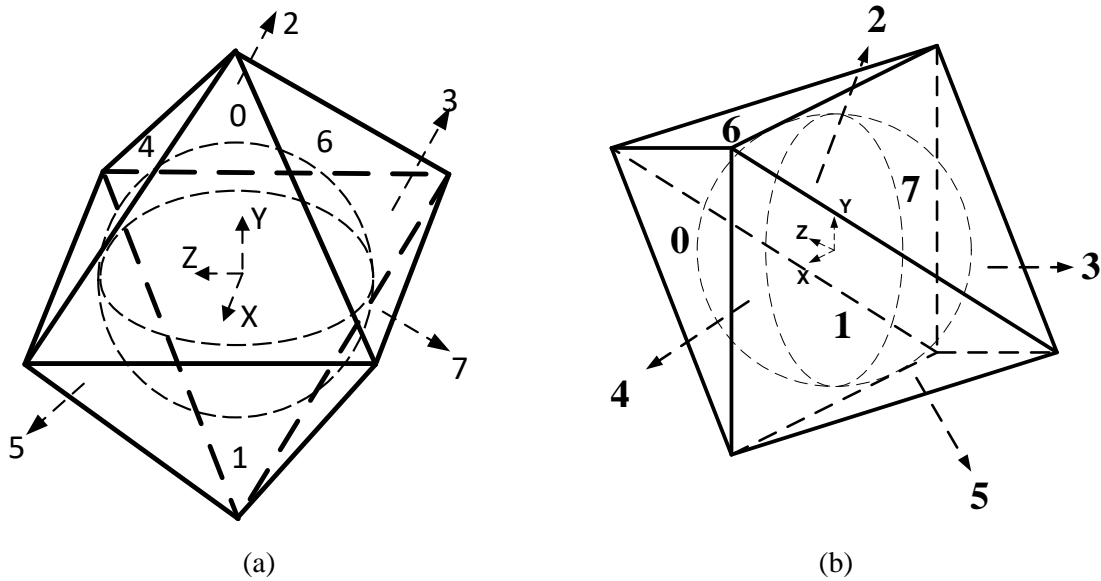
Figure 7. Octahedron definition: (a) definition for non-compact format; (b) definition for compact format

(a)
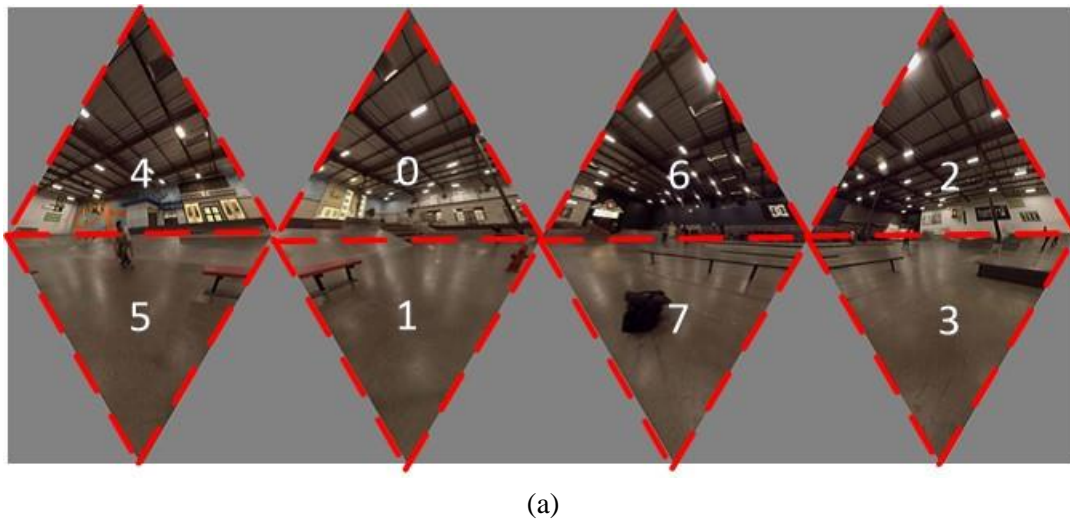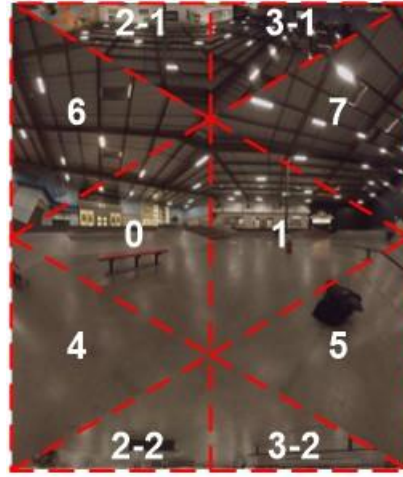
(b)
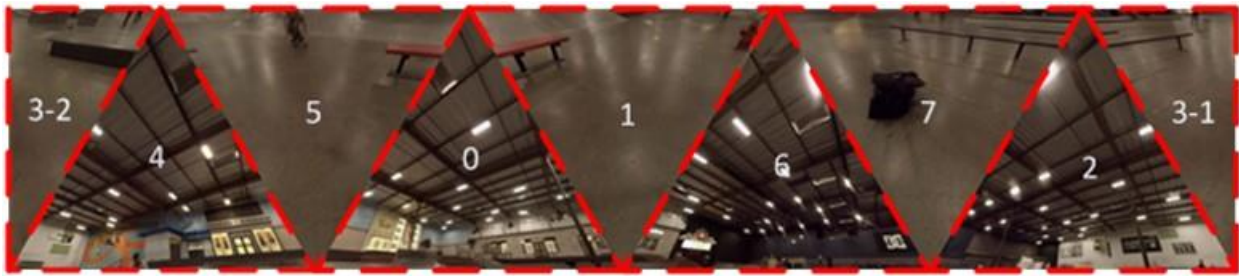
(c)

Figure 8. Octahedron input/output frame packing format: (a) non-compact format; (b) compact format option 1; (c) compact format option 2. Red lines indicate face boundaries.
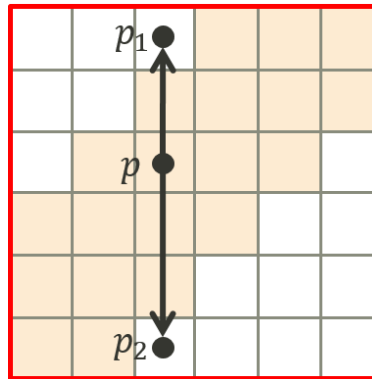
Figure 9. Padding along discontinuous edges using vertical linear interpolation.

