

**ISO/IEC JTC 1/SC 29/WG 11**

**Coding of moving pictures and audio**

**Convenorship: UNI (Italy)**

**ISO/IEC JTC 1/SC 29/WG 11 N18442**

**Document type: Approved WG 11 document**

**Title: Call for MPEG-I Visual Test Materials**

**Status: Approved**

**Date of document: 2019-04-12**

**Source: Video**

**Expected action:**

**No. of pages: 10**

**Email of convenor: leonardo@chiariglione.org**

**Committee URL: mpeg.chiariglione.org**

**INTERNATIONAL ORGANISATION FOR STANDARDISATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

**ISO/IEC JTC1/SC29/WG11**

**CODING OF MOVING PICTURES AND AUDIO**

**ISO/IEC JTC1/SC29/WG11 MPEG2019/N18442**

**March 2019, Geneva,** **Switzerland**

|  |  |
| --- | --- |
| **Source** | **Video** |
| **Status** | **Approved** |
| **Title** | **Call for MPEG-I Visual Test Materials** |
| **Editors** | Gauthier Lafruit, Renaud Doré, Gun Bang, Bart Kroon, Joel Jung |

# Introduction

During exploration experiments, MPEG has realized that there is a lack of test material for video experiments into MPEG-I Immersive Video [N17762], especially for 3DoF+, 6DoF and Dense Light Field. This document is actually an update of the former [N18163] in order to take multiple camera architectures into account, e.g. ERP, linear perspective camera arrays and lenslet cameras. Accordingly, all camera parameters are expressed in the same JSON format adopted during the 3DoF+ activities, cf. Annex.

We hereby solicit new test material for all categories, in particular using non-parallel camera setups. Since all technologies in MPEG-I Visual are based on multiple camera views and their depth maps, we recommend that not only the camera views, but also depth maps (measured or estimated) are provided by the proponents.

Content may be provided as computer-generated/synthetic **3D models of static and dynamic scenes**, as this material can be used for rendering various viewpoints with computer graphics techniques, creating video footage required in all experiments and comparative studies (e.g. PSNR quality evaluation with reference views).

MPEG is also calling for **natural content**, both **indoor and outdoor**, directly captured with camera rigs. Content with **objects close to the camera** are also warmly recommended, since this will challenge the proposed technologies for parallax rendering, e.g. heavy motion parallax for nearby foreground objects.

Content should be provided in any **image-based representation format**, e.g. **lenslet** format, or **Multiview + depth**. If content is generated in computer-generated/synthetic format, it should be rendered to create various image-based projections (and their depth maps) to be eligible as test material.

Please consult [N18167] for an overview of available test material.

# 3DoF+ content

For the investigation on 3DoF+ visual compression [N17612], video/image material in the following format is requested: Multi-view 360° texture + depth.

They should have the following characteristics:

Capture:

* Texture+depth 360° x 180° video/image from two or more simultaneously operating cameras in equirectangular format or in rectilinear, with possibly different viewing positions. By “camera” in this document, one understands either a full capture of the scene from a camera device or a portion of this scene, as explained in [N17612] fig A and B.
* The divergence of the multiple capture shall be enough in order to describe a 3DoF+ scene with at least 180° azimuth range.
* There are no “holes” meaning that all views together form a consistent set.
* A view shall always have a depth map, except when there is at least one other view with the exact same position that does have a depth map. Therefore, several videos may also be provided with the same viewing position.
* There is no limit in the number of videos. The number of cameras should define the 3DoF+ scene so that the operation of subjective tests is possible.
* A desirable content duration for subjective test should be 10 seconds. Mandatory content duration is 1 frame, e.g. just enough to apply objective metrics.
* fps shall be 30 or 60 or 90 fps.

Texture:

