 ISO/IEC JTC 1/SC 29/WG 7 N555

**ISO/IEC JTC 1/SC 29/WG 7  
MPEG 3D Graphics and Haptics Coding   
Convenorship: AFNOR (France)**

**Document type:** Output Document

**Title:** Common Test Conditions for V3C and V-DMC

**Status:** Approved

**Date of document:** 2023-03-11

**Source:** ISO/IEC JTC 1/SC 29/WG 7

# Expected action: None

# Action due date: None

**No. of pages:** 21 (with cover page)

**Email of Convenor:** marius.preda @ imt . fr

**Committee URL:** [https://isotc.iso.org/livelink/livelink/open/jtc1sc29wg7](https://isotc.iso.org/livelink/livelink/open/jtc1sc29wg3)

**INTERNATIONAL ORGANIZATION FOR STANDARDIZATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

**ISO/IEC JTC 1/SC 29/WG 7 MPEG 3D Graphics Coding**

**ISO/IEC JTC 1/SC 29/WG 7 N** **555**

**January 2023, Virtual**

|  |  |
| --- | --- |
| **Title** | **Common Test Conditions for V3C and V-DMC** |
| **Source** | **WG 7, MPEG 3D Graphics Coding** |
| **Status** | **Approved** |
| **Serial Number** | **MDS22485** |

# Abstract

This document defines common test conditions and software reference configurations to be used in the context of core experiments (CE) conducted after the 141st MPEG meeting. These CTCs are recommended for use in technical contributions to the 142nd MPEG meeting.

# Introduction

Common test conditions (CTC) are desirable to conduct experiments in a well-defined environment and ease the comparison of the outcome of experiments. The structure of this document reflects parts of the MPEG CfP for Dynamic Mesh Coding [1] and 3D graphics renderer for the CfP on dynamic mesh compression [3] with the following set of test conditions:

Table 1 List of test conditions and applicability to test models

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Condition** | **Test condition** | **TMM** | | |
| **AI** | **LD** | **RA** |
| C0 | Lossless Geometry – Lossless Attributes | ✓ |  |  |
| C1 | Lossy Geometry – Lossy Attributes | ✓ |  |  |
| C2 | Lossy Geometry – Lossy Attributes |  | ✓ |  |

Note: “lossy” and “lossless” geometry/colour encoding is defined in the “Requirements for Mesh Coding” output document [5].

A subset of these test conditions might be used for a particular experiment.

The V-DMC test model (TMM) is tested on content in Dynamic Objects with Texture Mapping category. The software is available in the MPEG Gitlab repository at

<http://mpegx.int-evry.fr/software/MPEG/dmc/mpeg-vmesh-tm>

The following sections define sequences and encoding parameters to be used for each category. Input contributions should include a set of results as complete as possible that apply to the proposal. Results should be reported using the attached Excel sheets.

# Test sequences

Below is a list of the dynamic mesh test material test sequences to be used. The test class is an indicator of how complex a dynamic mesh is to encode, where A is the lowest and C is the highest complexity. All test sequences are available in the MPEG Content repository accessible under the following URL (MPEG password is required):

<https://mpegfs.int-evry.fr/mpegcontent/ws-mpegcontent/CfP/Mesh/CONTENT/voxelized>

Note: Downloaded test material should be verified with MD5 checksums. Each zip file of a test material contains an MD5 file (with the corresponding md5 sums for each file in the archive).

Table 2 Test material datasets (number (#) of vertices and faces are expressed per frame).

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test**  **Category** | **Test**  **Class** | **Test material**  **dataset filename** | **# Frames** | **# Vertices** | **# Faces** | **Geometry**  **Precision** | **Texture Coord.**  **Precision** | **Texture**  **Map Size** | **Color Attribute** | **Sequence**  **Number** |
| Dynamic Objects with Texture Mapping | A | longdress | 300 | 22k | 40k | 10 bits | 12 bits | 2k x 2k | NA | 1 |
| soldier | 300 | 22k | 40k | 10 bits | 12 bits | 2k x 2k | NA | 2 |
| B | basketball\_player | 300 | 20k | 40k | 12 bits | 12 bits | 2k x 2k | NA | 3 |
| dancer | 300 | 20k | 40k | 12 bits | 12 bits | 2k x 2k | NA | 4 |
| C | mitch | 300 | 16k | 30k | 12 bits | 13 bits | 4k x 4k | NA | 5 |
| thomas | 300 | 16k | 30k | 12 bits | 13 bits | 4k x 4k | NA | 6 |
| football | 300 | 25k | 40k | 12 bits | 13 bits | 4k x 4k | NA | 7 |
| levi | 150 | 20k | 40k | 12 bits | 13 bits | 4k x 4k | NA | 8 |

Notes:

1. The order of the vertices as they are stored in the file is not necessary to be maintained in the decoded versions.

2. The order of the connectivity of the vertices as they are stored in the file is not necessary to be maintained in the decoded versions, but for lossless compression condition, the orientation of the triangles should be maintained (see Lossless Connectivity Compression definition in [1] ).

3. The datasets contain one PNG, one MTL and one OBJ file for each mesh frame, whereby the OBJ naming convention includes a suffix that indicates the precision for geometry and texture coordinates integer values, which are also indicated in Table 2. The PNG corresponds to the texture map, the MTL file corresponds to the wavefront material template library file that links the mesh with the corresponding texture image, and the OBJ corresponds to the voxelized mesh. The voxelized meshes available in the dataset contain only one unique connectivity information for both vertex positions and texture coordinates (notice that this could lead to vertices with duplicate 3D positions but different texture coordinates).