2.4 Frame packing for icosahedron projection

Icosahedron projection has twenty faces, each of which is in the same triangle shape as that of octahedron projection. Figure 10 illustrates the definition of the twenty faces for the icosahedron. Similar to octahedron projection, both non-compact and compact format are support by 360Lib for the input/output frame packing arrangement, as specified by the configuration parameters SourceCompactFPStructure and CodingCompactFPStructure. Figure 11 shows examples of the non-compact (Figure 11 (a)) and compact (Figure 11 (b)) frame packing arrangement for the icosahedron projection. In order to shape the input/output into rectangle, in Figure 11 (b), Face 4, 5, 14 and 15 are vertically split to two parts and put along the left and right boundaries of the input/output picture; "4-1", "5-1", "14-1" and "15-1" represent the first part of those faces while "4-2", "5-2", "14-2" and "15-2" represent the second part of those faces. Faces "4-1", "5-2", "6", "7", "8", "9", "10", "11", "12, "13", "14-1" and "15-1" are placed in reverse vertical direction.

Additional vertical padding of N samples is introduced between faces "14-2", "16", "18" and "10", "12", "14-1"; and faces "4-2", "6", "8" and "0", "2", "4-1" (Figure 11 (b)); padded samples are ignored during inverse conversion. The conversion between the non-compact format and the compact format for the icosahedron projection is also lossless.
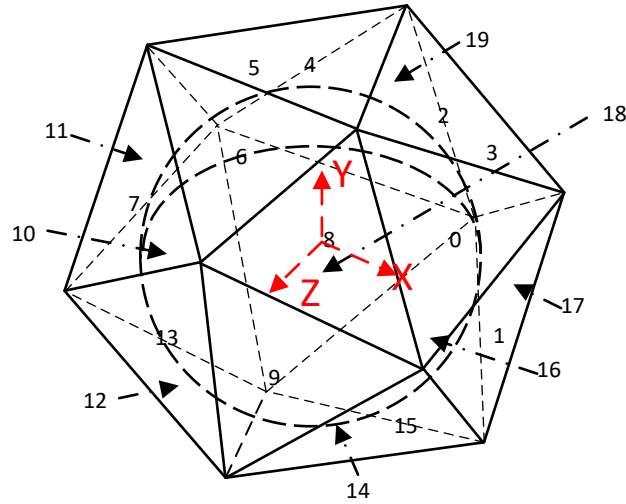
Figure 10. Icosahedron format
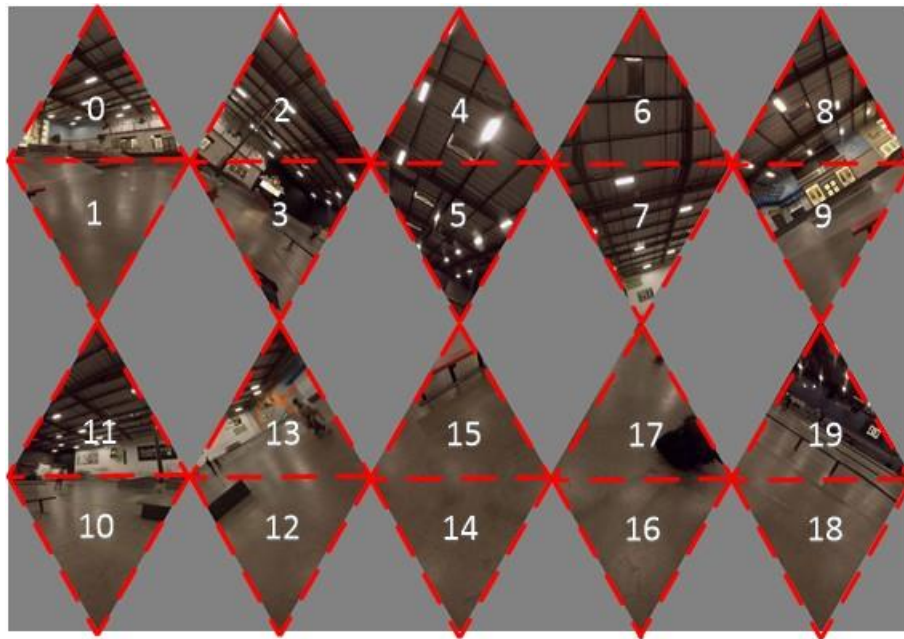
(a)

(b)
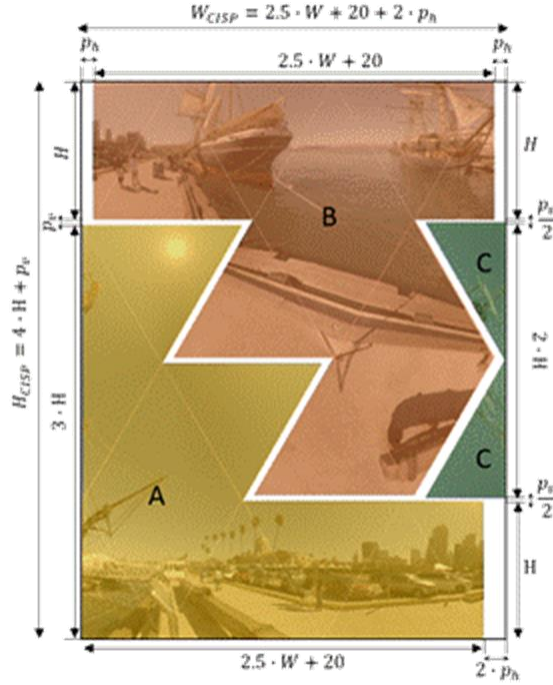
(c)

Figure 11. Icosahedron input/output frame packing format: (a) non-compact format; (b) compact format, red lines indicate face boundaries; green lines indicate padding; (c) the compact frame packing with padding, the while regions are padded regions.

2.5 *Frame packing for truncated-square-pyramid*

360Lib includes the truncated-square-pyramid (TSP) for viewport-aware bitstream switching schemes. The included integration utilizes the cube geometry and warps the six cube faces into a compact frame with 25% resolution of the original equirectangular input frame. When the input/output geometry is TSP (i.e., InputGeometryType = 7 or CodingGeometryType = 7), the frame packing in Figure 12 is supported. Figure 12 defines the (x,y) coordinates inside the packed TSP frame. The coordinates (x',y') inside a single cubemap face have normalized ranges [0.0,1.0]. In this implementation, the back face is subsampled by 4

horizontally and vertically, while the side faces are warped to trapezoidal regions. Table 3 specifies the forward and inverse mapping equations.
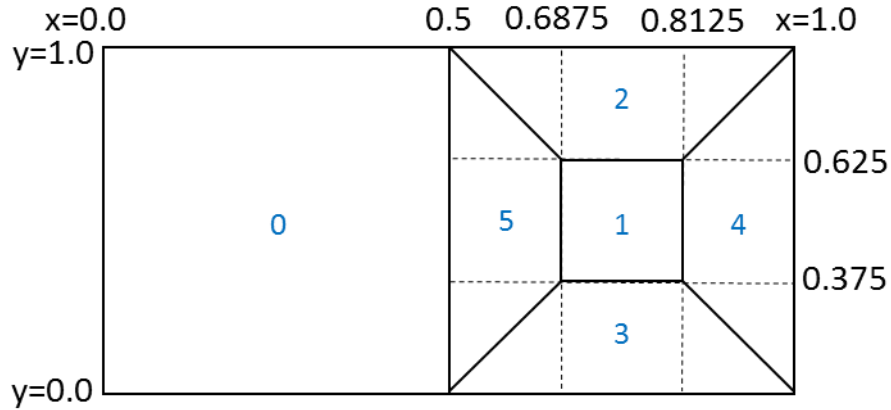
Figure 12. Truncated-square-pyramid frame packing format

Table 3 Forward and inverse equations between cube faces and TSP faces.