* Resolution shall be higher or equal to 4096x2024 and lower or equal to 8192×4096 (with exact 2:1 aspect ratio) for the full 360°x180° FoV. If the FoV is reduced, the resolution limits are adapted accordingly, so that angular resolution remains in the same range.
* Each capture can have a reduced Field of View (FoV) below 360° x 180°, e.g. 180° x 180° or 70° x 60° provided that:
	+ Radial distortion is removed from the camera views, with the optical axis passing through the center of the image as in a perfect pinhole camera model (possibly cropping the image to align the center of the image (Width /2, Height / 2) with the optical axis)
	+ Camera parameters are therefore simplified, cf. its JSON format in annex:
		- Intrinsics are reduced to a single parameter: a single focal length expressed in pixels
		- Extrinsics are represented by a position vector [x,y,z] and rotation vector [yaw, pitch, roll] as exemplified in section 3
	+ Depth maps are available in the same resolution and shape as the texture,
* All test material is progressively scanned and uses 4:2:0 colour sampling with 8 or 10 bits per sample per color component.
* The file format should be Planar YUV (ant not Packed YUV)
* Color primaries, sample range and transfer function used for RGB to YCbCr conversion (and needed for YCbCr to RGB conversion at display side) should follow ITU-R Recommendation BT 709. In particular, limited (219/255) sample range is expected for Y component, and limited (224/255) sample range is expected for Cb and Cr components. Also, the BT 709 non-linear transfer function with 0.45 exponent is expected.
* It is possible to provide content under a form of png or ppm / pgm or tiff, provided that they are convertible from RGB to YUV 4:2:0 exactly like the anchor, through a tool (like deriving from HDRTools) to be identified and made explicit by the content provider. In any case, the contents serving as references are those in YUV 4:2:0 only.
	+ For example, a png file provider could ask to use first a PNG-uncompressed TIFF converter:
		- *ffmpeg -vcodec png -i $1.png -compression\_algo raw $1.tiff*
	+ Then a TIFF to YUV converter:
		- *HDRConvert -f HDRConvertTiff8ToYuv420.cfg -p SourceFile=$1.tiff -p OutputFile=$1.yuv*

Depth:

* When a view has a corresponding depth map, then this depth map shall be provided by default as a raw monochrome stream with a default bit depth of 16-bit, at the resolution of the texture and in little-endian file format. The depth can be delivered with 10 bits only, which is a typical case when output from the360lib software (cf. [N17726]). In this case, this value of 10 should be indicated in the metadata .json file.
* When the format is omnidirectional, the depth is here meant to be the radius from the optical center of the omnidirectional camera.
	+ Depth values, in case there are, shall be coded as the normalized disparity, as described in section 3.2 of approved document [N16730] and adapted for radius dimension instead of z-distance. This requires the definition for each content of Rnear and Rfar values mentioned here below in the metadata section. The content provider is free to put the value Rfar value to infinite, which simplifies the depth relation. By convention, Rfar value put to 1000.0 is understood to mean infinite value.
		- $Depth=vmax\*(\frac{1}{R}-\frac{1}{Rfar})/(\frac{1}{Rnear}-\frac{1}{Rfar})$
		- $with vmax=65535$
* When the format is perspective, the depth is here meant to be z coordinate along the optical axis and from the optical center of that perspective camera
	+ Depth values, in case there are, shall be coded as the normalized disparity, as described in section 3.2 of approved document [N16730]. This requires the definition for each content of Znear and Zfar values mentioned here below in the metadata section. The content provider is free to put the value Zfar value to infinite, which simplifies the depth relation. By convention, Zfar value put to 1000.0 is understood to mean infinite value.
		- $Depth=vmax\*(\frac{1}{Z}-\frac{1}{Zfar})/(\frac{1}{Znear}-\frac{1}{Zfar})$
		- $with vmax=65535$
* In both previous cases, a reserved null value (0) of depth means non-available pixels, thus corresponding to a binary alpha mask channel.

Delivery Packaging:

* The default packaging is to have individual images of texture and of depth. Frame range does not have to start by frame index 1 but the sequence shall be continuous.
* There is an option to package textures into a raw texture video and depth into a raw depth video, and this should be indicated in the metadata .json file.
* The name of the texture file shall include for convenience all necessary information useful by a viewer of an elementary image, as described below, where *Camera\_name* can be any name relevant for the content producer, *width* and *height* are image resolution, *nb\_bits* is the number of bits for each color component, and *####* shall be replaced by the frame index.
	+ *[Camera name]\_[width]\_[height]\_420\_[nb\_bits] b\_####.yuv*
	+ Exemple: *camS1\_2048\_2048\_420\_10b\_1551.yuv*
* The name of the depth file – when present – shall include all necessary information useful by a viewer of an elementary depth image. The fields *Rnear* and *Rfar* are added in the format where integer part is separated from fractional part by “\_”, as illustrated here below where *Rnear* and *Rfar* are 0.5m and 25.0m respectively. Frame index range and camera name shall be aligned with the frame index range and camera or view name of the texture. The suffix is “.depth” for the file name of depth coded on 16bits, and becomes .yuv when the depth is coded on 10 bits. File name examples are given below in both cases:
	+ *[Camera name]\_[ width]\_[ height]\_[ Rnear]\_[ Rfar]\_ ####.depth*
	+ Exemple: *camS1\_2048\_2048\_0\_5\_25\_0\_1551.depth*
	+ *[Camera name]\_[ width]\_[ height]\_[ Rnear]\_[ Rfar]\_420\_10b.yuv*
	+ Exemple: *v1\_2048\_2048\_0\_5\_25\_0\_420\_10b.yuv*
* The texture+depth files should be zipped before uploaded to the server, so that it can be conveniently retrieved by one or a couple of zipped files downloads.