# Reporting

Proposals will be evaluated based on bit rate, objective quality metrics, and complexity. An Excel sheet is provided together with this document to report the results. Proponents are expected to deliver a full set of results for each targeted testing condition. Each proposal must include modifications to the specification including syntax, semantics, and decoding process description when applicable.

Reference results for the unmodified test models are available after each update of the test model software, e.g. in [4] for TMM including respective software tags.

# Bit rate reporting

Bitrates shall be reported overall, as well as separately for geometry, texture and metadata, where applicable. The provided Excel sheet reporting specifies which data shall be reported for which testing conditions.

For conditions C1 and C2 the rate shall be reported as Mbit/s (1 Mbit = 1,000,000 bits).

For Condition C0, the rate shall be reported as bpf (bits per face). The evaluation will be made based on the rate-distortion (RD) performance, and RD curves shall be plotted using PSNR as the quality measure for all categories.

# Distortion reporting

# General

The distortion metrics compare the original meshes with the decoded meshes and provide PSNR values describing geometry and attribute distortion.

Proposals will be evaluated using a set of objective metrics, including PSNR of an image-based geometric error (GEO\_PSNR) and point-based geometric errors (D1 and D2) for geometry, as well as PSNR of color attributes (Y\_PSNR, U\_PSNR and V\_PSNR for point-based, and Y-PSNR for image-based metrics). The evaluation will be made based on the rate-distortion (RD) performance, and RD curves shall be plotted using PSNR as the quality measure.

* + 1. **Geometric distortions**

For geometric distortions GEO\_PSNR, D1, and D2, the PSNR calculated from the symmetric distortions shall be reported. The distortion metrics and calculation of PSNR is specified in Annex A.1 of this document.

* + 1. **Attribute distortions**

The color distortions, described in Annex A are measured as image based metrics as Y-PSNR or per point in YUV space, and is reported as three PSNR values one for each channel: Y, U, and V. The PSNRs for each component will be individually used to compare methods. The reported distortion measures shall be averaged over all frames. In addition, proponents shall report the MSE and PSNR per frame as a CSV file upon request. The metric software provides means to compute and export these results.. Details on the image-based attribute distortion calculation are provided in Annex A.2 of this document.

# Complexity reporting

The encoder and decoder runtimes shall be reported with respect to the unmodified test model runtimes on the same system. The TMM software reports wall time and user time. In this case user time shall be reported. User time shall be reported with video codec runtimes.

In addition, contributions should contain the following aspects regarding codec complexity:

* theoretical study of the complexity on encoder and decoder, for example number of operations, memory access patterns, floating point operations, etc.;
* execution time with respect to anchor software;
* implications with respect to the video codec type, e.g. chroma subsampling, bit depth, profiles, type, tools;
* suitability of the hardware implementation, i.e. avoid floating points, memory usage;
* potential for parallelization and decoder parsing dependency.

# Distortion metric software

The metric software for calculating the objective distortion can be accessed from the MPEG GitLab:

<http://mpegx.int-evry.fr/software/MPEG/PCC/mpeg-pcc-mmetric/-/tree/1_1_2>

Annex A.3 provides details on how to run the evaluation software.

The Mesh renderer software can be found in the MPEG GIT:

<http://mpegx.int-evry.fr/software/MPEG/PCC/mpeg-pcc-renderer>

The master branch contains the latest version of this software (tag. V7.0) and will be used for subjective evaluation.

# Summary of reporting data per condition

The following table summarizes the data to be reported per test condition.

Table 3 Test conditions distortions reporting.

|  |  |  |  |
| --- | --- | --- | --- |
| **Condition** | **C0** | **C1** | **C2** |
| enc.total.bits | ✓ | ✓ | ✓ |
| dec.numfaces |  | ✓ | ✓ |
| dec.d1-psnr |  | ✓ | ✓ |
| dec.d2-psnr |  | ✓ | ✓ |
| dec.y-psnr |  | ✓ | ✓ |
| dec.cb-psnr |  | ✓ | ✓ |
| dec.cr-psnr |  | ✓ | ✓ |
| dec.ibpsnr.geo-psnr |  | ✓ | ✓ |
| dec.ibpsnr.luma-psnr |  | ✓ | ✓ |
| enc.runtime | ✓ | ✓ | ✓ |
| dec.runtime | ✓ | ✓ | ✓ |

*enc.bits* represents the output stream/file size as reported by TMM.

*dec.numfaces* is the number of decoded faces in the coded sequence.

*y-*, *cb-*, *cr-, d1-, d2-, ibpsnr.geo-,* and *ibpsnr.luma-* are end-to-end distortion metrics that compare the output of the decoder with the input to the encoder.

When reporting results, codec runtime is reported by the software, corresponding to the TMM software itself.

# Common Test Conditions for TMM

This sub-section will be continuously updated as more detailed information is made available. For the time being, the most up-to-date software description and usage description for TMM are provided in  [4].

# Software – TMM

The common software used for any experiments in V-DMC is available from the MPEG GitLab: <http://mpegx.int-evry.fr/software/MPEG/dmc/mpeg-vmesh-tm>

Software documentation and usage description are provided in [4].

