Alliance for Open Media
Volumetric Visual Media Working Group

Document: **VVM-2023-004o-v3**

Call for Proposals on Static Polygonal Mesh Coding

May 12th, 2023

Status: **Output document**
Purpose: **Call for Proposals**
Source: **VVM Working Group**

Abstract

This document is the Call for Proposals (CfP) on Static Polygonal Mesh Coding technology, which targets lossless and lossy compression of static 3D polygonal meshes. It provides the timeline of the project, details on the test content, and all information related to the submission of a proposal to this CfP.

1 Introduction

3D computer graphics and modeling are widely used in areas like gaming, filmmaking, animation, geospatial applications, online commerce, AR and VR applications, entertainment, architecture, manufacturing, and cultural heritage applications. 3D objects are commonly modeled as polygonal meshes. A mesh surface is described by a collection of vertices, their positions in 3D space, and their connectivity via edges that compose polygonal faces. A mesh may have additional attributes associated with its vertices, edges, faces, and/or corners. Two common types of attributes that affect rendering are normal vectors and texture coordinates. Normal vectors affect how the light bounces off the mesh. Texture coordinates map a 3D mesh surface onto a 2D plane and define how to apply texture maps onto the mesh surface. Such mesh representation is firmly established in 3D formats (*e.g.* OBJ [1], glTF [2], USD [3]), and rendering APIs (*e.g.* OpenGL [4], Direct3D [5], Vulkan [6]). Demand for transmission and storage of 3D meshes will continue to grow since more and more digital experiences start taking advantage of augmented reality or are completely moving into immersive 3D environments.

Polygonal mesh compression tools aim at providing efficient transmission and storage. Compression for transmission can reduce network bandwidth requirements, resulting in shorter transmission times and delivery of more detailed 3D meshes with higher visual quality. Compression for storage can reduce data center costs and enable client devices to cache more 3D data.

Current state of the art compression libraries target mainly manifold triangular 3D meshes that are intended for Graphics Processing Unit (GPU) rendering. These libraries achieve compression by only quantizing vertex attributes with a minimal loss of visual quality, exploiting regular structures and patterns in the mesh, and through entropy coding. This does not result in optimal rate distortion performance for some applications such as streaming.

It is desirable that next generation mesh compression tools and standards would offer improved compression efficiency while overcoming existing limitations. Such tools and standards should also support efficient lossless and lossy compression of all types of polygonal 3D meshes. This includes meshes with quads and/or higher order polygonal faces commonly used in authoring software, and non-manifold meshes with, potentially, degenerate faces. Additional functionalities, such as quality or spatial scalability, error resilience, and parallel encoding or decoding, are highly desirable (see Section 7 for more details).

AOM members are invited to submit proposals in response to this Call for Proposals (CfP). The submissions will be evaluated based on the objective and subjective metrics described in Section 6. Evaluation results will be anonymized before being shared within the Volumetric Visual Media (VVM) Working Group (WG). Proponents shall declare interest in making a submission by May 13th, 2023 and submit their results by September 5th, 2023. For a detailed timeline see Section 2. Descriptions of proposals shall be registered as input documents to the proposal evaluation meeting in October 2023. Proponents are required to attend that meeting to present their proposals.

The remainder of this document is organized as follows. Section 2 provides a detailed timeline of the CfP. Section 3 defines various terms used in this document. Section 4 describes the test dataset. Section 5 specifies the lossless and lossy test conditions considered in this CfP and describes the anchor generation process. The evaluation methodology, requirements, and submission rules are described in Sections 6, Section 7, and Section 8, respectively. Intellectual Property Rights (IPR) considerations and contact information are provided in Section 9 and Section 10, respectively. Detailed information about the dataset, scripts, and tools required for this CfP are described in the Annexes.

2 CfP timeline

The timeline for this CfP is described in Table 1.

Date	Action	By	Remarks
Mar. 15th, 2023	Release of the CfP	VVM WG	
Jun. 13th, 2023,	Declaration of intention of answering the CfP	Proponent	Registration to be made by email to the contact addresses listed in Section 10.
Jun. 16th, 2023	Proponents are provided with an individual account	VVM test coordinators	This account will serve to submit test material and as a registration confirmation.
Oct. 31st, 2023	Submission package to be uploaded	Proponent	See Section 8 for details.

