IMPROVED MOTION COMPENSATION FOR 360° VIDEO PROJECTED TO POLYTOPES

Johannes Sauer, Jens Schneider, Mathias Wien

Institut für Nachrichtentechnik, RWTH Aachen University, Germany

ABSTRACT

360° video consists of images capturing the complete scene as seen from a single point in any direction. In contrast to conventional images, 360° images cannot be natively represented in a planar fashion. Consequently, video created from a sequence of 360° images suffers from geometric distortions. These are caused by mapping of the images from a sphere in 3D space to a planar image.

This paper introduces an approach to improve motion compensation for 360° video by compensating the geometric distortions resulting from projection to the faces of a polytope. Each face is extended with content projected from faces connected to it. This provides better prediction candidates for motion compensation at face borders.

Here, the applicability of the proposed method is demonstrated for 360° video projected to the faces of a cube. An evaluation of the method on two sets of sequences, static camera and non-static camera, verifies its effectiveness. It works especially well on non-static camera sequences, which have plenty of global motion. For the non-static camera sequences in the used test set rate savings of -2.14% were achieved.

Index Terms— video compression, 360° video, motion compensation

1. INTRODUCTION

Recently, 360° video has gained attention in the video coding research community [1, 2]. 360° video is now being investigated in an Ad-hoc-group of the Joint Video Exploration Team (JVET) [3]. It is a new type of content for video sequences. 360° content has challenges specific to its kind, which have to be addressed.

Motion compensation is a method of inter prediction where a block from another frame is copied in order to predict a block in the frame being encoded. Inter prediction relies on the assumption that only small displacements occur between subsequent frames, which can be represented by translation. A translational motion model is part of all video coding standards which are based on the hybrid video coding scheme [4].

This assumption is violated in 360° video, because it can not be represented in a planar image without introducing geometrical distortions. The intensity of these distortion depends on the location in the image and on the projection used for the representation of the 360° video. Consequently, it has to be taken into account how the geometrical distortions affect the source and destination block during prediction. Further, 360° video features rotational symmetry. Content which moves out of the frame at a border appears again at another border, instead of vanishing completely. This can also be exploited in order to improve motion compensation.

This paper proposes a method to perform motion compensation correctly for 360° video which has been projected to the faces of a convex polytope. For this case the geometric distortions occur only at the borders of polytope faces. In this approach, blocks crossing borders of polytope faces are corrected. The content of the connected face is projected to the image plane of the face of the predicted block. The method was implemented and tested for the case of 360° video projected to the faces of a cube. The reported results show that the proposed method yields considerable coding gains, thus proving that it treats the geometrical distortions correctly during motion compensation. Because of this, the method has been proposed to the 360° Ad-hoc-group of JVET [5].

Two other similar approaches were reported by Ma et al. [6] and He et al. [7]. They report similar results to those of the proposed method. However, they differ by the derivation of the face extension method, since they are both based on direct calculation of corresponding points on the other faces using line equations. The respective derivations are only valid for 360° content projected to a cube. They can not be readily applied for 360° content projected to other convex polytopes.

The rest of this paper is organized as follows. In Section 2, different representation formats of 360° video are discussed. Section 3 introduces a method for correcting distortions between two faces of a convex polytope. In Section 4, the method is then applied for the case of the cube format. Section 5 discusses the integration of the method into a hybrid video coding scheme. Results for different 360° sequences are reported in Section 6. Section 7 concludes this paper.

2. 360° VIDEO REPRESENTATION

There are multiple flat representations in discussion, all based on different types of projection from the sphere to a plane [8, 9]. Among the different projections are: equirectangular projection, cube projection, octahedron projection, icosahedron projection, segmented sphere projection and truncated square

978-1-5090-6067-2/17/$31.00 ©2017 IEEE

pyramid projection. The different representations can be classified into two groups according to how distortions affect them. In the first group geometrical distortions are distributed over the whole image (equirectangular projection, segmented sphere projection). The representations of the second group are derived from convex polytopes. Here, distortions occur across face borders, but there are no geometrical distortions inside of faces. In the following, the relevant projection formats are described.

2.1. Equirectangular representation

The whole sphere is projected to a rectangle. This is similar to the projection of the globe to a map. Distortions are distributed over whole image. The distortions are smallest where the equator is projected to the image. Accordingly, there are extreme distortions at top and bottom of the image, where the poles are projected to lines. Currently, equirectangular representation is the common source representation. It is also a potential representation for coding.