Metadata:

* Metadata shall be provided under the form of a JSON file that enlists in any order the following properties per video, the properties being the same for all frames of the content, and listed here below
* There is a block of general information:
	+ A general name linking this file to a given content
	+ The fps of the content (30 / 60 / 90)
	+ The total number of frames
	+ The center of the bounding box, expressed in OMAF referential
	+ An optional informative part can be added for clarity and not used by the 3DoF+ encoder.
* For each camera
	+ Camera Name of the file, as used in the file names described here above
	+ Video optional field (0: image, 1: video). When this field is not present, textures and depth inputs for that camera are made of images
	+ BitDepth optional field specifies the couple of 2 integer [ bit depth for texture, bit depth for depth ]. When absent, the default value is [8,16]
	+ Position of the center of the camera as three values [x, y, z] in meters in OMAF referential, as explained in figures 5.3 & 5.4 of [N17399], Orientation of the related camera [yaw, pitch, roll] expressed in degree and in OMAF referential, as explained in figures 5.3 & 5.4 of [3],
	+ If the view has a depth map or not (Boolean 1:true, 0:false),
	+ If the view is background or not (Boolean 1:true, 0:false)
	+ If so, the *Rnear* and *Rfar* values in meters. This “R” denomination should be understood here as generic: it is either a radius value if format is equirectangular or a z value if format is perspective. The *Rfar* value is permitted to be infinite. When the *Rfar* value is meant to be infinite, it will be arbitrarily written as 1000.0 value.
	+ Image/video resolution [width x height]
	+ Image/video horizontal and vertical range [Phimin ; Phimax] x [ Thetamin; Thetamax]. Full FoV is [-180; 180] x [-90;90]. These ranges are expressed in the camera referential. This field is only valid for ERP. When this field is present, the camera is understood to be ERP
	+ Focal expressed in pixel: this field is only valid for linear perspective camera. When this field is present, the camera is understood to be in linear perspective (pinhole camera).
* Format of real numbers is eee.ffff where eee and ffff are respectively integer and fractional part of any length.

# 6DoF Content

The goal of 6DoF is to enable virtual navigation in a scene with the user moving over a substantially large scene volume, i.e. the user takes multiple steps.

Recent contributions have brought new interesting, synthetic content reaching photorealism with perfect depth maps, e.g. [m41824], unfortunately mainly for static content. New - preferably natural - test sequences with good depth maps are requested to conduct the virtual view synthesis as described in EE\_Synthesis [N18166, N17761, N18070].

According to the MPEG-I PDTR [N17741], Omnidirectional 6DoF corresponds to a viewer being able to take multiple steps in the virtual scene captured with omni-directional cameras looking outward. Windowed 6DoF, however, requests content obtained in a convergent way, where the set of cameras are outside the scene that is intended to be rendered.

## Omnidirectional 6DoF content

We request content of any (virtual) multi-camera configuration with an omnidirectional nature (at least 180°) and nearby objects. The content should be provided as the YUV 420 or YUV 444 uncompressed video output of each camera in the rig. Examples of potentially suitable rigs can be Nokia Ozo, Facebook Surround 360, Vuze or any other custom-built rig. These camera systems typically have ultra-wide-angle or fisheye lenses. Also, a 3D synthetic model is allowed for omnidirectional 6DoF content. For these contents, complex objects and smooth motion should be provided. Intrinsic and extrinsic camera parameters have to be included, and it is highly recommended, but not strictly required, for a participant to supply depth maps or scripts to generate them. If the depth maps are provided, it should be in the YUV 400 video format.

In case of ultra-wide-angle (e.g. 140°) and fish eye lenses, we recommend receiving both the original and rectified video frames. For camera frames with a smaller FOV, rectified video frames are sufficient.