# Coding parameters and Configuration files

# General

The TMM software supports config file handling via a -c/--config=option command line parameter. Multiple config files may be specified by repeating the option, with settings in later files overriding those in earlier ones.

The coding parameters (common configurations, rate points, video coding configurations) are available in the respective directories under the /cfg directory of the provided software. The configuration parameters for each sequence are available in cfg/sequences.yaml. The configuration parameters for each CTC conditions are available in the other yaml files as shown in 4.2.2, 4.2.3, and 4.2.4 for C0, C1, and C2, respectively.

The process of generating configuration files according to specific system paths is performed using scripts provided with TMM reference software:

1. copy and edit cfg/cfg-site-default.yaml as cfg/cfg-site.yaml, by setting corresponding paths for the binaries, sequence prefix, and the external tool configuration prefix;
2. execute the ./scripts/gen-cfg.sh' script:

./scripts/gen-cfg.sh --cfgdir=./cfg/ --outdir=</path/to/generated/test/cfgfiles>

The same process can also be executed with the following command:

./scripts/create\_configuration\_files.sh \

--outdir=generatedConfigFilesHM \

--seqdir=/path/to/contents/voxelized/ \

--codec=hm

### Coding parameters and Configuration files for test condition C0

The following encoder parameters and corresponding values shown below along with the configuration files files cfg/cfg-cond-ai-ll.yaml for C0 lossless coding are used for lossless geometry-and-attribute coding in all-intra mode.

Example command line inputs for condition C0 are provided in Annex B.1.2.

### Coding parameters and Configuration files for test condition C1

The following encoder parameters and corresponding values shown below along with the configuration files cfg/cfg-cond-ai.yaml for C1 are used for lossy geometry-and-attribute coding in all-intra mode.

Example command line inputs for condition C1 are provided in Annex B.1.3.

### Coding parameters and Configuration files for test condition C2

The following encoder parameters and corresponding values shown below along with the configuration files cfg/cfg-cond-ld.yaml for C2 are used for lossy geometry-and-attribute coding in low delay mode.

Example command line inputs for condition C2 are provided in Annex B.1.4.

### Command line example

Below is a selection of command line examples to create the three CTC conditions:

# C0

./build/Release/bin/encode \

--config=./generatedConfigFiles/s3c0r1\_bask/encoder.cfg \

--frameCount=1 \

--compressed=s3c0\_bask.vmesh \

./build/Release/bin/decode \

--config=./generatedConfigFiles/s3c0r1\_bask/decoder.cfg \

--compressed=s3c0\_bask.vmesh \

--decMesh=s3c0\_bask\_%04d\_dec.obj \

--decTex=s3c0\_bask\_%04d\_dec.png \

--decMat=s3c0\_bask\_%04d\_dec.mtl \

# C1

./build/Release/bin/encode \

--config=./generatedConfigFiles/s3c1r1\_bask/encoder.cfg \

--frameCount=1 \

--compressed=s3c1r1\_bask.vmesh \

./build/Release/bin/decode \

--config=./generatedConfigFiles/s3c1r1\_bask/decoder.cfg \

--compressed=s3c1r1\_bask.vmesh \

--decMesh=s3c1r1\_bask\_%04d\_dec.obj \

--decTex=s3c1r1\_bask\_%04d\_dec.png \

--decMat=s3c1r1\_bask\_%04d\_dec.mtl \

# C2

./build/Release/bin/encode \

--config=./generatedConfigFiles/s3c2r1\_bask/encoder.cfg \

--frameCount=1 \

--compressed=s3c2r1\_bask.vmesh \

./build/Release/bin/decode \

--config=./generatedConfigFiles/s3c2r1\_bask/decoder.cfg \

--compressed=s3c2r1\_bask.vmesh \

--decMesh=s3c2r1\_bask\_%04d\_dec.obj \

--decTex=s3c2r1\_bask\_%04d\_dec.png \

--decMat=s3c2r1\_bask\_%04d\_dec.mtl \

# References

1. CfP for Dynamic Mesh Coding, ISO/IEC JTC1/SC29 WG7 Doc. N21000, Online, October 2021.
2. Corrigendum of Dynamic Mesh Coding CfP, ISO/IEC JTC1/SC29 WG7 Doc. N21150, Online, January 2022.
3. [V-Mesh] Renderer software, ISO/IEC JTC1/SC29 WG7 Doc. m58240, Online, October 2021.
4. V-DMC performance evaluation and anchor results, ISO/IEC JTC1/SC29 WG7 Doc. N22464, Online, January 2023.
5. Draft Requirements for Mesh Coding, ISO/IEC JTC1/SC29 WG7. N20363, Online, April 2021.
6. [V-DMC][metrics] new mmetric software and CTC conditions, ISO/IEC JTC1/SC29 WG7 Doc. m58240, Online, October 2022

**Annex A: Objective Evaluation Metrics & Usage of Metric Software**

***A.1 Point-based metric***

The point-based metric converts the reference and distorted meshes into two point clouds by applying the sampling procedure described in section A.1.1.

***A.1.1 Mesh sampling***

The point cloud is created by performing ray-casting in the axis direction (x,y,z), depending on the normal of the triangle. A hit test determines if the casted ray hits the triangle, then the color is obtained by barycentric interpolation (to determine the UV coordinate of the point), and then bilinear interpolation (to get the RGB value from texture map). The normal of the triangle is computed as the cross product of its two edges and normalized to have a unit length. All the points obtained by sampling the triangle inherit its normal vector.