2.2. Convex polytopes

Each face of the convex polytope is the projection of part of the sphere. A face looks like a usual 2D image, with no visible geometrical distortions. The relationship between the image of a face and the 3D world is that of an image captured by a pinhole camera. The geometrical distortions in 360° video represented in this way occur at the borders of faces. The intensity of the distortion depends of the angle between the faces. The distortions decrease for flatter angles, as the resulting polytope is a better approximation of a sphere.

Cube The sphere is mapped to the faces of a cube. Since the faces of a cube are quadratic it can be easily split into quadratic blocks. This makes this representation suitable for video coding, as state-of-the-art video codecs also work on pictures split into quadratic blocks. However, there are geometrical distortions at borders of faces, see e.g. Figure 1.

Different layouts are possible for arranging the cube faces into a rectangular frame, particularly the cube 4x3 layout and the compact cube 3x2 layout [10]. The cube 4x3 layout can be obtained by unfolding a cube, the faces form a connected region shaped like a lying cross. However, there are empty faces in this arrangement, which introduce an overhead for coding. The compact cube 3x2 layout avoids this problem. Here the six faces of the cube are arranged in two rows of three faces each. Each row is a connected region on the cube. The resulting video has no empty regions. For the rest of this paper we will assume the compact cube 3x2 layout when referring to 360° video in cube representation.

Icosahedron The sphere is mapped to the faces of an icosahedron. Since the angle between faces is wider than for the cube the geometrical distortions at borders are not as strong. Due to the triangular nature of the icosahedron faces, a face can not be split into quadratic blocks. This is a problem for coding, since it introduces face edges inside single coding blocks. As lines bend at face edges, this is a problem for directional prediction used in intra coding. It is also a problem for inter prediction, since motion also changes at these face edges.

As for the cube there are different arrangements of the icosahedron faces into a rectangular picture, an arrangement based on unfolding and its compaction [11].

3. POLYTOPE FACE EXTENSION

Lines which are straight appear to bend at the border between two faces, see Figure 1. This is a problem for video coding, which causes motion compensation to fail if there is motion across the edges of a face.

(a) Face A (b) Face B

Fig. 1: A cube face and its neighbor face. There are geometrical distortions when crossing the border between face A and face B: Straight lines bend at the border.

This issue can be addressed by extending the area of a face with data from the faces next to it. The neighbors[1] of a face are projected to the same plane as the face which shall be extended. Thus, a larger face without major geometrical distortions is created. This is illustrated in Figure 2.

The projections of the neighboring faces to the current face can be expressed as homographies. In the following, the homography projecting the data of a single neighbor face B to the current face A will be derived. The different faces are treated as images captured by pin-hole cameras. They all share the same camera center C, which is in the center of the object (also the center of the sphere on which the stitching was performed). Thus, a point of face B can be projected to face A without requiring knowledge of the distance of the point from the camera center. $\mathbf{K}_\mathrm{A}$ and $\mathbf{K}_\mathrm{B}$ are the camera

[1]Neighbors are faces which are connected with each other by a single edge in the 3D polytope.

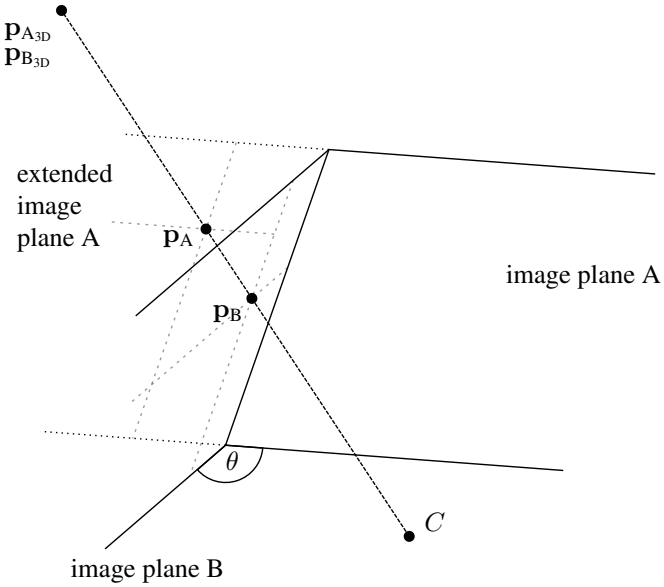

Fig. 2: Extension of face A with data from face B. Both image planes of the faces A and B share the same camera center C.