Nov. 7th, 2023	Verification of MD5 checksums of decoded content	VVM test coordinator	
Nov. 10th, 2023	Generation and sharing of the rendered PNG content for all proposals based on the selected camera paths	VVM test coordinators	The rendered PNG files for each proposal will only be shared with their corresponding proponent.
Nov. 13th, 2023	Validation of the rendered PNG content	Proponent	Confirmation to be sent by email to the VVM test coordinators.
Nov. 13th, 2023	Proposal documentation submission	Proponent	To be uploaded to the proponent account (see Section 8) and submitted as an input contribution to the VVM October meeting.
Nov. 13th, 2023	Compilation of submitted objective results in a unique spreadsheet	VVM test coordinators	To be uploaded as an input contribution.
Nov. 13th- 25th, 2023	Subjective evaluation with naïve viewers	Universities	
Nov. 13th- 25th, 2023	Cross-check of objective results	VVM test coordinators	
Nov. 28th- 30th, 2023 (VVM face-to-face meeting)	Detailed technical presentation of proposals	Proponent	
Nov. 28th- 30th, 2023 (VVM face-to-face meeting)	Presentation of subjective results	Universities	
Nov. 28th- 30th, 2023 (VVM face-to-face meeting)	Selection of best proposal(s)	VVM WG	Selection of best proposal(s) (see Section 6). Selecting a single proposal is the ideal case. If no agreement can be achieved on the selection of the best proposal, the following options are possible: (1) combination of best proposals, (2) selection of a subset of proposals. The corresponding technical steps will be addressed in subsequent meetings.
Jan. 12th, 2024	Submission of encoder and decoder source code	Proponent(s)	Source code shall allow to reproduce test results in a bit

	of the selected proposal(s)		exact manner on the platform on which the proposal was submitted, and source code shall match with the submitted documentation.
Jan. 26th, 2024	Establishment of the first version of the VVM test model based on the selected proposal(s).	VVM WG	

Table 1: CfP timeline (Note: Anywhere on Earth time zone is used for all the deadlines).

3 Definitions

Attribute maps. Attribute maps are attributes of the static polygonal mesh surface stored as 2D images, such as colour texture images.

Connectivity information. The connectivity information of a polygonal mesh is specified by a vector of integers indicating for each face the number of its vertices, and by a vector of indices describing how to connect the mesh vertices to create faces.

Corner attributes. Corner attributes are specified by a set of binary, scalar, or vector values associated with the mesh face corners, together with a set of indices indicating the mapping between corners and corner attribute values. The corner attribute values have finite precision and range.

Edge attributes. Edge attributes specify different sets of indices indicating the mapping between edges and edge attribute values. They are specified by binary, scalar, or vector values associated with the mesh edges. The edge attribute values have finite precision and range.

Face attributes. Face attributes are specified by binary, scalar, or vector values associated with the mesh faces. The face attribute values have finite precision and range.

Geometry information. The geometry information is specified by a vector of positions in the 3D space, associated with the mesh vertices. The position of each vertex is described by three Cartesian coordinates (X, Y, Z) exhibiting finite precision and range.

Lossless connectivity compression. The connectivity information is losslessly compressed if and only if the reconstructed connectivity and the attribute indices are the same as those of the input mesh, up to a permutation of the vertices and a circular permutation of the indices within faces.

Lossless corner attribute compression. The corner attributes are losslessly compressed if and only if the reconstructed corner attribute values and indices are the same as those of the input mesh, up to a permutation of the vertices and a circular permutation of the indices within faces.

Lossless geometry compression. The geometry information is losslessly compressed if and only if the reconstructed vertex positions are the same as those of the input mesh, up to a permutation of the vertices.

Lossless mesh compression. A mesh is losslessly compressed if and only if its connectivity, geometry, and attributes are all compressed in a lossless manner.

Lossless vertex attribute compression. The vertex attributes are losslessly compressed if and only if the reconstructed vertex attributes are the same as those of the input mesh, up to a permutation of the vertices.

Lossy mesh compression. A mesh is compressed in a lossy manner if its connectivity, geometry, and/or attributes are not losslessly compressed.

Mapping information. The mapping information indicates a mapping between the surface of a mesh and a 2D region of the plane.

Mesh quality scalability. A compressed mesh bitstream supports quality scalability if it is structured such that decoding a portion of the bitstream results in a mesh of a certain geometry and/or attribute information precision (quality), which can be progressively refined by further decoding the bitstream. Attribute maps shall not be considered when assessing mesh quality scalability.

Mesh spatial scalability. A compressed mesh bitstream supports spatial scalability if it is structured such that decoding a portion of the bitstream results in a mesh of a certain resolution (*i.e.* number of faces or vertices), which can be progressively refined by further decoding the bitstream. Attribute maps shall not be considered when assessing mesh spatial scalability.

Static polygonal mesh. A static polygonal mesh is a surface representation that consists of different components: connectivity information, geometry information, zero or more attribute maps, and zero or more vertex, face, corner, and/or edge attributes.

Vertex attributes. Vertex attributes are specified by binary, scalar, or vector values associated with the mesh vertices. The vertex attribute values have finite precision and range.

4 Test material

4.1 Description of the source content

The test material to be used for the CfP is organized in six classes, some of which are further divided into sub-classes. The six classes are as follows:

- Class A includes video game meshes obtained by Digital Content Creation (DCC) tools. It is divided into two sub-classes: A1 for game assets and A2 for game characters.
- Class B is a collection of avatars captured as 3D scans of humans.
- Class C consists of meshes obtained by DCC tools and are suitable for online commerce applications.

- Class D consists of several professional 3D scans suitable for digital museum and cultural heritage applications. It is divided into two sub-classes: D1 for meshes composed of mainly triangular faces and D2 for meshes composed of mainly quadrilateral faces.
- Class E contains non-professional 3D scans of objects.
- Class F contains professional 3D scans from outdoor and indoor scenes suitable for virtual tour applications.

