

Title: **GoPro test sequences for Virtual Reality video coding**

Status: Input Document to JVET

Purpose: Proposal

Author(s) or

Contact(s): Adeel Abbas

Email: aabbas@gopro.com

Source: GoPro

Abstract

This contribution provides new Virtual Reality (VR) video sequences for the Future Video Coding Standardization activity. VR video content is rapidly gaining in popularity and since it is a natural progression of video, it is hoped that a future video codec will be more efficient at compression of this type of content.

Introduction

Virtual Reality video imposes enormous demands on codecs and in almost all cases, using existing codecs makes deployment of a high-quality solution impossible. The most common use case for VR and 360-degree video content consumption is that a viewer is looking at a small window (sometimes called viewport) inside an image that represents data captured from all sides. Viewer could be watching this video on a smartphone app, for instance using GoPro VR App (<https://vr.gopro.com>). Viewer may also be watching this content on a head-mounted display (HMD) like Oculus Rift or Gear VR.

The viewport size is usually small (e.g. HD) but the video resolution corresponding to all sides can be significantly more (e.g. 8K). Delivery and decoding of an 8K video to a mobile device is impractical in terms of latency, bandwidth and computational resources. As a result, one option is to stream just the viewport window to the mobile device. This seems workable at first, but as viewer moves viewport window (by either pinching or physically moving the device), server has to stream an intra frame corresponding to the updated viewport. In addition, the device may have to decode all prior reference frames corresponding to the updated viewport. And all of this has to happen with very low latency, especially on HMD devices because viewers can experience fatigue. As a result of all this, there is a need for more efficient compression of VR content, to allow people to experience VR in high resolution, with low latency and using most battery friendly algorithms.

Proposed test sequences

Stitched sequences are provided in the following two formats:

1.1 Equirectangular Format

The equirectangular projection is a simple projection where 360-degree image data looks like a world map. The pixels that lie on the equator are stretched minimally (or not stretched at all), and as we go towards to poles, pixels get stretched horizontally, with poles being most distorted. This non-uniform distortion makes pixel-wise operations like filtering, subsampling and compression

very difficult. As a result, it was suggested recently that other projections like cubic projection can make video data better suited for compression [2].

Following sequences are provided in equirectangular format:

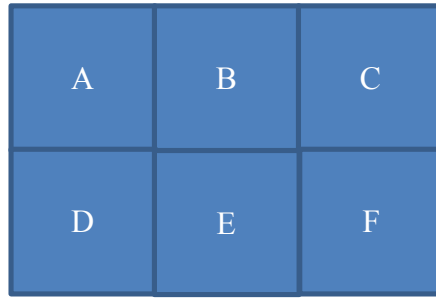
Sequence	Resolution	# Frames	FPS	Duration
abyss_great_shark_vr_30p_3840x1920.yuv	3840x1920	1145	30	38s
ballooning_vr_25p_3840x2160.yuv	3840x2160	1501	25	60s
bicyclist_vr_25p_3840x1920.yuv	3840x1920	1500	25	60s
glacier_vr_24p_3840x1920.yuv	3840x1920	1440	24	60s
paramotor_training_vr_25p_3840x1920.yuv	3840x1920	1279	25	51s
skateboarding_vr_60p_4096x2048.yuv	4096x2048	1372	60	22s
ski_downhill_vr_30p_3840x2160.yuv	3840x2160	1799	30	60s
timelapse_basejump_vr_25p_3840x1920.yuv	3840x1920	330	25	13s
timelapse_building_vr_25p_3840x1920.yuv	3840x1920	327	25	13s

An example frame in equirectangular format is shown below:



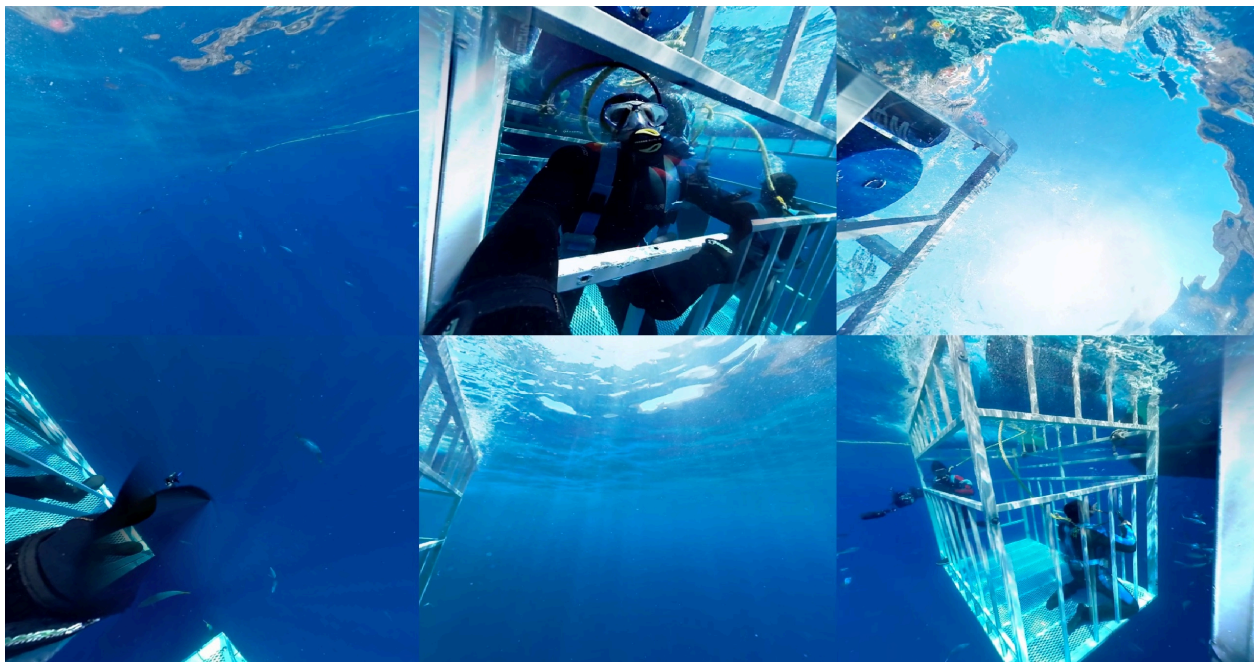
1.2 Cubic Format

It was reported in [2] that transforming equirectangular layout into cubic layout is better because none of the faces in a cubic layout have distortion. As a result, a bitrate savings of up to 25% (compared to equirectangular) was reported. A simple conversion tool [3] was shared that transforms video from equirectangular layout to cubic layout. The layout generated by this tool outputs six faces of the cube in the following format:

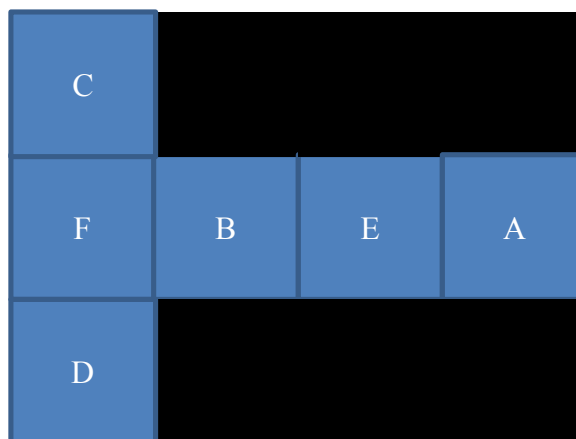


We will refer to this layout as Cubic-3x2 layout. C is the top face of the cube, D is the bottom face, and A, B, E and F are side faces. Size of each cube face is 720x720 pixels.