***A.1.2 Geometric Distortions***

Let and denote the original and the compressed point cloud, respectively. Consider evaluating the compression errors, denoted as in point cloud relative to reference point cloud. The steps to compute both point-to-point error (D1) and point-to-plane error (D2) for geometric errors are summarized in the following and illustrated in the below figure.

For each point in point cloud, i.e., the black point in the figure, identify a corresponding point in point cloud, i.e. the red point in the figure. Nearest neighbour is used to locate the corresponding point. In particular, a KD-tree search is used to perform the nearest neighbour search in order to reduce the computation complexity.

***A.1.2.1 Computing D1***

Determine an error vector by connecting the identified point in reference point cloud  to point in point cloud . The length of the error vector is the point-to-point error, i.e.,

Based on the point-to-point distances for all points , the point-to-point error (D1) for the whole point cloud, withas the number of points in point cloud, is defined as:

***A.1.2.2 Computing D2***

Project the error vector along the normal direction and get a new error vector. In this way, the point-to-plane error is computed as,

The point-to-plane error (D2) for the whole point cloud is then defined as,

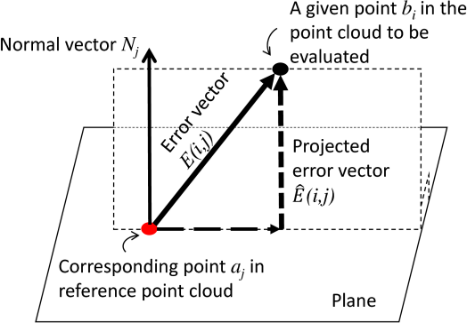


Figure 1. Illustration of point-to-point distance (D1) and point-to-plane distance (D2)

***A.1.3 Geometric PSNR Calculation***

The geometric PSNR value is computed as:

where is the maximum length of the sequence bounding box (maxBBLength) as defined in Table 4, and is the symmetric mean squared point-to-point () or point-to-plane () error, which are obtained by considering the maximum distortions and computed as follows:

and .

For dynamic content, the peak value is unchanged over the frames of a sequence.

***A.1.4 Attribute Distortions***

The attribute PSNR value is computed as:

For color attributes, the MSE for each of the three color components is calculated. A conversion from RGB space to YUV space is conducted using ITU-R BT.709, since YUV space correlates better with human perception. A symmetric computation of the distortion is utilized, in the same way as is done for geometric distortions. The maximum distortion between the two passes is selected as the final distortion. Since the color attributes for all test data have a bit depth of 8 bits per point, the peak value for PSNR calculation is 255.

***A.2 Image-based metric***

The computation of the lossy metric is based on image MSE/PSNR processing [7][8]. An overview of the approach is given in Figure 2. For each frame, the reference and the distorted models are rendered for several view directions , using an orthographic projection (see section A.2.1). The images obtained from the rendering of reference and distorted models are then compared using some adapted image MSE/PSNR metrics (see section A.2.3). The results are averaged over a set of view directions for the frame and over the frames of the sequence.

Diagram

Description automatically generated

Figure 2. Overview of the image-based metric. Ref/DisColor are the color images/buffers. Ref/DisMask are binary images where pixel[i,j]=1 if there exist a projection in associated color buffer, and 0 otherwise. Ref/DisDepth are the depth buffers. All the buffers have same dimensions of 2048x2048 pixels.

The number of views is fixed to 16 and the resolution of image, depth and mask buffers is set to 2048x2048 pixels. The Figure 3 gives an example of rendered image buffers for the 16 view directions.

Graphical user interface, application

Description automatically generated

Figure 3. Example of color images generated for the basketball player using 16 views.

***A.2.1 Rendering of one view***

Graphical user interface, application

Description automatically generated

Figure 4. Rendering of image buffers for a given view direction vdi.

The rendering is part of the metric and is implemented inside the metric software.

The rendering of one view is illustrated in Figure 4. The pixels of the image could be obtained by ray tracing or rasterization of the mesh.

The bounding sphere is obtained by summing the axis aligned bounding box of the distorted and reference objects and taking the diagonal and center of the resulting bounding box.

The view directions always points toward the center of the bounding sphere in 3D space.

The mesh is rendered using an *orthogonal projection*. The projection plane for the direction is the plane tangent to the bounding sphere and perpendicular to the view direction vector.

The mesh is rendered using *clockwise back-face culling* which suits all the models of the anchor in terms of visual rendering.

Note: if the provided answer does not preserve faces orientations based on clockwise orientations it will get low MSE/PSNR scores for the metric.

The rendering step generates the color, the mask, and the depth buffers.

* The color buffer contains for each pixel i,j the RGB value, of the nearest projected triangle.
  + In case of a textured mesh, the RGB color is obtained by bilinear interpolation of the texture map using triangle UV coordinates.
  + In case of color per vertex meshes (no texture map), the vertex colors are blended using barycentric coordinates.
* The mask buffer contains for each pixel i,j a binary value set to 1 if a projection for this pixel exists and 0 otherwise.
* The depth buffer contains for each pixel i,j the distance from the projection plane to the 3D surface in 3D space.

***A.2.2 Positioning of views***

The positioning of the 16 views, that is to say the set of orientations, is obtained by using a [Fibonacci sphere lattice](http://extremelearning.com.au/evenly-distributing-points-on-a-sphere/) [11]. This distribution aims at generating points over a sphere in an evenly spaced manner. Once one has the points , the directions are the vectors passing through these points and pointing toward the center of the sphere. The Figure 3 gives an example of generated images for 16 views using this method.