The list of test material and details on each mesh, including its type, generation process, vertex count, face count, and texture resolution, are provided in Annex A.

From this source dataset, the following three input categories are derived:

- Unconstrained input category C0,
- Constrained input category C1, and
- Single-connectivity input category C2.

C0, C1, and C2 will be used to evaluate the lossless and the lossy mesh compression technologies submitted as responses to this CfP (see Section 8 for details). The pre-processing schemes used to generate the input categories C0, C1, and C2 are described in Subsections 4.2, 4.3, and 4.4, respectively. The three input categories can be re-generated from the source content by using the scripts described in Annex B. A pre-generated version is available at:

s3://aom-vvm-datasets/spmc-cfp/input_categories

4.2 Unconstrained input category C0

The input category C0 corresponds to unconstrained input meshes, which are generated according to the pre-processing pipeline described in Figure 1.

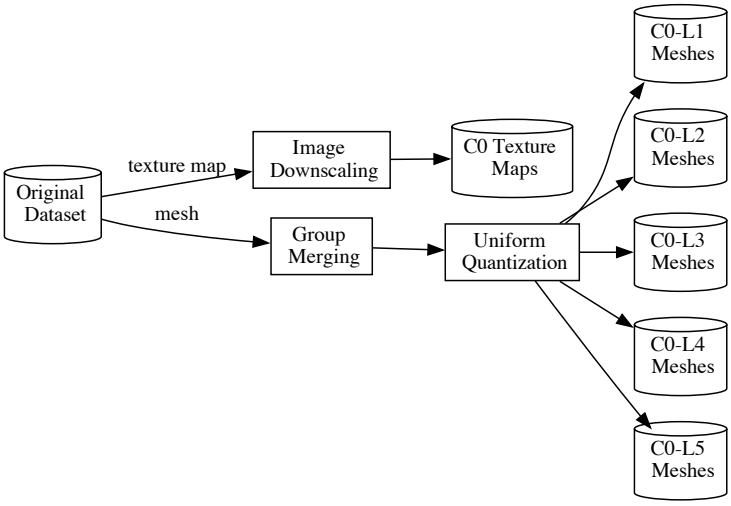

Figure 1: Generation process for the unconstrained input category C0.

The C0 generation process consists of the following pre-processing modules:

- “Group Merging”, which merges all the mesh groups into a single mesh group.
- “Uniform Quantization”, which applies uniform quantization to the mesh positions, texture coordinates, and normal vectors.

- "Image Downscaling". The texture map is optionally downscaled, while preserving the aspect ratio of the image, to guarantee that both horizontal and vertical dimensions do not exceed 4096 pixels.

Both the "Group Merging" and the "Uniform Quantization" processes are performed with the tool "mesh_quantize" (see Annex C). Five quantization levels L1, L2, L3, L4, and L5 are generated according to the quantization parameters reported in Table 2. The parameters qp, qt, and qn correspond to the bit depth that is used to quantize positions, texture coordinates, and normal vectors, respectively.

Quantization Level	Set of parameters	qp	qt	qn
L1	Lowest	9	8	6
L2	Lower	10	9	7
L3	Default	11	10	8
L4	High	13	12	9
L5	Highest	16	14	10

Table 2: Quantization parameters.

The "Image Downscaling" process is performed using the scripts described in Annex D. The resolutions of the original and the downscaled texture maps are reported in Annex A.

Note: Only uniform quantization is considered in this CfP. Alternative quantization techniques (*e.g.* octahedral normal quantization [7]) will be studied during the standard development phase.

4.3 Constrained input category C1

The input category C1 corresponds to constrained input meshes that were obtained by further processing C0 meshes (see Figure 2) so they conform to the following constraints:

1. The correspondence between position and non-position attribute indices should be a one-to-many mapping. That is, any non-position attribute index cannot be shared by different spatial vertices defined by position indices.
2. The connectivity for all attributes, defined by their associated indices, must be a 2-manifold.
3. The mesh has no degenerate faces.

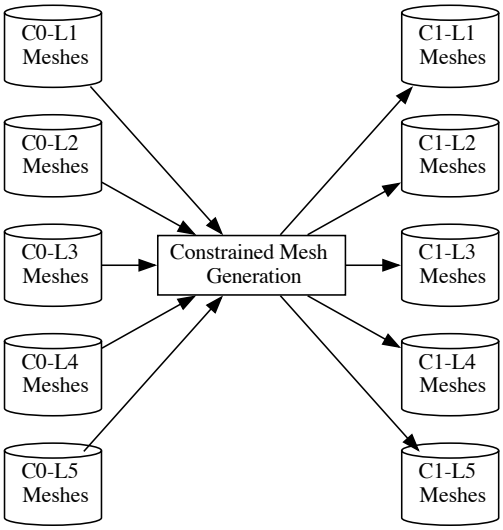

Figure 2: Generation process for the constrained input category C1.

The “Constrained Mesh Generation” pre-processing module converts non-manifold meshes into manifold ones by duplicating vertices and edges. Degenerate faces are removed. A one-to-many mapping between position and non-position attribute indices is enforced by duplicating attribute values. Attribute maps are left unchanged.