Forward equations (cube faces to TSP)	Inverse equations (TSP to cube faces)
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

2.6 *Frame packing for segmented sphere*

Segmented Sphere Projection (SSP) segments the sphere into 3 segments: north pole, equator and south pole. The boundaries of 3 segments are 45°N and 45°S. The north pole and south pole are mapped into 2 circles, and the projection of the equatorial segment is the same as EAP. The diameter of the circle is equal to the height of the equatorial segments because both pole segments and equatorial segment have a 90° latitude span.
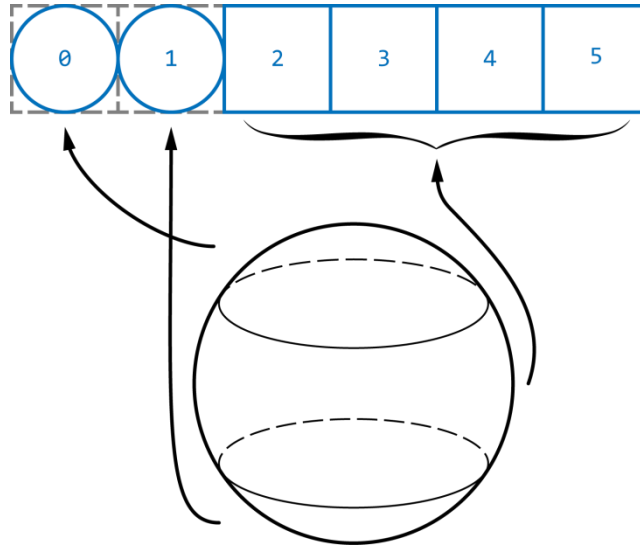
Figure 13. Segmented sphere conversion from sphere with horizontal frame packing

The equatorial segment is split into 4 squares in order to get faces of same size. The frame packing structure is depicted in Figure 14. The corners of poles are filled with "null" values. The face index is defined as Table 4.

Table 4. Face index definition of segmented sphere projection

Face index	Face region
0	North pole region (latitude > 45)
1	South pole region (latitude < -45)
2	Partition 1 in equator region (0<=longitude<90 and -45<=latitude<=45)
3	Partition 2 in equator region (90<=longitude<180 and -45<=latitude<=45)
4	Partition 3 in equator region (180<=longitude<270 and -45<=latitude<=45)
5	Partition 4 in equator region (270<=longitude<360 and -45<=latitude<=45)

CodingFPStructure : 1 6   0 0 1 0   2 0 3 0 4 0 5 0
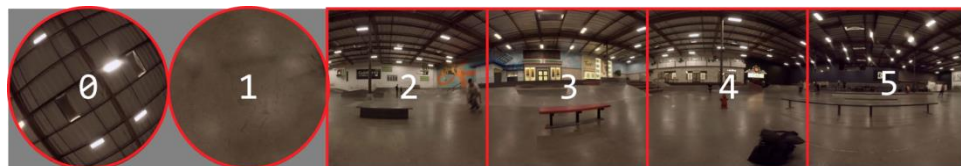
Figure 14. Horizontal segmented sphere frame packing format

The vertical frame packing is also used for the sake of a smaller line buffer. This is achieved by altering the "CodingFPStructure" parameter in the configuration file. In Figure 16(b), padding with the size of 8

pixels is applied on the boundary of the two pole segments, and the boundary between the pole segment and the equatorial segment.

CodingFPStructure   **: 6 1**   **0 0 1 0 2 270 3 270 4 270 5 270**
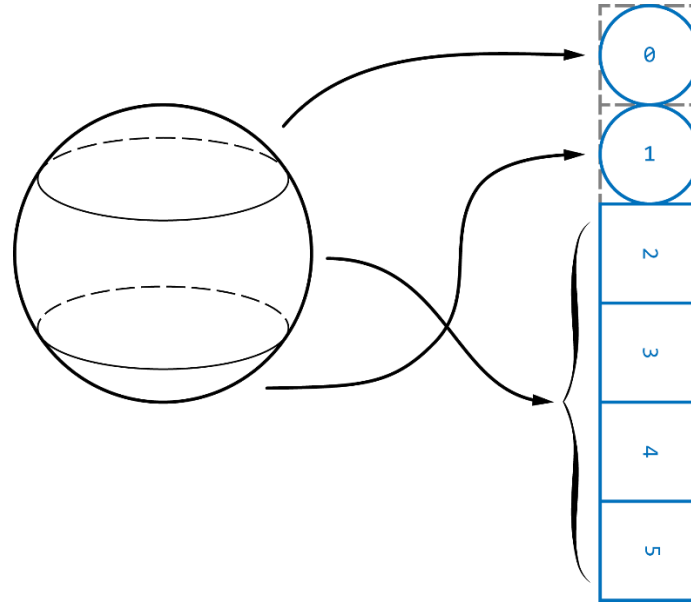
Figure 15. Segmented sphere conversion from sphere with vertical frame packing

(a)   (b)
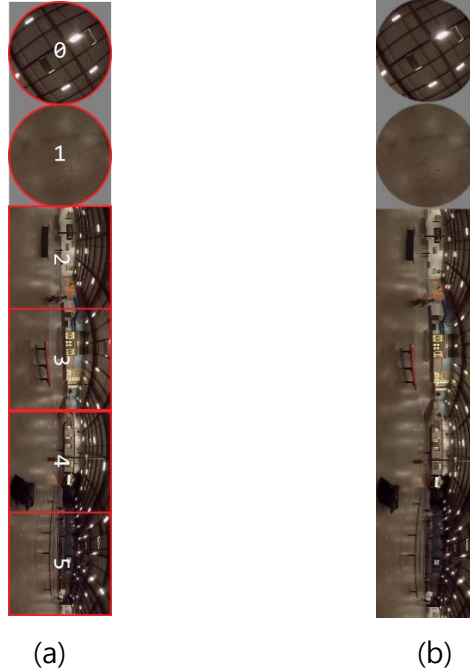
Figure 16. Vertical segmented sphere frame packing format: (a) vertical frame packing without padding; (b) vertical frame packing with padding.