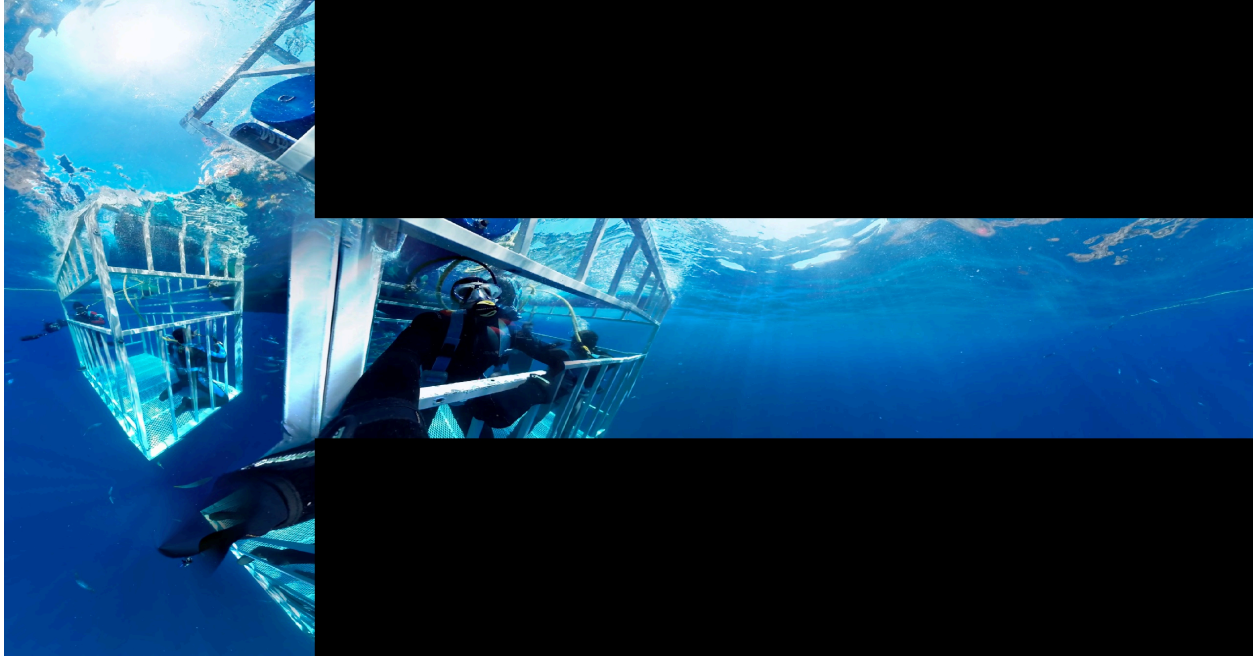
An example image of this format is shown below:



Since the image in the above format does not have any continuity across cube boundaries, we felt the need to rearrange cube faces in a layout that preserves continuity, as shown below:



We are calling this layout as Cubic-4x3 layout. An example image in this layout is shown below:



The following sequences are being provided in Cubic-4x3 layout:

Sequence	Resolution	# Frames	FPS	Duration
abyss_great_shark_vr_30p_2880x2160.yuv	2880x2160	1145	30	38s
ballooning_vr_25p_2880x2160.yuv	2880x2160	1501	25	60s
bicyclist_vr_25p_2880x2160.yuv	2880x2160	1500	25	60s
glacier_vr_24p_2880x2160.yuv	2880x2160	1440	24	60s
paramotor_training_vr_25p_2880x2160.yuv	2880x2160	1279	25	51s
skateboarding_vr_60p_2880x2160.yuv	2880x2160	1372	60	22s
ski_downhill_vr_30p_2880x2160.yuv	2880x2160	1799	30	60s
timelapse_basejump_vr_25p_2880x2160.yuv	2880x2160	330	25	13s
timelapse_building_vr_25p_2880x2160.yuv	2880x2160	327	25	13s

We feel that majority of the tools in existing video compression standards like HEVC and AVC will work well in compressing images in Cube-4x3 format. Ideally, the areas marked in black region will not be encoded and since there is continuity across faces, we expect it to perform better (when compared against Cube-3x2) in a single tile or slice formulation.

It is important to clarify that GoPro is not predisposed to Cubic-4x3 format. Cubic-4x3 was selected because it appears to be more compression friendly than Cubic-3x2 format. Experts are encouraged to use the tool in [3] on equirectangular sequences in order to generate any format that they choose.

Compression Performance of Cube-3x2 vs Cube-4x3

Cube-4x3 is expected to provide bitrate savings over Cube-3x2 because of motion and content continuity. We ran a simple test of encoding frames 200-499 of abyss_great_shark_vr sequence in Cube-4x3 and Cube-3x2 formats. Default RA configuration at QP 32 was run with HM-16.9.