4.4 Single connectivity input category C2

The input category C2 corresponds to single connectivity triangular input meshes, which could be directly consumed for rendering purposes, without the need to apply any post-processing procedures. C2 meshes are generated by further processing C1 meshes according to the pre-processing pipeline described in Figure 3.

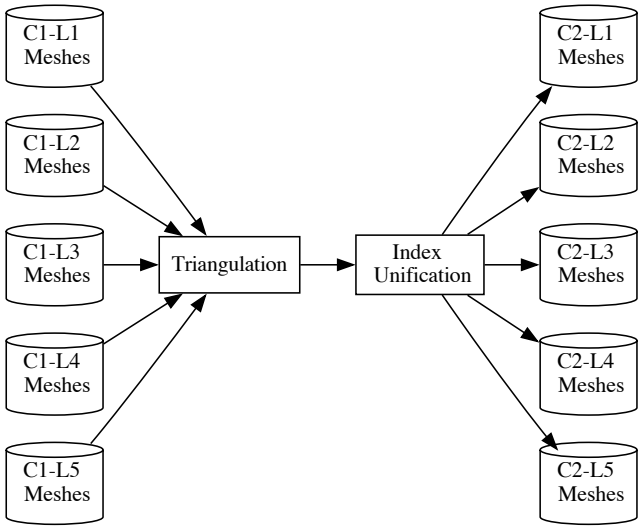

Figure 3: Generation process for the single connectivity input category C2.

The following pre-processing modules are applied:

- "Triangulation": triangulates polygonal faces to convert them to triangular ones.
- "Index unification": duplicates attribute values and generates a single set of indices shared by all attributes.

5 Test conditions and anchor generation

This CfP evaluates mesh compression technologies under two separate test conditions, lossless and lossy.

5.1 Lossless test condition

A formal definition of lossless mesh compression is provided in Section 3. It guarantees an exact reconstruction of the connectivity, the geometry, and the attribute information modulo a permutation of the mesh vertices, attributes, indices, and faces. Compression technologies evaluated under the lossless mesh compression test condition shall meet the requirements detailed in Section 7. The evaluation shall be performed according to the methodology described in Section 6.1.

5.2 Lossy test condition

Under the lossy test condition, proposals are allowed to modify the mesh connectivity, geometry, or attributes to achieve better rate distortion performances. Allowed pre-processing operations include and are not limited to:

- mesh re-parameterization,
- mesh decimation, and
- texture map re-sampling and downscaling.

Under this condition, the following mesh components are not encoded:

- normal vectors, and

- mapping information for non-textured meshes.

Compression technologies evaluated under the lossy test condition shall meet the requirements detailed in Section 7. Objective and subjective evaluation shall be performed according to the methodologies described in Section 6.2 and Section 6.3, respectively.

5.3 Lossless anchor generation

Anchors for input categories C1 and C2 were generated by using the modified version of the Draco compression library described in Annex F. For input Category C0, the ZLib-based mesh compression scheme was used to generate the lossless anchors. Annex F provides detailed information about the anchor bitstreams generation process.

The compressed anchor bitstreams and decoded meshes for the three test categories and for all quantization settings are available at:

s3://aom-vvm-datasets/spmc-cfp/lossless_anchor

Results of the lossless anchor are available in the Excel spreadsheet lossless_reporting_template.xlsm provided with this CfP.

5.4 Lossy anchor generation

The lossy anchor generation framework is depicted in Figure 4. The lossy anchor bitstreams are generated by processing the C0-L5 input category meshes (see Section 6). The lossy anchor generation includes steps of decimation, Draco encoding and decoding, and rendering of the content using the decoded object, decoded texture, and a camera path. Additional modules, such as mesh dequantization and triangulation, are applied to guarantee an input that satisfies the requirements of the mesh decimation module. The texture is further downscaled and compressed as described in Annex I. Annex H provides detailed information about the anchor generation process. The list of tools, versions, download locations, and examples of command lines are provided in Annex I. The encoding parameters for the lossy anchor were selected according to the process described in Annex L.