calibration matrices relating to the faces A and B: they express the relation between 3D points and their projection onto face A and B. The form of a camera calibration matrix $\mathbf{K}$ for the pinhole model is [12]:

$$\mathbf{K} = \begin{pmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{pmatrix} \tag{1}$$

Here, the focal length f is the distance of the camera center from the image plane and the principal point $(p_x, p_y)^T$ is the projection of the camera center onto the image plane along a line perpendicular to the image plane.

Then, the homography transferring the image on face B to the image plane of face A consists of the following steps:

- The point $\mathbf{p}_B$ on the neighbor face B is projected to a 3D point $\mathbf{p}_{B_{3D}}$. The distance Z of this point from the camera is arbitrary. Since all faces share the same camera center, points can be projected between faces without knowledge of the distance from the camera.

$$\mathbf{p}_{B_{3D}} = \begin{pmatrix} & \mathbf{K}_B^{-1} & \\ 0 & 0 & Z^{-1} \end{pmatrix} \mathbf{p}_B \tag{2}$$

- The coordinate system is changed from the image plane of the neighbor face B to the image plane of the current face A. This is a rotation around the shared edge of the faces. It depends only on the angle θ between the faces. The 3D point $\mathbf{p}_{B_{3D}}$ can now be expressed as a 3D point in respect to face A, $\mathbf{p}_{A_{3D}}$:

$$\mathbf{p}_{A_{3D}} = \begin{pmatrix} \mathbf{R}(\theta) & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix} \mathbf{p}_{B_{3D}} \tag{3}$$

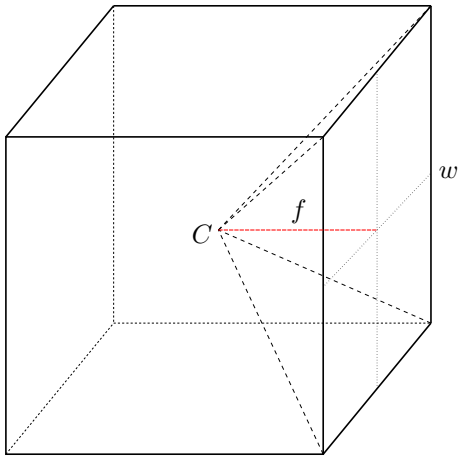

Fig. 3: The relationship between focal length f (red) and face width w is defined by the geometry of the cube. C is the center of the cube and the camera center of all faces.

- The 3D point $\mathbf{p}_{A_{3D}}$ is projected to a point $\mathbf{p}_A$ on face A. Note that this removes Z from the homography:

$$\mathbf{p}_A = \begin{pmatrix} \mathbf{K}_A & \mathbf{0} \end{pmatrix} \mathbf{p}_{A_{3D}} \tag{4}$$

The complete homography $\mathbf{H}_{B2A}$ can be derived by applying all steps together:

$$\begin{aligned} \mathbf{H}_{B2A} &= \begin{pmatrix} \mathbf{K}_A & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{R}(\theta) & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} & \mathbf{K}_B^{-1} & \\ 0 & 0 & Z^{-1} \end{pmatrix} \\ &= \mathbf{K}_A \mathbf{R}(\theta) \mathbf{K}_B^{-1} \end{aligned} \tag{5}$$

4. CUBE FACE EXTENSION

While the method of face extension was derived for an arbitrary polytope, the remainder of this paper will focus on the application of the method for the case of 360° video, which was mapped to the faces of a cube. Without loss of generality, we assume that the resolutions of the different cube faces are the same. Further, the principal point of the image plane associated with a face shall lie in the center of the respective faces. Then the camera calibration matrices are:

$$\mathbf{K}_A = \mathbf{K}_B = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{6}$$

By geometrical considerations, the value of f is determined to be $f = \frac{w}{2}$, where w is the width of a face (Figure 3).