2.7 Frame packing for adjusted cubemap projection

The adjusted cubemap projection (ACP) [14] adjusts the sampling position in each face of CMP. The face order is the same as that defined in Table 2 for CMP. The frame packing is also the same as CMP's frame packing. Following configuration parameters are specified for ACP with 3x2 frame packing layout.

    CodingGeometryType   **: 9**

    CodingFPStructure   **: 2 3**   **4 0 0 0 5 0**   **3 180 1 270 2 0**

2.8 Frame packing for rotated sphere projection

Rotated Sphere Projection (RSP) [15] and [19] segments the sphere into 2 identical segments stacked in two rows: top segment and bottom segment. The boundaries of two segments are arranged like the way a stitch line on a tennis ball is drawn. Since RSP has a 3:2 aspect ratio like cube map, it can be visualized as having 6 faces, as shown in Figure 17. Faces 4, 0 and 5 represent top segment, while faces 3, 1 and 2 represent bottom segment. It is important to note that unlike cube map, the two segments of RSP are perfectly continuous. The top segment is same as ERP, while bottom segment is also same as ERP, except that it is obtained by rotating sphere around Y and Z axis. In this way, both segments have a 90°x270° FOV span.
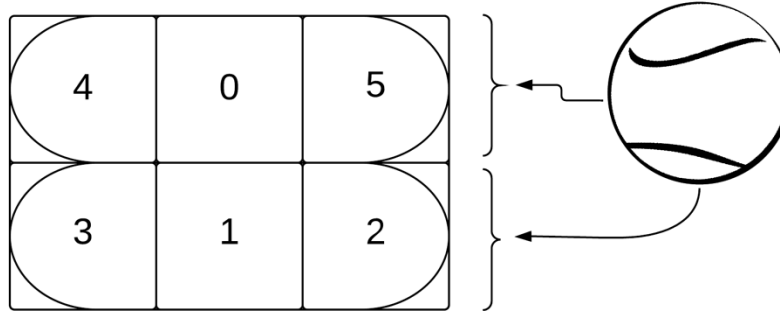
Figure 17. Rotated sphere conversion from sphere with horizontal frame packing

Corner pixels of each segment overlap with pixels from other segment and as a result, they can be greyed out and marked as inactive. This helps in achieving better coding efficiency. The arc marking inactive areas is drawn by taking a point on 2D plane, converting it to 3D and then back to 2D plane. During this conversion, if the face-index of the 2D point changes, the point lies in inactive region (because it actually belongs to the other face). RSP draws the arc on a 16x16 grid, although arc granularity is configurable as a compile time switch.

Following configuration parameters are specified for RSP.

    CodingGeometryType   : 10

    CodingFPStructure   : 2 3   4 0 0 0 5 0   3 0 1 0 2 0

2.9 Frame packing for equatorial cylindrical projection

The ERP picture is partitioned into 6 regions with equal area as shown in Figure 18. The 3x2 frame packing shown in Figure 19 is applied for ECP by default. In order to reduce the coding artifacts caused by face boundary, the padding around those face boundaries is applied. The padding size is 4 at each padding boundary in Figure 20.
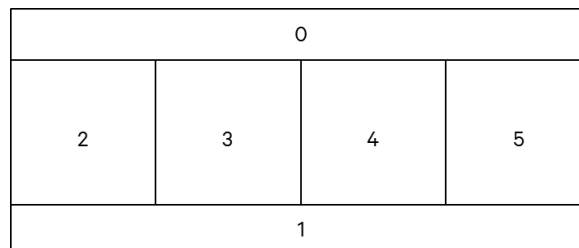
Figure 18. ERP portioning in ECP

Figure 19. 3x2 frame packing for ECP
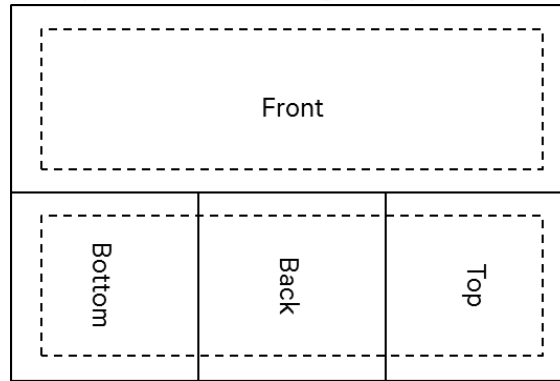
Figure 20. Padding scheme used in ECP

2.10 Frame packing for equi-angular cubemap projection

The frame packing for equi-angular cubemap (EAC) projection [18] is the same as cubemap 3x2 (section 2.2) by default. EAC supports any other frame packing layout that CMP supports in 360Lib.

2.10.1 Hemisphere EAC projeciton

Please refer to Hemisphere cubemap projection in section 2.2.1 for the major details.

GeometryType of Equi-angular cubemap projection should be used.

2.11 Frame packing for hybrid equi-angular cubemap projection

For hybrid equi-angular cubemap (HEC) projection [20], the face definition is the same as CMP as shown in Figure 1 and Table 2. HEC adopts the compact 3x2 layout with two contiuouse face rows, [4, 0, 5] and [3, 1, 2]. To reduce the seam artifacts caused by the discontinuous edge and the boundaries, padding with 4 pixel width is applied around the two face rows as shown in Figure 21.
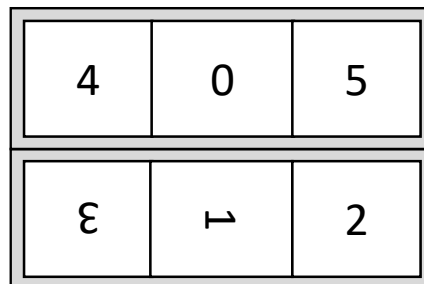
Figure 21. 3x2 frame packing for HEC with the gray region indicating the padding region

The following configuration parameters are specified for HEC with a 3x2 frame packing layout.

    CodingGeometryType   : 13

    CodingFPStructure   : 2 3   4 0 0 0 5 0   3 180 1 270 2 0

2.12 Frame packing for fisheye projection

The 360Lib supports conversion from a single fisheye circular projection format [21] to a ERP format, and vice versa. A point on an circular image of the fisheye circular projection can be matched with a point on a unit sphere where the coverage on a sphere is determined by the field of view (FoV) of the fisheye camera lens. e.g., when FoV is equal to 180 degrees, the coverage on a corresponding sphere is hemi-sphere.