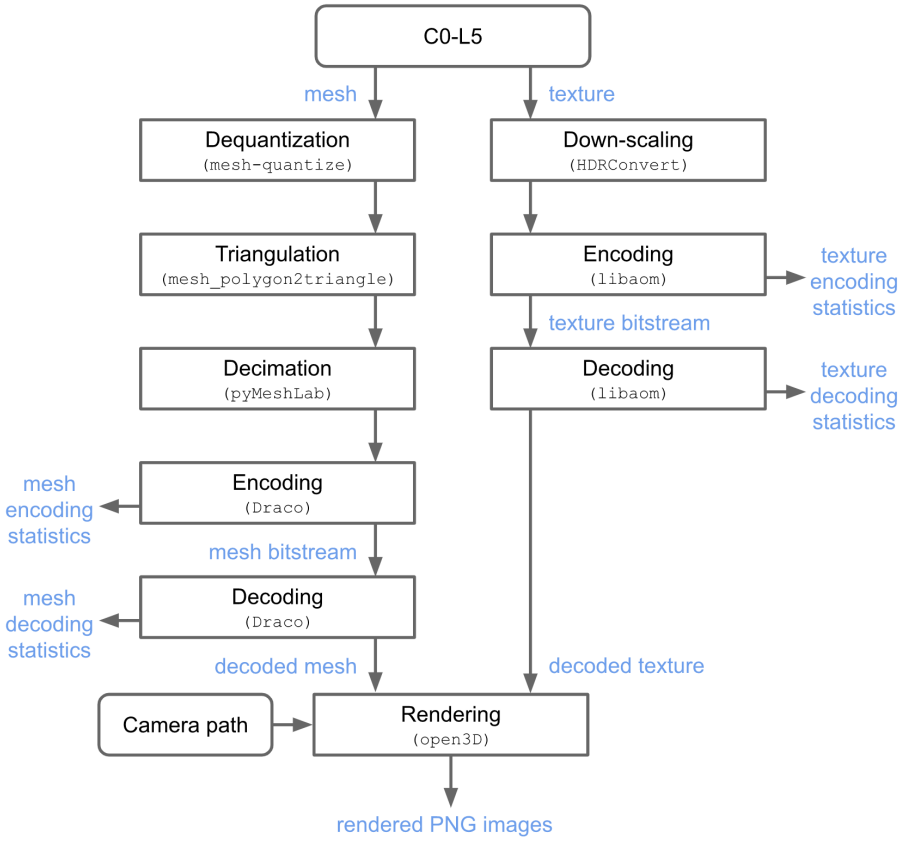

Figure 4: Lossy anchor framework.

6 Evaluation methodology

The lossless mesh compression technologies will be assessed solely based on the objective evaluation process described in Section 6.1. The lossy mesh compression technologies will be assessed based on both objective (see Section 6.2) and subjective criteria (see Section 6.3).

6.1 Objective evaluation for lossless condition

Figure 5 shows the evaluation process for the lossless mesh compression test condition.

The compressed bitstream shall be obtained by feeding the input mesh to the “Encoder” module. Besides generating the compressed bitstream, the encoder is required to generate position and attribute reordering information, which specifies for each position and attribute value the index of the corresponding attribute value in the input mesh. The reordering information shall conform to the JSON scheme described in Annex G. The decoded mesh shall be obtained by feeding the compressed bitstream to the “Decoder” module. The “Encoder” and “Decoder” modules are required to report the following statistics for each attribute separately:

- Compressed bitstream size, and
- Encoding and decoding time excluding any I/O operations (see Annex M).

The reported statistics should include both attribute values and attribute indices.

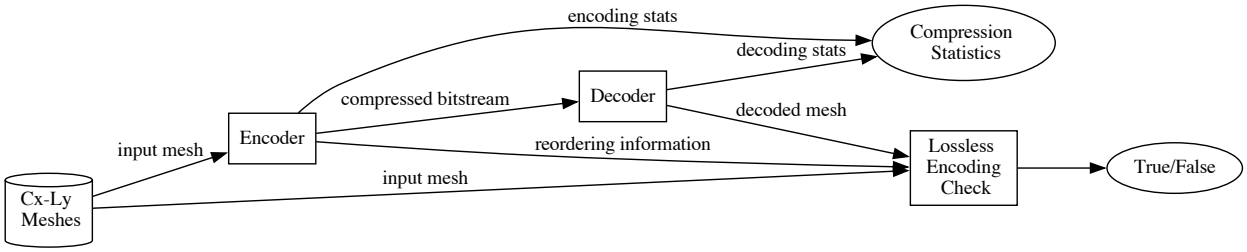

Figure 5: Evaluation process for the lossless mesh compression test condition, where Cx corresponds to C0, C1, or C2 and Ly corresponds to L1, L2, L3, L4, or L5.

The input mesh, the reordering information, and the decoded mesh are then fed to the "Lossless Encoding Check" module (see Annex G), which validates whether the mesh was losslessly encoded or not. Pre and post-processing of the input and output meshes are not allowed. For instance, the following operations are not allowed:

- welding vertices on the encoder side and applying vertex unification on the decoder side,
- converting non-manifold meshes to manifold meshes for test category C0, and
- generating new indices to guarantee 1-to-n mapping between attribute and position indices.

6.2 Objective evaluation for lossy condition

Model-based objective metrics are applied to compare the decoded mesh object and its associated decoded texture with the original C0-L5 content. Both the geometry and the texture qualities will be evaluated.

The evaluation will be made according to a set of performance indicators (see Annex K), split into two categories:

- Primary performance indicators, which are considered for decision making.
- Additional performance indicators, which are considered for in-depth understanding of performance.

The results will be reported in terms of BD-Rate gains [8][9], where the distortion will be computed with different quality metrics and the rate will depend on the considered metric. In addition, RD curves will be plotted. The rates will be reported as bits per frame.

The list of primary performance indicators includes:

- BD-Rate of d2PSNR: the geometry distortion is computed by the model-based point-to-plane metric [10], and the rate includes the size of the compressed positions. It is computed for all meshes.
- BD-Rate of lumaPSNR: the colour distortion is computed by the model-based luma PSNR metric of the texture map and the rate includes the size of the compressed positions, texture coordinates, and texture map. It is computed for all textured meshes.