For the derivation, we assume that face A is to the left of face B. Then the transformation aligning them is a rotation around the y-axis by 90°. Thus the resulting homography for

the projection of face B to face A is:

$$\begin{aligned}\mathbf{H}_{\text{B2A}} &= \mathbf{K}_{\text{A}}\mathbf{R}_{\mathbf{y}}(90^\circ)\mathbf{K}_{\text{B}}^{-1} \\ &= \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{pmatrix} \begin{pmatrix} f^{-1} & 0 & 0 \\ 0 & f^{-1} & 0 \\ 0 & 0 & 1 \end{pmatrix} \\ &= f^{-1} \begin{pmatrix} 0 & 0 & f^2 \\ 0 & f & 0 \\ -1 & 0 & 0 \end{pmatrix} \end{aligned} \tag{7}$$

Note that the scaling factor f^{-1} can been removed, since homogeneous coordinates are defined only up to scale. This avoids division operations until coordinate normalization.

The result of the application of the method is illustrated in Figure 4. As can be seen, the method is able to correct the geometrical distortions when crossing from one face to the next, compare the example in Figure 1. In the video in cube representation, there are kinks of lines at the borders between faces, even though these lines are straight in 3D space. The extended face does not have these distortions. However, it should be noted that the sample density in the extended region decreases with increasing distance to the border.

The homographies for the other connected faces can be derived accordingly. They are all combined in order to generate a face which is extended at all sides.

Fig. 4: Example: Extension of Face A (Figure 1a) with data projected from face B (Figure 1b).

5. INTEGRATION INTO A HYBRID VIDEO CODING SCHEME

We assume that the input for the coder is a 360° video sequence arranged in the compact cube 3x2 layout [10]. The method of face extension is used to modify the reference pictures used during motion compensation. The method is performed for each frame in the reference pictures list, creating six additional references for the different faces. This is necessary since the extended faces overlap each other. It is not possible to store them all in the same reference picture.

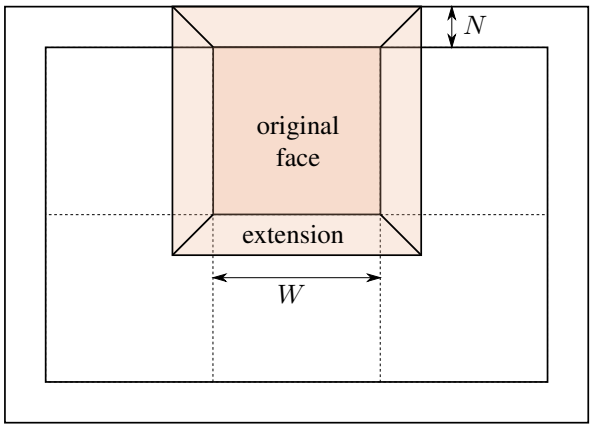

Fig. 5: A reference picture is partitioned into different faces, as defined by the compact 3x2 cube format [10]. Orange: the extended reference picture used for the cube face in the top-middle location.

The layout of each of the additional references is as follows (Figure 5). The face in the original reference picture is collocated with the face in the additional reference picture. A region around the face is extended with data projected from the connected faces. The size of this extension is configurable. Here it is chosen to be the same as is used for picture border extension for motion compensation. Thus, the rotational symmetry of 360° video is taken into account: content moving out of the image will appear in the extend region of another face. The remaining area of the additional reference picture is empty for easy implementation purposes. This is a first, straightforward implementation. Note that more efficient implementations can be made by storing only the additional areas of the extended faces.

The face, which is accessed, can be derived by the position of the block for which motion compensation is performed. In all tested video sequences, the face width and height were multiples of 8. This is also the minimal size of a coding unit (CU) in High Efficiency Video Coding (HEVC). Hence, every CU lies exactly in one face. The location of a CU is known to both en- and decoder. Thus the additional reference picture which has to be used for a given CU can be derived based on its location and this method requires no additional signaling.