## Windowed 6DoF content

The content shall be provided together with corresponding depth maps (captured or estimated). The camera parameters (intrinsic and extrinsic) shall be provided only for texture if depth and texture have the same optical center, otherwise camera parameters shall be provided both for texture and depth. The content (texture and depth) shall be provided at least in YUV 420 8bits (to match MV-HEVC reference software input). Additional depth maps in YUV 420 16bits are welcome (to improve the reference navigation path for subjective testing). Depth information may be characterized using the depth formats described in [N16730]. The cameras should have a 1D arrangement with a large number of views, or a 2D array arrangement with a planar, cylindrical or spherical surface, or any other camera arrangement (even random-like camera placement), as long as the cameras remain outside the scene. MPEG encourages proponents to provide video material with framerates larger than 25 fps, and 10s length to facilitate subjective testing process.

# Dense Light fields

Dense Light Fields can be either captured by dense multi-camera arrays or a lenslet camera, e.g. Lytro or Raytrix. Currently, two formats (multiview and lenslet) are considered. These data formats can be converted from one to another. Such a conversion can be invertible or not, depending whether we have plenoptic 1.0 or 2.0 camera modes.

This activity aims to compare the coding performance of different potential representations of dense Light Field data (lenslet, multiview, etc). Currently, several test sequences are provided with dense multiview video, and only a few with lenslet cameras. Moreover, new lenslet content in plenoptic 1.0 and 2.0 format is solicited, with priority for the latter.

We hence encourage participants to provide MPEG test material for this activity with following contents, by MPEG127: (1) lenslet video, dense multiview video (2) multiview/lenslet converted from lenslet/multiview, (3) camera parameters, and (4) the conversion tool for conversion from lenslet to multiview video and vice versa.

# 3D synthetic models

In the case of photorealistic synthetic content, it is recommended that complementary to the rendered image data, also the 3D model itself plus any scripting to generate the required dataset is supplied. This enables other MPEG experts to render variants of the same scene. It is for instance possible to simulate a virtual camera array by placing multiple cameras in the scene. Preferably, scenes should include a timeline that allows for rendering of short movies with some dynamics.

The format of the 3D model should be in a suitable interchange format [N17252], or open source (e.g. Blender[[1]](#footnote-1)). Typical representations of 3D models are:

* Texture, mesh and lighting information,
* Procedural scenes (e.g. algorithms).

Content providers are asked to render their 3D scene in order to create Multiview image-based representations. Also, the depth maps should be created in the rendering process.

# Copyright notice

Content owners should provide a copyright notice along with the dataset to inform MPEG about copyright and usage restrictions.

# Contact:

* Bart Kroon, bart.kroon@philips.com
* Gauthier Lafruit, gauthier.lafruit@ulb.ac.be
* Mehrdad Teratani, panahpour@nuee.nagoya-u.ac.jp

# Informative annex: example of JSON files

The following file is an example of JSON metadata file with an ERP camera:

{

 "Version": "2.0",

 "Content\_name": "ClassroomVideo",

 "BoundingBox\_center": [0.0, 0.0, 0.0],

 "Fps": 30,

 "Frames\_number": 120,

 "Informative":

 {

 "BoundingBox\_size": 0.207846,

 "Document": "MPEG2018/M42415"

 },

 "cameras":

 [

 {

 "Name": "v0",

 "Projection": "Equirectangular",

 "Position": [0.0000000000, -0.0000000000, 0.0000000000],

 "Rotation": [0.0, 0.0, 0.0],

 "Depthmap": 1,

 "Background": 0,

 "Depth\_range": [0.8, 1000.0],

 "Resolution": [5000, 2000],

 "BitDepthColor": 10,

 "BitDepthDepth": 16,

 "ColorSpace": "YUV420",

 "DepthColorSpace": "YUV420",

 "Hor\_range": [-180.0, 180.0],

 "Ver\_range": [-90.0, 90.0],

 "Focal": [3000, 3000],

 "Principle\_point": [2500, 1000],

 },

 (…)

 ]

}

The following file is an example of a JSON metadata file with 3 pinhole / rectilinear cameras with texture and depth coming as 10 bits video:

{

 "Content\_name": " scene\_shot\_with\_rectilinear\_cameras",

 “BoundingBox\_center”: [0.0, 0.0, 1.65],

 "Fps": 30,

 "Frames\_number": 300,

 "Informative":

 {

 "BoundingBox\_size": 0.5,

 },

 "cameras":