The list of additional performance indicators includes:

- BD-Rate of d1PSNR: the geometry distortion is computed by the model-based point-to-point metric [10] and the rate includes the size of the compressed positions. It is computed for all meshes.

- BD-Rate of cbPSNR: the colour distortion is computed by the model-based chroma cbPSNR metric of the texture map and the rate includes the size of the compressed positions, texture coordinates, and texture map. It is computed for all textured meshes.
- BD-Rate of crPSNR: the colour distortion is computed by the model-based chroma crPSNR metric of the texture map and the rate includes the size of the compressed positions, texture coordinates, and texture map. It is computed for all textured meshes.
- Triangle count ratio, representing the ratio between the number of triangles of the input mesh (*i.e.* C0-L5 mesh) and the one generated by decoding the proposed compressed bitstream. The number of triangles should be computed as described in Annex M. It is computed for all meshes.
- Geometry encoder runtime ratio, representing the encoder runtime (excluding any IO operations) ratio between the anchor and the proposal for the geometry. It is computed for all meshes.
- Texture encoder runtime ratio, representing the encoder runtime (excluding any IO operations) ratio between the anchor and the proposal for the texture. It is computed for all textured meshes.
- Geometry decoder runtime ratio, representing the decoder runtime (excluding any IO operations) ratio between the anchor and the proposal for the geometry. It is computed for all meshes.
- Texture decoder runtime ratio, representing the decoder runtime (excluding any IO operations) ratio between the anchor and the proposal for the texture. It is computed for all textured meshes.

The renderer runtime ratio is an optional performance indicator representing the renderer runtime ratio between the anchor and the proposal. It will exclude the IO operations, as reflected in the renderer script (see vvmRenderer.py described in Annex J). It is computed for all meshes belonging to the viewing set (see Section 6.3.1), based on the camera path implemented in the renderer script.

More details on the computation of the different performance indicators are given in Annex K.

6.3 Subjective evaluation

The subjective evaluation will be the primary attribute for the decision-making process for the lossy scenario for meshes belonging to the viewing set (a subset of the VVM dataset). For all other meshes, the objective criteria (see Section 6.2) will be used.

The overall process, information on test laboratories and test coordinators, and the reporting of the subjective tests is presented in this section. More details are provided in [14], on the laboratories setup, on how the results of the two laboratories will be merged, and on the level of content overlap in the tests.

6.3.1 Content

The subjective evaluation will be performed on the viewing set. The selected content is provided in Table 3. Meshes belonging to the viewing set were selected to ensure most of the classes are represented, while exhibiting various mesh intrinsic characteristics (*e.g.* type of polygonal faces and vertex count). In addition, the selected content has visual characteristics that are appropriate for subjective evaluation.

Class	Id	Name
A2	9	grey_knight
	11	mira_w_gun
B	13	frederic_fr00001
	16	nathalie_fr00036
D1	25	apollo_11
	26	apothecary_vase
	27	orbiter_space_shutter
	30	zakopane_chair
	32	hussar_on_horseback
D2	40	buste_cuirasse_de_marc_aurele_age
E	43	dead_rose_smallCCremoved
	46	luna_lionfish
F	51	police_station

Table 3: Viewing set (mesh, and class).

The subjective evaluation will be performed on four target qualities denoted as TQ1, TQ2, TQ3 and TQ4 (from highest to lowest quality).

6.3.2 Rendering

The videos will be rendered from the bitstreams submitted as a response to the CfP. The renderer described in Annex J will be used with the following characteristics:

- Uniform gray (128, 128, 128) background, with uniform dark gray floor (64, 64, 64).
- Resolution of 1080p at 60 fps, yuv 420 10 bits format (see Annex I).

The camera paths will be selected by the Universities and will not be known by the proponents before the bitstream submission deadline, to avoid any kind of optimization.

The generated camera paths will meet the following rules:

- The duration of the rendered video will be between 10 seconds and 20 seconds.
- The camera path can include any kind of rotation.
- The camera path can include any kind of translation.
- The camera path can start from any viewpoint in the 3D space.
- The motion speed (rotations and translations) is maintained low to guarantee a comfortable viewing experience and reliable scoring.
- The rendered video can include breaks (no motion).

6.3.3 Subjective tests

The subjective tests will be conducted in laboratory with naïve viewers, following the ACR-HR methodology with the 11-grade scale [11]. At least 20 valid scores (obtained after outlier removal) will be used to compute the Mean Opinion Score (MOS).

The rendered PNG files, corresponding to the camera path generated by the Universities, will be released to the proponent (after the bitstreams submission deadline, and before starting the viewing test), for cross-check by the proponent, to ensure the correctness of the rendered content. The Python script used to generate the paths, modified from the Python script vvmRenderer.py described in Annex L, will be released to the proponents.