In Figure 22, correspondence of a point on fisheye circular projection, sphere coordinate, and ERP is described. In the conversion from fisheye circular projection to the sphere coordinate, the angle φ from the YZ plane is assumed to be a linear function of the radius. After the fisheye circular projection format to sphere coordinate conversion, the point could be mapped to ERP format by calculating longitude and latitude of the point.
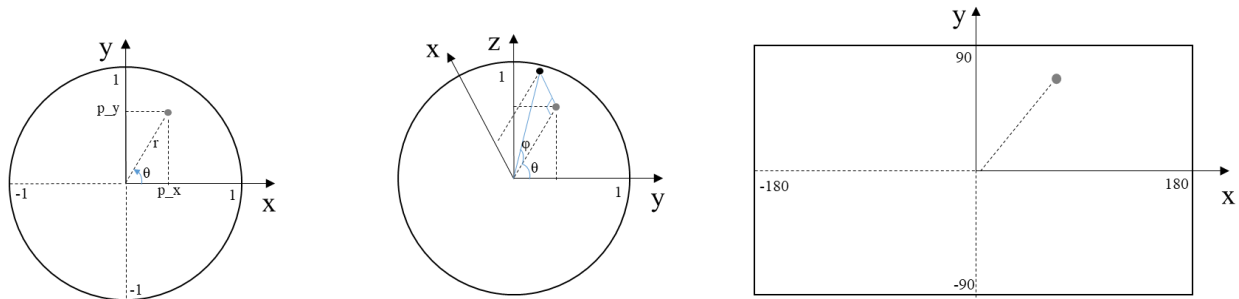
Figure 22. Correspondence of a point on (a) fisheye circular projection, (b) sphere coordinate, and (c) ERP

For the conversion from fisheye to ERP, the input and coding geometry types of a configuration file are set as follows:

InputGeometryType   **: 14**

CodingGeometryType   **: 0**

In addition, fisheye video parameter valuses should be set in the configuration file, which describe xy center position and radius of the circular image, field of view of the fisheye lens, XYZ spherical position corresponding to the fisheye video center, and size and location of the rectangular frame. Those parameters should be set for both forward and backward conversion. As an example, the followings are used in the example configuration files:

```
FisheyeCircularRegionCentreX        : 511.5
FisheyeCircularRegionCentreY        : 511.5
FisheyeCircularRegionRadius         : 511.5
FisheyeFieldOfView                  : 21000
FisheyeCenterAzimuth                : 0
FisheyeCenterElevation              : 0
FisheyeCenterTilt                   : 0
FisheyeRectRegionTop                : 0
FisheyeRectRegionLeft               : 0
FisheyeRectRegionWidth              : 1024
FisheyeRectRegionHeight             : 1024
```

In Figure 23, a PoleVault content in ERP format is used to creat fisheye circular projection image with the parameters defined above. When the fisheye circular projection is converted back to ERP, the resultant image shows partially covered ERP where the gray color represents the area that is not covered by the fisheye circular image.

(a) Input video (PoleVault_le_3840_1920_30fps_8bit_420_erp.yuv)

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

(c) Reconstructed ERP from fisheye format
Figure 23. Result of conversion between fisheye circular projection and ERP

2.13 *Frame packing for generalized cubemap projection*

The generalized cubemap projection (GCMP) [22][23] supports different packing types including the hemisphere cubemap projection layouts, mapping function types and guard bands. The face definition of GCMP is the same as CMP as shown in Figure 1 and Table 2.

The packing layout is specified by the number of rows and columns in the parameter **FPStructure**. When five faces are specified in **FPStructure**, the hemisphere cubemap will be applied. Otherwise, if six faces are specified, the general cubemap corresponding to the 360°x180° full sphere will be applied.

The padding on the discontinuous face boundaries can be enabled by **GCMPPaddingFlag**. When the padding is enabled, **GCMPPaddingType**, **GCMPPaddingExteriorFlag**, and **GCMPPaddingSize** should be specified. **GCMPPaddingType** indicates the padding type as listed in Table 5. **GCMPPaddingExteriorFlag** controlles the padding on the layout boundaries. **GCMPPaddingSize** indicates the size of the padding. Table 6 shows the supported packing layouts with the different padding settings.

Table 5. Padding types supported in GCMP

GCMPPaddingType	Padding type
0	Reserved

1	Repetitive padding
2	Copy from spherical neighboring faces
3	Geometry padding

Table 6. The combinations of the packing types and padding supported in GCMP. The number in the faces corresponds to the order of the face indices specified in FPStructure.

# of rows	# of columns	GCMPPaddingFlag=0	GCMPPaddingFlag=1	
			GCMPPaddingExteriorFlag=0	GCMPPaddingExteriorFlag=1
6	1			
3	2			
2	3			
1	6			
1	5			
5	1			

For the mapping function, CMP, EAC, or a parameterized mapping function is specified by the parameter **GCMPMappingType** as listed in Table 7. When a parameterized mapping function is used, the parameters a_x, b_x, a_y, and b_y should be specified in **GCMPSettings** for each face.

Table 7. Mapping function types supported in GCMP

GCMPMappingType	Mapping function type

0	CMP
1	EAC
2	parameterized mapping function: $$\begin{cases} f_x(u,v) = \dfrac{u}{1 + a_x(1 - b_x v^2)(1 - u^2)} \\ f_y(u,v) = \dfrac{v}{1 + a_y(1 - b_y u^2)(1 - v^2)} \end{cases},$$ where $a_x \in [0,1], b_x \in \{0,1\}, a_y \in [0,1], and\ b_y \in \{0,1\}$

The following configuration parameters are an example for simulating HEC with a 3x2 frame packing layout without padding.

CodingGeometryType   **: 15**

CodingFPStructure   **: 2 3   4 0 0 0 5 0   3 180 1 270 2 0**

CodingGCMPMappingType: 2

CodingGCMPSettings: 0.28 0 0.4 1   0.28 0 0.4 1   0.28 0 0.4 1   0.28 0 0.28 0   0.28 0 0.4 1 0.28 0 0.28 0

CodingGCMPPaddingFlag: 0

The following configuration parameters are an example for simulating the hemisphere equi-angular cubemap projection with repetitive padding on the discontinuous edges.

CodingGeometryType   **: 15**

CodingFPStructure   **: 1 5   3 0 4 0 0 0 5 0 2 180**

CodingGCMPMappingType: 1

CodingGCMPPaddingFlag: 1

CodingGCMPPaddingType: 1

CodingGCMPPaddingExteriorFlag: 0

CodingGCMPPaddingSize: 8

VVC supports GCMP SEI messages. The face flipping needs to be considered in frame packing when SEI framepacking is applied because 360-degree video is viewed from spherical center in the definition of VVC GCMP SEI message. The flipping is supported in 360Lib with rotation degrees plus 360 degrees.

2.14 Bit-depth and chroma format supported

360Lib supports video bit-depth of up to 16 bits. Four bit-depth related parameters are supported in the configuration files: "InputBitDepth" specifies input bit-depth, "OutputBitDepth" specifies output bit-depth, and "InternalBitDepth" specifies internal bit-depth. "ReferenceBitDepth" specifies the bit-depth of the original (i.e., reference) video used in metric calculation. By default, the input bit-depth is set to 8, and the internal and output bit-depths are set to the input bit-depth, and reference bit-depth is set to the output bit-depth. If the reference bit-depth and the output bit-depth are different, for the purpose of metric calculation, the sample values in lower bit-depth are first scaled to the higher bit-depth by left shifting the bit-depth difference, then the metric is calculated in the higher bit-depth domain.