The Fibonacci sphere samples are computed as follows:

const double pi = std::atan(1.0) \* 4;

// golden angle in radians

Const float phi = (float)(pi \* (3. - std::sqrt(5.)));

// glm::vec3 a vector of three floats, each entry a direction

std::vector<glm::vec3> camDir;

for (size\_t i = 0; i < targetNbSamples; ++i) {

float y = 1 - (i / float(targetNbSamples - 1)) \* 2; // y goes from 1 to - 1

float radius = std::sqrt(1 - y \* y); // radius at y

float theta = phi \* i; // golden angle increment

float x = std::cos(theta) \* radius;

float z = std::sin(theta) \* radius;

camDir.push\_back(glm::vec3(x, y, z));

}

A special attention is also given to generate the up vectors (see Figure 4). The up vector determines, for a given direction , the rotation of the camera on the direction axis, and by side effect the final orientation of the model in the image. At the end, we obtain the camera matrix using the well-known [LookAt function from OpenGL](https://www.khronos.org/registry/OpenGL-Refpages/gl2.1/xhtml/gluLookAt.xml) [12].

For a given view direction viewDir (), the viewUp () vector is defined as follows:

if (glm::abs(viewDir) == glm::vec3(0, 1, 0))

viewUp = glm::vec3(0, 0, 1);

else

viewUp = glm::vec3(0, 1, 0);

In other terms, whenever view direction is not a north or south pole, we use y axis vector as the up vector , otherwise we use the z axis vector as the up vector .

Those values were selected so most of the models visually renders head at top of image when rendered by side or front views. The special case of poles is the simplest positive default value for .

***A.2.3 distortion for one view***

The calculus of the distortion is based on the general Mean Squared Error (MSE) formula. Let be a sample of an original image, a sample of a distorted image and the width of both buffers in pixels. The for the two images is calculated as follows:

In the rest of this document, we will decline this formula to our needs, especially restricting processing to parts where the mask generated from the reference and distorted models is equal to one. Let be a matrix of same size of and (Each mask buffer contains for each pixel i,j a binary value set to 1 if a projection for this pixel exists and 0 otherwise. See section B.2.1).

Let be the width of the buffers in pixels. The combined number of projected pixels of all the view directions for one frame is:

A conversion from RGB space to YUV space is performed by using ITU-R BT.709, since YUV space correlates better with human perception.

In the following the color image/buffer is considered YUV.

Let be a sample of the color image/buffer of the reference model for view direction , a sample of the color image/buffer of the distorted model for view direction . The color MSE for the YUV images, noted , is calculated as follows on each component in :

Let be a sample of the depth image/buffer of the reference model for view direction , a sample of the depth image/buffer of the distorted model for view direction . Let be the dynamic range of the depth signal initialized with the maximum between the diagonals of the bounding boxes of both models. The depth MSE for the depth images/buffers, noted , is calculated as follows:

This depth renormalization to 255 is used to get geometric MSEs and PSNRs comparable to the color ones.

The final metric results per frame are the followings:

And the respective associated with are computed with the following formula:

The metric tool reports also the ratio of unmatched samples defined as follows:

where is given by:

Additionally, the metric tool reports also the ratio () of the bounding box diagonal of the reference and the diagonal of the bounding box of the distorted mesh as follows:

***A.3 Calculation of MSE and PSNR for a sequence***

Metrics for a sequence are obtained by computing the **Minimum**, the **Maximum**, the **Mean**, the **Variance,** and the **Standard deviation** of the metric evaluated on each frame of the sequence.

***A.4 Software usage for computing the point cloud and image-based metrics***

The “equ” and “ibsm” metrics are symmetrical, but the “pcc” is not. To ensure the numerical stability of the results among participants it is required that:

* the **--modelA and --MapA** are used for the **reference** model**.**
* the **--modelB and --mapB** are used for the **distorted** model**.**

***A.4.1 Lossless condition***

Lossless condition with log to console:

./mm compare \

--mode equ \

--inputModelA ref.obj –inputMapA ref.png \

--inputModelB dis.obj –inputMapB dis.png > summary.txt

***A.4.1 Lossy condition***

Lossy condition on dequantized models using the point-based metric with log to console and statistics reporting into files:

./mm \

reindex --sort oriented -i ref.obj -o ID:ref\_reordered END

sample --mode grid --gridSize 512 \

--useNormal --useFixedPoint --minPos "$globalMinPos" --maxPos "$globalMaxPos" \

--bilinear -i ID:ref\_reordered -m ref.png -o ID:pcRef END\

reindex --sort oriented -i dis.obj -o ID:dis\_reordered END

sample --mode grid --gridSize 512 \

--useNormal --useFixedPoint --minPos "$globalMinPos" --maxPos "$globalMaxPos" \

--bilinear -i ID:dis\_reordered -m dis.png -o ID:pcDis END\

compare --mode pcc \

--inputModelA ID:pcRef --inputModelB ID:pcDis \

--outputCsv perFrame.csv > summary.txt

Lossy condition on dequantized models using the image-based metric with log to console and statistics reporting into files:

./mm compare --mode ibsm \

--inputModelA ref.obj --inputMapA ref.png \

--inputModelB dis.obj --inputMapB dis.png \

--outputCsv perFrame.csv > summary.txt

**Attention:** Lossy condition shall be applied on dequantized models.

**Annex B: TM-D Software configuration**

***B.1 General***

This section provides several examples of configuration files for the TMM encoder, decoder, and MMetric software.