6.3.4 Test laboratories

The subjective evaluation of the submitted proposals will be carried out by two test laboratories from following universities:

- Kingston University, Prof. Maria Martini (m.martini@kingston.ac.uk)
- Waterloo University, Prof. Zhou Wang (zhou.wang@uwaterloo.ca)

6.3.5 Reporting of the subjective results

The reporting of the subjective tests will be uploaded as an input contribution to the VVM face-to-face October meeting. It shall include at least:

- Graphs representing the MOS versus bitrate curves for each mesh of the viewing set, comparing the anchor and the proposal.
- BD-Rate using the MOS for each mesh of the viewing set, comparing the anchor and the proposal.
- A CSV or Excel file containing the MOS and confidence intervals for each mesh of the viewing set.
- A CSV or Excel file containing the raw scores of each viewer for each mesh of the viewing set.
- Detailed explanation on how the outlier's removal has been performed.
- Detailed explanation on how the data were aligned and merged between the two laboratories.

7 Requirements

Submissions to this CfP shall meet the mandatory requirements (see Section 7.1) and are strongly encouraged to also meet the optional requirements described in Section 7.2.

7.1 Mandatory requirements

Submitted technologies shall comply with the following mandatory requirements:

- Deliver efficient lossless compression or lossy compression of static polygonal meshes according to the definitions provided in Section 3.
- Support attributes associated with the mesh vertices and corners.
- Support integer input and output values for relevant attributes.

7.2 Optional requirements

Submitted technologies are encouraged to comply with the following optional requirements:

- Support meshes with non-manifold geometry and degenerate faces.
- Support attributes associated with the mesh faces and edges.
- Support spatial and quality scalability.
- Support error resilience to cope with non-reliable transmission and storage.
- Support parallel encoding and decoding.
- Support efficient hardware and software implementations.
- Support tools that enable encoding and decoding with low complexity and low latency.
- Minimize the number of rendering draw calls.

Low-level programming optimizations, such as assembly code and intrinsic and external compression libraries, are discouraged. If any such optimization is implemented, then the rationale for, and extent of the optimization shall be described.

8 Submission rules

This section describes the set of mandatory rules that responses to this CfP shall follow. The rules are organized into four categories: general, lossless mesh compression, lossy compression, and package submission rules.

8.1 General submission rules

The submitted proposals are required to:

- Store decoded meshes in the OBJ format with integer values for all attributes.
- Avoid training large entropy coding tables, VQ codebooks, *etc*. If any processes in the encoder or decoder are designed using training data, then the coding test set shall not be used as part of the training set.
- Report both the total and per attribute compressed bitstream size (in bits) and the number of vertices.
- Report encoding and decoding logs as described in Annex M.

8.2 Submission rules for lossless mesh compression

For the lossless compression test condition, the submitted proposals are required to:

- Support meshes in at least one of the C0, C1, and C2 input categories.
- Support all quantization levels (*i.e.* L1, L2, L3, L4, and L5) associated with all of the supported input categories.
- Support compression of mesh geometry and attributes, including positions, texture coordinates, and normal vectors.
- Pass the lossless encoding check described in Section 6.1.

8.3 Submission rules for lossy mesh compression

For the lossy compression test condition, the submitted proposals are required to:

- Support all meshes in input category C0 with quantization level L5.
- Support compression of mesh geometry, texture coordinates, and texture maps.
- Use the HDRConvert tool and the configuration files described in Annex D when upscaling or downscaling texture maps.
- Perform colour space conversion of texture maps from RGB to YUV 420 BT.709 prior to AV1 compression by using the HDRConvert tool and the configuration files described in Annex I.
- Leverage the AV1 encoder and decoder implementation and encoding configuration settings described in Annex I. Proponents are allowed to change only the quantization level (i.e. cq-level parameter) and the input texture intrinsic parameters (e.g. resolution, bit depth).
- Perform colour space conversion of the decoded texture maps from YUV 420 BT.709 to RGB by using the HDRConvert tool and the configuration files described in Annex I. The decoded texture maps shall be stored in the PNG format (i.e. same format as C0-L5).
- Use the same renderer as the one used by the anchor, as described in Annex J.
- Meet the objective quality targets with an allowed variance of +/- 2 dB d2PSNR for objects without texture and +/- 1 dB lumaPSNR for objects with texture with respect to the anchor. The target qualities are listed in Annex L.
- Exclude the size of the mapping information from the BD-Rate calculation for non-textured objects.

8.4 Package submission rules

Proponents shall deliver their submissions according to the following folder structure:

- "app": containing the binary decoder executable (*i.e.* Windows, MacOS, or Linux executable), the configuration files, if required, and the usage documentation allowing for decoding the bitstreams.
- "doc": containing the documents specified in Section 8.4.1.
- "src": place holder for the potential delivery of source code (see Section 8.4.2).
- "enc": containing encoding related files for the lossless and/or the lossy test conditions described in Section 8.4.3 and Section 8.4.4, respectively.
- "scripts": containing a bash script "decode.sh" (without parameters) needed for decoding the compressed bitstreams and for generating all the decoding related files (see Section 8.4.3 and Section 8.4.4).
- "dec": all decoding related files for the lossless and/or the lossy test conditions described in Section 8.4.3 and Section 8.4.4, respectively.
- "stats": containing the proposals statistics as described in Section 8.4.5.