6. CODING RESULTS

Results are reported for an implementation into the reference software for HEVC, version HM 16.7 [13]. For the implementation of the proposed projection, OpenCV [14] was used. For the generation of the extended face regions linear inter-

Sequence	Provider	Resolution	Frame rate	Length	Bit depth	Camera type
abyss great shark	GoPro	2880x1920	30	300	8	static
paramotor training	GoPro	2880x1920	25	250	8	static
timelapse building	GoPro	2880x1920	25	250	8	static
PoleVault_le	CTC	2880x1920	30	300	10	static
Train_le	CTC	3552x2368	60	600	10	static
KiteFlite	CTC	3552x2368	30	300	10	static
Harbor	CTC	3552x2368	30	300	10	static
bicyclist	GoPro	2880x1920	25	250	8	non-static
glacier	GoPro	2880x1920	24	240	8	non-static
AerialCity	CTC	2880x1920	30	300	10	non-static
DrivingInCity	CTC	2880x1920	30	300	10	non-static
DrivingInCountry	CTC	2880x1920	30	300	10	non-static
SkateboardInLot	CTC	3552x2368	30	300	10	non-static
ChairliftRide	CTC	3552x2368	30	300	10	non-static

Table 1: Overview of the used 360° sequences.

polation has been applied. The implementation replaces one reference picture by six reference pictures.

The proposed method was tested on the 360° sequences listed in Table 1. In total, a set of 14 sequences was used. Five of the sequences were provided by GoPro [15], the others were taken from the JVET common test conditions and evaluation procedures for 360 video (CTC360) [16]. Half of the sequences feature static cameras, the others non-static cameras. The chosen coding scenario was a random access configuration, while the set of tested quantization parameters was $QP \in 22, 27, 32, 37$. The sequences from the CTC360 were converted to the compact 3x2 cube format prior to coding. The sequences from GoPro already were in this format. The table shows the details of the converted sequences.

The coding performance of the method was measured using the Bjøntegaard Delta (BD) [17]. The BD-statistics are shown in Table 2, which is split according to sequences with static cameras and sequences with non-static cameras. The reference for the BD-statistics was the coding performance of the reference software for HEVC, version HM 16.7.

The method increases the rate savings for the sequences with static cameras on average by $0.06\,\%$, without affecting the average PSNR. For the sequences featuring non-static cameras the method clearly outperforms the original HEVC encoder. Here, the rate savings increase by $2.14\,\%$, while the average PSNR increases by 0.07 dB.

Since the method applies for motion compensation, it works well on sequences with a lot of global motion. This specifically is the case for the sequences featuring non-static cameras, for which the best gains are observed.

For a few sequences there are losses. This indicates that for those sequences, the original reference pictures provide a better predictor for regions at face borders than the extended reference pictures.

There are two main aspects to consider in terms of the complexity of this approach, memory requirement and computational complexity.

	Sequence	Rate	PSNR
static	abyss great shark	-0.45 %	0.02 dB
	paramotor training	-0.24 %	0.01 dB
	timelapse building	-1.45 %	0.05 dB
	PoleVault le	-0.20 %	0.01 dB
	Train le	0.17 %	-0.01 dB
	KiteFlite	0.24 %	-0.01 dB
	Harbor	1.49 %	-0.05 dB
non-static	bicyclist	-1.28 %	0.06 dB
	glacier	-4.60 %	0.19 dB
	AerialCity	-1.63 %	0.03 dB
	DrivingInCity	-0.79 %	0.02 dB
	DrivingInCountry	-3.05 %	0.10 dB
	SkateboardInLot	-1.20 %	0.04 dB
	ChairliftRide	-2.40 %	0.08 dB
	Average static	-0.06 %	0.00 dB
	Average non-static	-2.14 %	0.07 dB

Table 2: Bjøntegaard Delta-Statistics of cube face extension.

The current implementation of the method uses seven times the memory of HM 16.7, since for each reference picture six additional reference pictures are created. However, the implementation focused on simplicity, not efficiency. Large portions of the additional reference pictures are empty. The only additional memory actually required is that of the area A_{Ext} of the extended regions. Given the width W of a face and the size N of the extended area it can be calculated as:

$$A_{\text{Ext}} = 6(4NW - 4N^2) \tag{8}$$

Since N is typically small compared to W the introduced overhead is very small, the area of whole frame without padding is $6W^2$.

The computational complexity of the method is low. It is sufficient to generate the extended regions once for each reference picture. Apart from this the number of computations required for motion search remains the same.

7. CONCLUSION

In this paper a method was presented which improves the performance of motion compensation in 360° video sequences. This is achieved by providing geometrically corrected reference pictures, which are used for the prediction of blocks at the borders of faces. Further, the method also accounts for the rotational symmetry of 360° content. The method was first derived for an arbitrary convex polytope, and its effectiveness has been shown for the cube representation of 360° content. Hence, it can be easily extended to other formats, which project the 360° video to a convex polytope. Since the original reference picture are replaced, the method avoids additional signaling and also has only minor impact on the complexity of en- and decoder.