360Lib supports 4:4:4 and 4:2:0 chroma formats as input and output. In the configuration file, "InputChromaFormat" specifies the input chroma format;
"OutputChromaFormat" specifies the input chroma format;

"InternalChromaFormat" specifies the internal chroma format for conversion;

By default, the input chroma format is set to 4:2:0, and internal and output chroma format are set to the input chroma format.

Two additional chroma format related parameters, "ChromaSampleLocType" and "OutputChromaSampleLocType", are added to specify the chroma sample location type for input and output 4:2:0 video, respectively. The chroma sample location types are defined according to the same location type ID in HEVC specification. By default, ChromaSampleLocType and OutputChromaSampleLocType are set to 0.

If the internal chroma format is 4:4:4 while the input is 4:2:0, the chroma upsampling is applied with proper upsampling filter according to ChromaSampleLocType. If the input chorma format is 4:2:0 and ChromaSampleLocType is 0, and the internal chroma format is 4:2:0, then the parameter "ResampleChroma" is used to control if align the chroma sampling phase with luma sampling phase before projection conversion.

2.15 Objective quality metrics supported

The 360Lib conversion software supports five quality metrics listed in Table 8. The user can select any quality metric to evaluate the objective quality between the reference images and the test images by setting the configuration file parameters "PSNR", "SPSNR_NN", "WSPSNR", "SPSNR_I", and "CPP_PSNR" to 1 to enable or to 0 to disable the corresponding metric calculation. By default, all metrics will be calculated in one conversion process.

In order to use any of these metrics, the user needs to provide a reference file name.   For example,

RefFile   :   reference_file_name

When "SPSNR_NN" or "SPSNR_I" is set to 1, S-PSNR calculation without interpolation (SPSNR_NN) or with interpolation (SPSNR_I) is enabled and one additional parameter file "SphFile" needs to be specified in the configuration. For example,

SphFile   :   sphere_655362.txt

The file sphere_655362.txt contains the positon of a set of points uniformly sampled on the unit sphere used for distortion calculation.

The file latweights.txt contains the lookup table of weights for different latitudes. Parameter file sphere_655362.txt is based on [8] and provided in the 360Lib package under "./cfg/360Lib/".

PSNR and WS-SPSNR calculation do not need additional parameter files.

For example, to evaluate the quality impact due to projection format conversion, the user can convert projection A to projection B and then convert from projection B back to projection A using 360Lib. The user can then measure the distortion using any of these metrics between the original picture in projection A and the back-converted picture in projection A. Or, to evaluate the quality impact due to compression, the user can measure the distortion using any of these metrics between the picture in a given projection before and after compression. The user can also evaluate the quality impact due to the combination of projection format conversion and compression. The "ReferenceFaceWidth" and "ReferenceFaceHeight" need to be set when those spherical metrics needs to be calculated using 360Lib standalone conversion application.

The 360Lib software integrated with the encoder supports ten quality metrics listed in Table 9.

Table 8. Quality metrics supported

Metric name	Characteristics

Date Saved: 2020-08-25

PSNR	Calculate PSNR based on all samples with equal weight.
WS-PSNR	Calculate PSNR based on all samples; the distortion is weighted by sample area on corresponding spherical surface.
Spherical PSNR without interpolation (S-PSNR-NN)	Calculate PSNR based on a point set evenly sampled on the unit sphere; the projected position is rounded to nearest neighbor integer position.
Spherical PSNR with interpolation (S-PSNR-I)	Calculate PSNR based on a point set evenly sampled on the unit sphere; the sample value at the projected position is interpolated with neighboring samples at integer positions.
CPP-PSNR	Calculate PSNR in Crasters Parabolic Projection format.

Table 9. Quality metrics supported in 360 video encoder

Metric name	Characteristics
End to end S-PSNR-NN	Calculate S-PSNR-NN between the original video and reconstructed video in the same projection format and resolution as original video.
End to end S-PSNR-I	Calculate S-PSNR-I between the original video and reconstructed video in the same projection format and resolution as original video.
End to end CPP-PSNR	Calculate CPP-PSNR between the original video and reconstructed video in CPP domain with the same resolution as original video.
End to end WS-PSNR	Calculate WS-PSNR between the original video and reconstructed video in the same projection format and resolution as original video.
Dynamic viewport PSNR	Calculate WS-PSNR between the original viewport generated from original video and the reconstructed viewport generated from encoder reconstructed video.
Cross-format S-PSNR-NN	Calculate S-PSNR-NN between the original video and the reconstructed video from encoder.
Cross-format S-PSNR-I	Calculate S-PSNR-I between the original video and the reconstructed video from encoder.

Cross-format CPP-PSNR	Calculate CPP-PSNR between the original video and the reconstructed video from encoder; the resolution is the same as the reconstructed video from encoder.
WS-PSNR	Calculate WS-PSNR between the converted video from original video in the coding projection format and the reconstructed video from encoder; the resolution for WS-PSNR calculation is coding resolution.
Codec S-PSNR-NN	Calculate S-PSNR-NN between the converted video in the coding projection format from the original video and reconstructed video from the encoder.
PSNR	Calculate PSNR between the converted video from original video in the coding projection format and resolution and the reconstructed video from encoder;

2.16 Interpolation filters

Choice of interpolation filter can be specified with the parameter "InterpolationMethod". Table 10 lists five interpolation methods supported by 360Lib and their respective indices in the configuration file. Further, the interpolation method for luma and chroma components can be specified separately. Parameter "InterpolationMethodY" is used to specify the interpolation method for the luma component, and parameter "InterpolationMethodC" is used to specify the interpolation method for the chroma components.

Table 10. Interpolation methods supported

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

2.17 Outputting sphere coordinates of the output projection format

To facilitate further analysis for those who are interested in doing so, the 360Lib tool supports writing out the sphere coordinates associated with the output projection format. Specifically, user can choose to write out to a file the latitude and longitude values of each point on the sphere corresponding to each sample position in the destination projection format. For example, if the output projection format is cubemap, then the corresponding latitude and longitude values on the sphere of each point in the frame-packed cubemap image can be written out to a txt file, whose file name is specified by the configuration parameter "SpherePointsFile". For example,

SpherePointsFile   :   CMP_SpherePoints.txt

The latitude value is in the range of [-90, 90], and the longitude value is in the range of [-180, 180).

2.18 Sphere Rotation

The 360Lib tool supports to rotate the sphere during the projection format conversion. For example, the center in original ERP format is the center front view. When converting to another format, the user may want to put the center to another position in the sphere. The software defines 3D rotation along X, Y, Z.