***B.1.1 Lossless AI condition***

Encoder:

# This file was automatically generated from:

# /home/us000185/work/test/mpeg-vmesh-tm/cfg/cfg-site-default.yaml

# /home/us000185/work/test/mpeg-vmesh-tm/cfg/cfg-tools.yaml

# /home/us000185/work/test/mpeg-vmesh-tm/cfg/cfg-cond-ai-ll.yaml

# /home/us000185/work/test/mpeg-vmesh-tm/cfg/cfg-cond-ai.yaml

# /home/us000185/work/test/mpeg-vmesh-tm/cfg/cfg-cond-ld.yaml

# /home/us000185/work/test/mpeg-vmesh-tm/cfg/sequences.yaml

# /home/us000185/work/test/mpeg-vmesh-tm/cfg/cfg-site.yaml

srcMesh: /Streams/dataset/basketball\_player\_fr%04d\_qp12\_qt12.obj

srcTex: /Streams/dataset/basketball\_player\_fr%04d.png

startFrameIndex: 1

frameCount: 300

positionBitDepth: 12

texCoordBitDepth: 12

target: 1

minCCTriangleCount: 0

minPosition: -725.812988 -483.908997 -586.02002

maxPosition: 1252.02002 1411.98999 1025.34998

textureParametrizationWidth: 2048

textureParametrizationHeight: 2048

textureParametrizationGutter: 16

textureParametrizationQuality: QUALITY

ai\_subdivIt: 0

ai\_sdeform: 1

ai\_forceNormalDisp: 0

ai\_unifyVertices: 1

ai\_deformNormalThres: -2.0

ai\_deformNNCount: 1

ai\_deformFlipThres: -2.0

ai\_useInitialGeom: 1

ai\_smoothMotion: 0

ai\_smoothMethod: 1

# skipped 1 elements: 'defined ${fitsubdiv\_with\_mapping} and !${fitsubdiv\_with\_mapping}' is false

# with fitsubdiv\_with\_mapping = '(undef)'

# skipped 10 elements: 'defined ${pp} and ${pp} < 1.0 and ( defined ${fitsubdiv\_inter} and ${fitsubdiv\_inter})' is false

# with pp = '1'

# and fitsubdiv\_inter = '(undef)'

invertOrientation: 0

baseMeshTexCoordBitDepth: 10

gofMaxSize: 32

liftingBias: 0.3333,0.3333333333333333,0.3333333333333333

unifyVertices: 0

textureVideoWidth: 2048

textureVideoHeight: 2048

textureVideoEncoderConvertConfig: /mpeg-vmesh-tm/cfg/hdrconvert/bgr444toyuv420.cfg

textureVideoDecoderConvertConfig: /mpeg-vmesh-tm/cfg/hdrconvert/yuv420tobgr444.cfg

analyzeGof: 0

subdivIsBase: 1

baseIsSrc: 1

encodeGeometryVideo: 0

ai\_fitSubdiv: 0

liftingIterationCount: 0

textureTransferEnable: 0

textureBGR444: 1

unifyVertices: 0

dracoMeshLossless: 1

textureVideoEncoderConfig: /mpeg-vmesh-tm/cfg/hm/texture-ai-ll.cfg

baseMeshPositionBitDepth: 12

baseMeshTexCoordBitDepth: 12

Decoder:

# This file was automatically generated from:

# ./scripts/../cfg-site-default.yaml

# ./scripts/../cfg/cfg-tools.yaml

# ./scripts/../cfg/cfg-cond-ai-ll.yaml

# ./scripts/../cfg/cfg-cond-ai.yaml

# ./scripts/../cfg/cfg-cond-ld.yaml

# ./scripts/../cfg/sequences.yaml

# ./scripts/../cfg/cfg-site.yaml

startFrameIndex: 1

textureVideoDecoderConvertConfig: /mpeg-vmesh-tm/cfg/hdrconvert/yuv420tobgr444.cfg

MMetric:

# This file was automatically generated from:

# ./scripts/../cfg/cfg-site-default.yaml

# ./scripts/../cfg/cfg-tools.yaml

# ./scripts/../cfg/cfg-cond-ai-ll.yaml

# ./scripts/../cfg/cfg-cond-ai.yaml

# ./scripts/../cfg/cfg-cond-ld.yaml

# ./scripts/../cfg/sequences.yaml

# ./scripts/../cfg/cfg-site.yaml

startFrameIndex: 1

frameCount: 300

srcMesh: /Streams/dataset/basketball\_player\_fr%04d\_qp12\_qt12.obj

srcTex: /Streams/dataset/basketball\_player\_fr%04d.png

positionBitDepth: 12

texCoordBitDepth: 12

minPosition: -725.812988 -483.908997 -586.02002

maxPosition: 1252.02002 1411.98999 1025.34998

pcc: 1

ibsm: 1

gridSize: 1024

resolution: 1977.833008

***B.1.2 Lossy AI condition***

Encoder:

# This file was automatically generated from:

# ./scripts/../cfg/cfg-site-default.yaml

# ./scripts/../cfg/cfg-tools.yaml

# ./scripts/../cfg/cfg-cond-ai.yaml

# ./scripts/../cfg/cfg-cond-ld.yaml

# ./scripts/../cfg/sequences.yaml

# ./scripts/../cfg/cfg-site.yaml

imesh: /Streams/dataset/longdress\_voxelized/longdress\_fr%04d\_qp10\_qt12.obj

itex: /Streams/dataset/longdress\_voxelized/longdress\_fr%04d.png

fstart: 1051

fcount: 300

gdepth: 10

tdepth: 12

qt: 12

target: 0.03

cctcount: 8

minPosition: -0.475553989 -1.4576 -0.284981996

maxPosition: 481.324005 1023.37 659.137024

width: 2048

height: 2048

quality: QUALITY

gutter: 16

ai\_subdivIt: 3

ai\_sdeform: 1

ai\_forceNormalDisp: 0

ai\_unifyVertices: 1

ai\_deformNormalThres: -2.0

ai\_deformNNCount: 1

ai\_deformFlipThres: -2.0

ai\_useInitialGeom: 1

ai\_smoothMotion: 0

ai\_smoothMethod: 1

# skipped 1 elements: 'defined ${fitsubdiv\_with\_mapping} and !${fitsubdiv\_with\_mapping}' is false

# with fitsubdiv\_with\_mapping = '(undef)'

# skipped 10 elements: 'defined ${pp} and ${pp} < 1.0 and ( defined ${fitsubdiv\_inter} and ${fitsubdiv\_inter})' is false

# with pp = '0.03'

# and fitsubdiv\_inter = '(undef)'

invorient: 0

tqp: 10

gofmax: 32

dqb: 0.3333,0.3333333333333333,0.3333333333333333

normuv: 0

unifvertices: 0

texwidth: 2048

texheight: 2048

cscencconfig: /mpeg-vmesh-tm/cfg/hdrconvert/bgr444toyuv420.cfg

cscdecconfig: /mpeg-vmesh-tm/cfg/hdrconvert/yuv420tobgr444.cfg

analyzeGof: 0

# skipped 1 elements: '${pp} >= 0.24 and ${pp} < 1.0' is false

# with pp = '0.03'

# skipped 2 elements: 'not defined ${pp} or ${pp} == 1.0' is false

# with pp = '0.03'

gvencconfig: /mpeg-vmesh-tm/cfg/hm/ctc-hm-displacements-map-ai-main10.cfg

tvencconfig: /mpeg-vmesh-tm/cfg/hm/ctc-hm-texture-ai.cfg

liftingIt: 3

gqp: 10

tvqp: 47

dqp: 30,42,42

Decoder:

# This file was automatically generated from:

# ./scripts/../cfg/cfg-site-default.yaml

# ./scripts/../cfg/cfg-tools.yaml

# ./scripts/../cfg/cfg-cond-ai.yaml

# ./scripts/../cfg/cfg-cond-ld.yaml

# ./scripts/../cfg/sequences.yaml

# ./scripts/../cfg/cfg-site.yaml

fstart: 1051

cscdecconfig: /mpeg-vmesh-tm/cfg/hdrconvert/yuv420tobgr444.cfg

normuv: 0

MMetric:

# This file was automatically generated from:

# ./scripts/../cfg/cfg-site-default.yaml

# ./scripts/../cfg/cfg-tools.yaml

# ./scripts/../cfg/cfg-cond-ai.yaml

# ./scripts/../cfg/cfg-cond-ld.yaml

# ./scripts/../cfg/sequences.yaml

# ./scripts/../cfg/cfg-site.yaml

fstart: 1051

fcount: 300

srcMesh: /Streams/dataset/longdress\_voxelized/longdress\_fr%04d\_qp10\_qt12.obj

srcTex: /Streams/dataset/longdress\_voxelized/longdress\_fr%04d.png

qp: 10

qt: 12

minPosition: -0.475553989 -1.4576 -0.284981996

maxPosition: 481.324005 1023.37 659.137024

pcc: 1

ibsm: 1

gridSize: 1024

***B.1.3 Lossy LD condition***

Encoder:

# This file was automatically generated from:

# ./scripts/../cfg/cfg-site-default.yaml

# ./scripts/../cfg/cfg-tools.yaml

# ./scripts/../cfg/cfg-cond-ai.yaml

# ./scripts/../cfg/cfg-cond-ld.yaml

# ./scripts/../cfg/sequences.yaml

# ./scripts/../cfg/cfg-site.yaml

imesh: /Streams/dataset/longdress\_voxelized/longdress\_fr%04d\_qp10\_qt12.obj

itex: /Streams/dataset/longdress\_voxelized/longdress\_fr%04d.png

fstart: 1051

fcount: 300

gdepth: 10

tdepth: 12

qt: 12

target: 0.03

cctcount: 8

minPosition: -0.475553989 -1.4576 -0.284981996

maxPosition: 481.324005 1023.37 659.137024

width: 2048

height: 2048

quality: QUALITY

gutter: 16

ai\_subdivIt: 3

ai\_sdeform: 1

ai\_forceNormalDisp: 0

ai\_unifyVertices: 1

ai\_deformNormalThres: -2.0

ai\_deformNNCount: 1

ai\_deformFlipThres: -2.0

ai\_useInitialGeom: 1

ai\_smoothMotion: 0

ai\_smoothMethod: 1

# skipped 1 elements: 'defined ${fitsubdiv\_with\_mapping} and !${fitsubdiv\_with\_mapping}' is false

# with fitsubdiv\_with\_mapping = '(undef)'

# skipped 10 elements: 'defined ${pp} and ${pp} < 1.0 and ( defined ${fitsubdiv\_inter} and ${fitsubdiv\_inter})' is false