 [

 {

 "Name":" V1”,

"Projection": "Perspective",

 “Video”: 1,

 "BitDepthColor": 10,

 "BitDepthDepth": 10,

 "ColorSpace": "YUV420",

 "DepthColorSpace": "YUV420",

 "Position": [0.3, 0.4, 1.65],

 "Rotation": [-30.00, 0.00, 0.00],

 "Depthmap": 1,

 "Background": 0,

"Depth\_range": [0.0, 25.0],

 "Resolution": [2048, 2048],

 "Hor\_range": [-35.0, 35.0],

 "Ver\_range": [-30.0, 30.0],

 “Focal”: [1210, 1210],

"Principle\_point": [1024, 1024]

 },

 {

 "Name":" V2”,

"Projection": "Perspective",

 “Video”: 1,

 "BitDepthColor": 10,

 "BitDepthDepth": 10,

 "ColorSpace": "YUV420",

 "DepthColorSpace": "YUV420",

 "Position": [0.3, 0.4, 1.65],

 "Rotation": [0.00, 0.00, 0.00],

 "Depthmap": 1,

 "Background": 0,

"Depth\_range": [0.0, 25.0],

 "Resolution": [2048, 2048],

 "Hor\_range": [-35.0, 35.0],

 "Ver\_range": [-30.0, 30.0],

 “Focal”: [1210, 1210],

"Principle\_point": [1024, 1024]

 },

 "Name":" V3”,

"Projection": "Perspective",

 “Video”: 1,

 "BitDepthColor": 10,

 "BitDepthDepth": 10,

 "ColorSpace": "YUV420",

 "DepthColorSpace": "YUV420",

 "Position": [0.3, 0.4, 1.65],

 "Rotation": [30.00, 0.00, 0.00],

 "Depthmap": 1,

 "Background": 0,

"Depth\_range": [0.0, 25.0],

 "Resolution": [2048, 2048],

 "Hor\_range": [-35.0, 35.0],

 "Ver\_range": [-30.0, 30.0],

 “Focal”: [1210, 1210],

"Principle\_point": [1024, 1024]

 }

}

# References

[N16730] Depth map formats used within MPEG 3D technologies, ISO/IEC JTC1/SC29/WG11 MPEG2017/ N16730, Geneva, January 2017.

[N17252] Interchangeable Formats in VFX Industry (v1), ISO/IEC JTC1/SC29/WG11 MPEG2017/N17252, Macau, October 2018.

[N17399] Revised text of ISO/IEC FDIS 23090-2 Omnidirectional Media Format, ISO/IEC JTC1/SC29/WG11 MPEG2018/N17399, Gwangju, January 2018.

[N17612] Investigation of 3DoF+ Video, ISO/IEC JTC1/SC29/WG11 MPEG2018/N17612, San Diego, April 2018.

 [N17741] Draft 1.0 Text of TR ISO/IEC 23090-1 Immersive Media Architectures, ISO/IEC JTC1/SC29/WG11 MPEG2018/N17741, Ljubljana, July 2018.

[N17761] 3DoF+ Software Platform Description, ISO/IEC JTC1/SC29/WG11 MPEG2018/N17761, Ljubljana, July 2018.

[N18064] Call for MPEG-I Visual Test Materials, ISO/IEC JTC1/SC29/WG11 MPEG2018/N18064, Macao, October 2018.

[N18065] Exploration Experiments for MPEG-I: Windowed-6DoF, ISO/IEC JTC1/SC29/WG11 MPEG2018/N18065, Macau SAR, CN, October 2018.

[N18066] Exploration Experiments for MPEG-I: Omnidirectional 6DoF, ISO/IEC JTC1/SC29/WG11 MPEG2018/N18066, Macau SAR, CN, October 2018.

[N18067] Exploration Experiments for MPEG-I: Compression of Dense Representation of Light Fields, ISO/IEC JTC1/SC29/WG11 MPEG2018/N18067, Macau SAR, CN, October 2018.

[N18070] 3DoF+ Software Platform Description, ISO/IEC JTC1/SC29/WG11 MPEG2018/N18070, Macau SAR, CN, October 2018.

[N18166] Summary on MPEG-I Visual Activities, ISO/IEC JTC1/SC29/WG11 MPEG2019/N18166, Marrakech, January 2019.

[N18167] Overview of MPEG-I Visual Test Materials, ISO/IEC JTC1/SC29/WG11 MPEG2019/N18167, Marrakech, January 2019.

1. https://www.blender.org/ [↑](#footnote-ref-1)