SVideoRotation   :   yaw   pitch   roll

yaw: counterclockwise rotation in degree along Y axis;
pitch: counterclockwise rotation in degree along (-Z) axis;
roll: counterclockwise rotation in degree along X axis;
Those input angles can be in floating point, and they are represented in fixed point internally. The precision of fixed point is 1/100 degree.

3   360Lib software package

3.1   Projection conversion software

The 360Lib software package includes many example configuration files that can be used to perform projection conversion between any two projection formats, as listed in Table 11. These example configuration files are stored in the directory "./cfg/360Lib/". Table 11 shows an additional value of CodingGeometryType (CodingGeometryType = 4). This additional value will be explained next in section 3.2. For convenience, the input video property related settings for all 360 test sequences can be found in the directory "./cfg/per-sequence/360/". A complete list of parameters supported by 360Lib can be obtained with following command:

360ConvertApp –help

For example, to convert one ERP test sequence (e.g. skateboarding_vr) to cubemap 4x3 for one frame, the following command can be used:

360ConvertApp –c ../cfg/360Lib/360convert_ERP_Cubemap4x3.cfg –c ../cfg/per-sequence/360/360test_skateboarding_vr.cfg –o cubemap4x3_fromERP.yuv –f 1

Note that the output picture resolution, sample bit-depth, and chroma format will be automatically appended to the output file name specified by the user (cubemap4x3_fromERP).

Table 11. Example configuration files for different conversions

InputGeometryType	CodingGeometryType	Example configuration file
0	0	360convert_ERP_ERP.cfg
(ERP)	0 (CodingPERP = 1)	360convert_ERP_PERP.cfg

	1(3x2)	360convert_ERP_Cubemap3x2.cfg
	1 (HemisphereFlag = 1)	360convert_ERP_HCMP.cfg
	2	360convert_ERP_AEP.cfg
	3(compact)	360convert_ERP_COHP.cfg
	4	360convert_ERP_RVP.cfg
	5(compact)	360convert_ERP_CISP.cfg
	7	360convert_ERP_TSP.cfg
	8	360convert_ERP_SSP.cfg
	9	360convert_ERP_ACP.cfg
	10	360convert_ERP_RSP.cfg
	11	360convert_ERP_ECP.cfg
	12	360convert_ERP_EAC.cfg
	12 (HemisphereFlag = 1)	360convert_ERP_HEAC.cfg
	13	360convert_ERP_HEC.cfg
	14	360convert_ERP_FISHEYE.cfg
	15	360convert_ERP_GCMP.cfg
	15 (viewing from spherical center)	360convert_ERP_GCMPSEI.cfg
0 (ERP, InputPERP = 1)	0 (CodingPERP = 0)	360convert_PERP_ERP.cfg
	4	360convert_PERP_RVP.cfg
1 (Cubemap 3x2)	0	360convert_Cubemap4x3_ERP.cfg
	1(3x2)	360convert_Cubemap3x2_Cubemap3x2.cfg
	2	360convert_Cubemap3x2_AEP.cfg
	3(compact)	360convert_Cubemap3x2_COHP.cfg
	4	360convert_Cubemap3x2_RVP.cfg
	5(compact)	360convert_Cubemap3x2_CISP.cfg
1 (HCMP, HemisphereFlag=1)	0	360convert_HCMP_ERP.cfg
	4	360convert_HCMP_RVP.cfg
2 (Adjusted equal-area)	0	360convert_AEP_ERP.cfg
	1(3x2)	360convert_AEP_Cubemap3x2.cfg
	2	360convert_AEP_AEP.cfg
	3(compact)	360convert_AEP_COHP.cfg
	4	360convert_AEP_RVP.cfg
	5(compact)	360convert_AEP_CISP.cfg

Date Saved: 2020-08-25

3 (Compact octahedron)	0	360convert_COHP_ERP.cfg
	1(3x2)	360convert_COHP_Cubemap3x2.cfg
	2	360convert_COHP_AEP.cfg
	3(compact)	360convert_COHP_COHP.cfg
	4	360convert_COHP_RVP.cfg
	5(compact)	360convert_COHP_CISP.cfg
5 (Compact icosahedron)	0	360convert_CISP_ERP.cfg
	1(3x2)	360convert_CISP_Cubemap3x2.cfg
	2	360convert_CISP_AEP.cfg
	3(compact)	360convert_CISP_COHP.cfg
	4	360convert_CISP_RVP.cfg
	5(compact)	360convert_CISP_CISP.cfg
7 (TSP)	0	360convert_TSP_ERP.cfg
8 (SSP)	0	360convert_SSP_ERP.cfg
	1(3x2)	360convert_SSP_Cubemap3x2.cfg
	2	360convert_SSP_AEP.cfg
	3(compact)	360convert_SSP_COHP.cfg
	4	360convert_SSP_RVP.cfg
	5(compact)	360convert_SSP_CISP.cfg
	7	360convert_SSP_TSP.cfg
9 (ACP)	0	360convert_ACP_ERP.cfg
	4	360convert_ACP_RVP.cfg
10 (RSP)	0	360convert_RSP_ERP.cfg
	4	360convert_RSP_RVP.cfg
11 (ECP)	0	360convert_ECP_ERP. cfg
	4	360convert_ECP_RVP. cfg
12 (EAC)	0	360convert_EAC_ERP. cfg
	4	360convert_EAC_RVP. cfg
12 (HEAC)	0	360convert_HEAC_ERP. cfg
	4	360convert_HEAC_RVP. cfg
13 (HEC)	0	360convert_HEC_ERP. cfg
	4	360convert_HEC_RVP. cfg
14 (Fisheye)	0	360convert_FISHEYE_ERP.cfg
15 (GCMP)	0	360convert_GCMP_ERP.cfg
	0 (viewing from spherical center)	360convert_GCMPSEI_ERP.cfg

	4	360convert_GCMP_RVP.cfg
	4 (viewing from spherical center)	360convert_GCMPSEI_ERP.cfg

3.2 Viewport generation

360-degree video is unique in that, unlike conventional 2D planar video, only a portion (i.e., the viewport) of the entire spherical video will be rendered and presented to the viewer. To correspond to this unique behavior, 360Lib provides the functionality to render images within a given viewport using rectilinear projection [7]. Once rendered, the 2D rectilinear viewport pictures can be viewed on conventional displays (TV, monitors) directly. To use this functionality, the user should set CodingGeometryType to 4 in the configuration file. 360Lib supports viewport generation from any of the seven projection formats: equirectangular, equal-area, cubemap, octahedron, icosahedron, truncated-square-projection, segmented sphere projection.