# with pp = '0.03'

# and fitsubdiv\_inter = '(undef)'

invorient: 0

tqp: 10

gofmax: 32

dqb: 0.3333,0.3333333333333333,0.3333333333333333

normuv: 0

unifvertices: 0

texwidth: 2048

texheight: 2048

cscencconfig: /mpeg-vmesh-tm/cfg/hdrconvert/bgr444toyuv420.cfg

cscdecconfig: /mpeg-vmesh-tm/cfg/hdrconvert/yuv420tobgr444.cfg

analyzeGof: 1

# skipped 1 elements: '${pp} > 0.24 and ${pp} < 1.0' is false

# with pp = '0.03'

# skipped 2 elements: 'not defined ${pp} or ${pp} == 1.0' is false

# with pp = '0.03'

# skipped 1 elements: 'defined ${fitsubdiv\_inter}' is false

# with fitsubdiv\_inter = '(undef)'

gvencconfig: /mpeg-vmesh-tm/cfg/hm/ctc-hm-displacements-map-ld-main10.cfg

# skipped 1 elements: 'defined ${fitsubdiv\_inter} and ${fitsubdiv\_inter} == 1' is false

# with fitsubdiv\_inter = '(undef)'

tvencconfig: /mpeg-vmesh-tm/cfg/hm/ctc-hm-texture-ai.cfg

liftingIt: 3

gqp: 10

tvqp: 49

dqp: 32,44,44

Decoder:

# This file was automatically generated from:

# ./scripts/../cfg/cfg-site-default.yaml

# ./scripts/../cfg/cfg-tools.yaml

# ./scripts/../cfg/cfg-cond-ai.yaml

# ./scripts/../cfg/cfg-cond-ld.yaml

# ./scripts/../cfg/sequences.yaml

# ./scripts/../cfg/cfg-site.yaml

fstart: 1051

cscdecconfig: /mpeg-vmesh-tm/cfg/hdrconvert/yuv420tobgr444.cfg

normuv: 0

MMetric:

# This file was automatically generated from:

# ./scripts/../cfg/cfg-site-default.yaml

# ./scripts/../cfg/cfg-tools.yaml

# ./scripts/../cfg/cfg-cond-ai.yaml

# ./scripts/../cfg/cfg-cond-ld.yaml

# ./scripts/../cfg/sequences.yaml

# ./scripts/../cfg/cfg-site.yaml

fstart: 1051

fcount: 300

srcMesh: /Streams/dataset/longdress\_voxelized/longdress\_fr%04d\_qp10\_qt12.obj

srcTex: /Streams/dataset/longdress\_voxelized/longdress\_fr%04d.png

qp: 10

qt: 12

minPosition: -0.475553989 -1.4576 -0.284981996

maxPosition: 481.324005 1023.37 659.137024

pcc: 1

ibsm: 1

gridSize: 1024

**Annex C: Visualization of the voxelized mesh content**

# For the dataset content, the vertices attributes (3D position and texture coordinate) are provided in integer format. For correct visualization of these voxelized meshes, a dequantization procedure that converts the integer values to floating point values should be conducted. The mmetric software provides the capability to dequantize the voxelized meshes by applying the following command line:

**mm** \

**dequantize** --inputModel ${voxelizedMesh} --outputModel ${outMesh} \

--qp ${qp} --qt ${qt} \

--minPos ${globalPosMin} --maxPos ${globalMaxPos} \

--minUV “0.0 0.0” --maxUV “1.0 1.0” \

--useFixedPoint

# The values for qp (quantization parameter for vertex 3D coordinate) and qt (quantization parameters for texture coordinate) is equivalent to the options of Geometry Precision and Texture Coordinate Precision respectively, and are indicated in Table 2. The values for globalPosMin and globalPosMax are provided in Table 4.

Table 4 Conversion parameters for the Mesh sequences

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test material**  **dataset filename** | **globalPosMin** | | | **globalMaxPos** | | | **maxBBLength** |
| **(x)** | **(y)** | **(z)** | **(x)** | **(y)** | **(z)** |
| Longdress | -0.475553989 | -1.4576 | -0.284981996 | 481.324005 | 1023.37 | 659.137024 | 1024.8276 |
| Soldier | -0.366236001 | 1.10722005 | 0.224947006 | 508.764008 | 1023.37 | 637.421997 | 1022.26277995 |
| Basketball\_player | -725.812988 | -483.908997 | -586.02002 | 1252.02002 | 1411.98999 | 1025.34998 | 1977.833008 |
| Dancer | -902.244995 | -486.196991 | -670.518005 | 621.093994 | 1576.04004 | 738.028992 | 2062.237031 |
| Mitch | -588.255981 | 5.80515003 | -469.799011 | 734.567993 | 1829.69995 | 697.179016 | 1823.89479997 |
| Thomas | -265.006989 | -4.04448986 | -248.710999 | 320.546997 | 1820.93005 | 400.225006 | 1824.97453986 |
| Football | -0.000159517003 | 3.32326999e-06 | 0.000132931003 | 1024 | 980.619995 | 966.692993 | 1023.96268540018 |
| Levi | -0.780686975 | -0.0424938016 | -0.594317973 | 0.857237995 | 1.90897 | 0.687259018 | 1.9514638016 |