8. REFERENCES

[1] M. Budagavi, J. Furton, G. Jin, et al., "360 degrees video coding using region adaptive smoothing," in *2015 IEEE International Conference on Image Processing (ICIP)*, Sept 2015, pp. 750–754.

[2] G. Jin, A. Saxena, and M. Budagavi, "Motion estimation and compensation for fisheye warped video," in *2015 IEEE International Conference on Image Processing (ICIP)*, Sept 2015, pp. 2751–2755.

[3] Jill Boyce, Elena Alshina, Geert van der Auwera, et al., "JVET AHG report: 360 video test conditions (AHG8)," Doc. JVET-D0008, Joint Video Exploration Team (on Future Video coding) of ITU-T VCEG and ISO/IEC MPEG, Chengdu, CN, 4th meeting, Oct. 2016.

[4] Mathias Wien, *High Efficiency Video Coding – Coding Tools and Specification*, Springer, Berlin, Heidelberg, Sept. 2014.

[5] Johannes Sauer and Mathias Wien, "Geometry correction for motion compensation of planar-projected 360VR video," Doc. JVET-D0067, Joint Video Exploration Team (on Future Video coding) of ITU-T VCEG and ISO/IEC MPEG, Chengdu, CN, 4th meeting, Oct. 2016.

[6] Xiang Ma, Haitao Yang, and Zhijie Zhao, "Co-projection-plane based motion compensated prediction for cubic format VR content," Doc. JVET-D0061, Joint Video Exploration Team (on Future Video coding) of ITU-T VCEG and ISO/IEC MPEG, Chengdu, CN, 4th meeting, Oct. 2016.

[7] Yuwen He, Yan Ye, Philippe Hanhart, et al., "Geometry padding for 360 video coding," Doc. JVET-D0075, Joint Video Exploration Team (on Future Video coding) of ITU-T VCEG and ISO/IEC MPEG, Chengdu, CN, 4th meeting, Oct. 2016.

[8] Jill Boyce, "BoG Report on 360 Video," Doc. JVET-D0188, Joint Video Exploration Team (on Future Video coding) of ITU-T VCEG and ISO/IEC MPEG, Chengdu, CN, 4th meeting, Oct. 2016.

[9] Yule Sun, Ang Lu, and Lu Yu, "AHG8: A study on the influence of different projection schemes," Doc. JVET-D0068, Joint Video Exploration Team (on Future Video coding) of ITU-T VCEG and ISO/IEC MPEG, Chengdu, CN, 4th meeting, Oct. 2016.

[10] Minhua Zhou, "AHG8: A study on JEM3.0 compression efficiency on 360 video content," Doc. JVET-D0024, Joint Video Exploration Team (on Future Video coding) of ITU-T VCEG and ISO/IEC MPEG, Chengdu, CN, 4th meeting, Oct. 2016.

[11] Minhua Zhou, "AHG8: A study on compression efficiency of icosahedral projection," Doc. JVET-D0023, Joint Video Exploration Team (on Future Video coding) of ITU-T VCEG and ISO/IEC MPEG, Chengdu, CN, 4th meeting, Oct. 2016.

[12] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[13] "HEVC Test Model, version 16.7," `https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-16.7`, 2016.

[14] Itseez, "Open source computer vision library," `https://github.com/itseez/opencv`, 2015.

[15] Adeel Abbas, "GoPro test sequences for virtual reality video coding," Doc. JVET-C0021, Joint Video Exploration Team (on Future Video coding) of ITU-T VCEG and ISO/IEC MPEG, Geneva, CH, 3rd meeting, May 2016.

[16] Jill Boyce, Elena Alshina, et al., "JVET common test conditions and evaluation procedures for 360° video," Doc. JVET-D1030, Joint Video Exploration Team (on Future Video coding) of ITU-T VCEG and ISO/IEC MPEG, Chengdu, CN, 4th meeting, Oct. 2016.

[17] Gisle Bjøntegaard, "Calculation of average PSNR differences between RD-curves," Tech. Rep. Doc. VCEG-M33, ITU-T SG16/Q6 VCEG, Austin, USA, 2001.