Three modes of viewport generation are supported:

- Fixed viewport mode: this mode generates a fixed viewport for the entire video sequence. The configuration file parameter "ViewPortSettings" is used to specify the viewport, for example, as follows:

ViewPortSettings   :   75.0   75.0   270.0   0.0

In this example, the horizontal and vertical field of view (FOV) are both set to be 75.0 degrees, the center of the viewport is set at 270.0 degrees in longitude and 0.0 degree in altitude.

- Dynamic viewport mode: this modes allows the user to specify the viewport on a frame-by-frame basis. To use this mode, an external viewport setting file needs to be provided via the configuration parameter "ViewPortFile". For example, "ViewPortFile" is set as follows:

ViewPortFile   :   viewport.txt

In viewport.txt (or any other filename), the frame-by-frame viewport setting should be arranged as follows:

0   75 75 15 5

4   75 75 45 5

8   75 75 60 5

…

Where each line specifies a frame number followed by the viewport setting for each video segment. The first line specifies "75 75 15 5" viewport setting for the frame period from POC 0 to POC 3, inclusive. The second line specifies "75 75 45 5" viewport setting for the second video segment from POC 4 to POC 7, inclusive. The last line specifies "75 75 60 5" viewport setting for the last video segment, from POC 8 to the end of the sequence. This dynamic viewport mode allows the user to render and write out a video sequence with dynamically changing viewport, thus simulating the viewing behavior of 360 video when the user changes the viewing angle by moving his/her head around wearing a HMD.

- Dynamic viewport with transition mode: this modes allows the user to specify the viewports at the beginning and the end of the video. The conversion software will automatically calculate the viewport position with linear interpolation for each frame. To use this mode, an external viewport setting file needs to be provided via the configuration parameter "DynamicViewPortFile". For example, "DynamicViewPortFile" is set as follows:

DynamicViewPortFile   :   **dynamic_viewport.txt**

In dynamic_viewport.txt (or any other filename), the parameter should be arranged as follows:

75.0 75.0   0 -45 -15   299 45 15

The first two parameters specify the horizontal and vertical FOV size. The third parameter is the POC of the picture at the beginning, and the following two parameters are yaw and pitch for the center of the viewport for that picture. The sixth parameter is the POC of the picture at the end, and the following two parameters are yaw and pitch for the center of the viewport for the picture.

The width and height of viewport are specified by parameter "CodingFaceWidth" and "CodingFaceHeight", respectively.

3.3 Encoding with 360Lib

The 360Lib can be integrated with the VTM (VTM-3.0) encoder. The readme file is provided as "./360Lib_README.txt". Example configuration files listed in Table 12 that can be used to perform encoding together with projection conversion are available in the directory "./cfg" if the source video is in ERP format. The common test condition is provided in [13].

Table 12. Example configuration files for encoding with different projection formats given ERP source video

Coding projection format	Configuration file
ERP without padding	encoder_360_ERP.cfg
ERP with padding	encoder_360_PERP.cfg
CMP (3x2 frame packing)	encoder_360_CMP.cfg
AEP	encoder_360_AEP.cfg
COHP	encoder_360_COHP.cfg
CISP	encoder_360_CISP.cfg
TSP	encoder_360_TSP.cfg
SSP vertical frame packing	encoder_360_SSP_vert.cfg
ACP	encoder_360_ACP.cfg
RSP	encoder_360_RSP.cfg
ECP	encoder_360_ECP.cfg
EAC	encoder_360_EAC.cfg
HEC	encoder_360_HEC.cfg
Fisheye	encoder_360_Fisheye.cfg
GCMP	encoder_360_GCMP.cfg
GCMP (viewing from spherical center)	encoder_360_GCMPSEI.cfg

The configuration parameter "SphereVideo" is used to indicate if the video needs projection conversion before encoding or not. If SphereVideo is equal to 1, then all projection conversion related parameters will be applied to calculate the projection format conversion output. Then, the converted frames are sent directly to encoder. If SphereVideo is 0, then the input video is encoded without any projection conversion.

SphereVideo   : 1

4 References

[1] Y. He, B. Vishwanath, X. Xiu, Y. Ye, "AHG8: InterDigital's projection format conversion tool," Joint Video Exploration Team of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, JVET-D0021, Oct. 2016, Chengdu, China.

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

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

[4] M. Zhou, "AHG8: A study on quality impact of line re-sampling rate in EAP," JVET-G0051, Jul. 2017, Turin, IT.

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

[6] 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.

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

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

[9] 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.

[10] 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, Mar. 2017, Hobart, AU.

[11] 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.

[12] 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," JVET-G0097, Jul. 2017, Turin, IT.

[13] P. Hanhart, J. Boyce, K. Choi, "JVET common test conditions and evaluation procedures for 360° video," Joint Video Exploration Team of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, JVET-L1012, Oct. 2018.

[14] M. Coban, G. Van der Auwera, M. Karczewicz, "AHG8: Adjusted cubemap projection for 360-degree video," JVET-F0025, Mar. 2017, Hobart, AU.

[15] A. Abbas, D. Newman, "AHG8: Rotated Sphere Projection for 360 Video," JVET-F0036, Mar. 2017, Hobart, AU.

[16] J. Boyce, Z. Deng, "EE4: Padded ERP (PERP) projection format," JVET-G0098, Jul. 2017, Turin, IT.

[17] G. Van der Auwera, M. Coban, M. Karczewicz, "AHG8: ECP with padding for 360-degree video," JVET-G0074, Jul. 2017, Turin, IT.

[18] M. Zhou, "AHG8: A study on Equi-Angular Cubemap projection (EAC)," JVET-G0056, Jul. 2017, Turin, IT.

[19] A. Abbas and D. Newman, "AHG8: An Update on RSP Projection," JVET-H0056, Oct. 2017, Macau CN.

[20] Y.-H. Lee, J.-L. Lin, S.-K. Chang, C.-C. Ju (MediaTek), "CE13: Modified Cubemap Projection in JVET-J0019 (Test 5)," JVET-K0131, Jul. 2018, Ljubljana, SI.

[21] H.-M. Oh, "SW implementation of fisheye projection format", JCTVC-AK0031, Oct. 2019, Geneva, CH.

[22] Y.-H. Lee, J.-L. Lin, Y.-J. Chen, C.-C. Ju, J. Boyce, M. Dmitrichenko, "AHG6/AHG17: Generalized cubemap projection syntax for 360-degree videos", Joint Video Experts Team of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, JVET-P0597, Oct. 2019, Geneva, CH.

[23] Y.-H. Lee, J.-L. Lin, Y.-J. Chen, C.-C. Ju, "AHG6/AHG9: Signalling guard band type for generalized cubemap projection", Joint Video Exploration Team of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, JVET-Q0343, Jan. 2020, Brussels, Belgium.

[24] Y. Wang, Y. He, L. Zhang, "SW Support of 360 SEI Messages", Joint Video Exploration Team of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, JVET-S0257, June. 2020, Teleconference.