Since Cube-4x3 format has a lot of black areas, it is important not to include black areas in PSNR calculation. It is also important to use the same PSNR algorithm for both the formats. Hence, we modified the PSNR computation of a frame to be average PSNR of all 6 faces.

Format	Cube-3x2	Cube-4x3
Sequence	abyss_great_shark_vr_30p_2160x1440.yuv	abyss_great_shark_vr_30p_2880x2160.yuv
RATE	4856.6792	4844.8728
PSNRY	36.760	36.764
PSNRU	42.423	42.474
PSNRV	39.871	39.892

As expected, PSNR values of both sequences are very similar. Cube-4x3 appears to take slightly lesser bitrate than Cube-3x2, although it is still encoding all the black regions and has twice the number of pixels. It appears that there was some bitrate saved as a result of continuity, however those bits were spent on encoding all the black regions. It would be interesting to make following changes to the reference software to observe further bitrate savings:

- 1) Do not encode black areas.
- 2) Fill black area with rotated top and bottom faces to align with side faces. Do not encode these regions, but use this new picture for motion estimation (and motion compensation for the decoder). This way motion of an object going from a side face to top or bottom face will be captured. Extending this idea further, there is untapped continuity across leftmost side face and rightmost side face. Therefore, instead of using Cube-4x3 format, we can imagine using bigger reference frames in Cube-6x3 format where rightmost face is placed to the left of leftmost face (and vice versa).
- 3) Use rotated top and bottom faces for intra prediction as well.
- 4) If we carefully look at cube boundaries, it appears that Cubes do not perfectly align, rather there is small overlap (same object appears on both sides of the cube boundary). This overlap is expected to hurt compression performance and therefore, an Equirectangular to Cube-3x2 algorithm that improves on aligning cube edges is desired. It is worth mentioning that this misalignment is not noticeable inside a 360-degree viewer because perspective becomes different.

Copyright information

The proposed content remains a property of GoPro and Kolor. Content copyrights are as follows:

1. Abyss Great Shark: Copyright George Probst, GoPro and Kolor.
2. All other videos: Copyright GoPro and Kolor.

The following uses are allowed for the proposed sequences:

1. Can be published in technical papers, played at technology research and development events.
2. Can be used by Standards committees. (e.g., ITU, MPEG, VQEG,).

The following uses are not allowed for the proposed sequences:

1. Do not publish snapshots in product brochures.
2. Do not use video for marketing purposes
3. Do not redistribute video with a commercial product.
4. Do not use in television shows, commercials, or movies.

Sequences

abyss_great_shark_vr_30p_3840x1920.yuv



ballooning_vr_25p_3840x2160.yuv



bicyclist_vr_25p_3840x1920.yuv



glacier_vr_24p_3840x1920.yuv



paramotor_training_vr_25p_3840x1920.yuv



skateboarding_vr_60p_4096x2048.yuv



ski_downhill_vr_30p_3840x2160.yuv



timelapse_basejump_vr_25p_3840x1920.yuv



timelapse_building_vr_25p_3840x1920.yuv



Sequence Proxy Files

Uncompressed 8 bit YUV-420 sequences have been uploaded to MPEG content repository. We have also made available down-sampled proxy files for browsing and playback purposes, which can be played or downloaded from the following locations:

Equirectangular (391MB):

<https://www.dropbox.com/sh/90xqjqfncceihg/AABiYjKJwRKOUYsfLIKkZWRa?dl=0>

Cube-4x3 (179MB):

https://www.dropbox.com/sh/zmgvy844qjglwlfo/AADM_WSH-mON_EYP3hdD1vvna?dl=0

References

- [1] J. Ridge, M.M. Hannuksela, “Virtual reality requirements for future video coding” 53rd Meeting: 20–26 February 2016, San Diego, US (VCEG-BA07)
- [2] E. Kuzyakov and D. Pio “Facebook’s encoding techniques for 360 and VR video”. [Link](#)
- [3] Facebook’s equirectangular to cube map tool on GitHub. [Link](#)
- [4] FFmpeg, <https://www.ffmpeg.org/>
- [5] GoPro [Omni](#) Rig
- [6] “[N15452](#): Call for Test Materials for Future Video Coding Standardization“, Warszawa, 2015.