8.4.1 Technical description

Proponents shall provide full technical descriptions of their proposals. The technical descriptions should provide sufficient information for understanding and implementing the proposed technologies. The descriptions shall include all data processing paths and individual data processing components used to generate the bitstreams. The technical descriptions shall contain information suitable to assess the complexity of the implementation of the technology and how it addresses the requirements reported in Section 7.

8.4.2 Source code

Proponents are advised that, upon acceptance for further evaluation, it will be required that certain parts of any proposed technology be made available in source code format to participants in the core experiments process and for potential inclusion in the prospective standard as reference software. When a particular technology is a candidate for further evaluation, commitment to provide such software is a condition of participation. The software shall produce identical results to those submitted to the test on the same platform that the technology was submitted. Additionally, submission of improvements (*e.g.* bug fixes) is strongly encouraged.

8.4.3 Folder structure and naming conventions for the lossless test condition

Under the lossless test condition, the "enc" and "dec" folders shall follow the following folder structure:

- enc
    - P{pn}T0C{x}L{y}: folder for input category C{x}L{y}
        - {m-id}_{m-name}
            - license.txt/pdf: license for mesh M{m-id},
            - P{pn}T0C{x}L{y}M{m-id}.bin: compressed bitstream,
            - P{pn}T0C{x}L{y}M{m-id}_reorder.json: reordering information file (see Annex G),

            - P{pn}T0C{x}L{y}M{m-id}_enc.json: encoding log file (see Annex M).
- dec
    - P{pn}T0C{x}L{y}: folder for input category C{x}L{y}
        - {m-id}_{m-name}
            - license.txt/pdf : license for mesh M{m-id},
            - P{pn}T0C{x}L{y}M{m-id}_cmp.json: mesh comparison log file (see Annex G),
            - P{pn}T0C{x}L{y}M{m-id}_dec.json: decoding log file (see Annex M),
            - P{pn}T0C{x}L{y}M{m-id}.obj: decoded obj file

With:

- pn: proponent number assigned by the VVM test coordinators.
- x: input category index (0, 1, 2)
- y: quantization level index (1, 2, 3, 4, 5)
- m-id: mesh Id as specified in Annex A.
- m-name: mesh name as in Annex A.

Figure 6 shows an example of folder structure for the encoding and decoding related files.

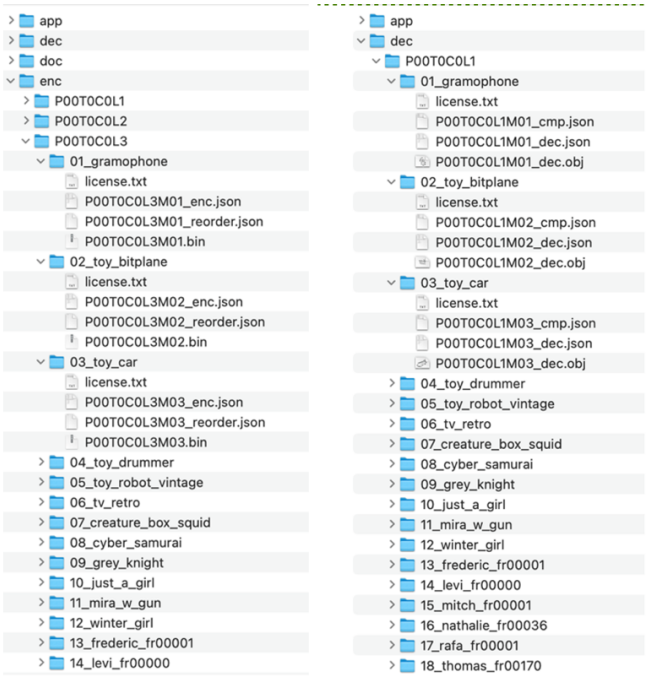

Figure 6: Folder structure and naming conventions for the lossless test condition.

8.4.4 Folder structure and naming conventions for the lossy test condition

Under the lossy test condition, the “enc” and “dec” folders shall follow the following folder structure:

- enc

    - P{pn}T1
        - {m-id}_{m-name}
            - license.txt/pdf : license for mesh M{m-id},
            - P{pn}T1Q{q}M{m-id}_mesh.bin: compressed mesh bitstream for quality target q,
            - P{pn}T1Q{q}M{m-id}_texture.bin: compressed texture bitstream for quality target q,
            - P{pn}T1Q{q}M{m_id}.mtl: the material file,
            - P{pn}T1Q{q}M{m_id}_enc.json: encoding log file.
- dec
    - P{pn}T1
        - {m-id}_{m-name}
            - license.txt/pdf : license for mesh M{m-id},
            - P{pn}T1Q{q}M{m-id}_dec.obj: decoded mesh obj file for quality target q,
            - P{pn}T1Q{q}M{m-id}_dec.png: decoded texture in PNG format for quality target q,
            - P{pn}T1Q{q}M{m-id}_dec.mtl: material file used in the decoded mesh,
            - P{pn}T1Q{q}M{m-id}_dec.json: decoding log file.

With:

- pn: proponent number assigned by the VVM test coordinators.
- q: quality target index (TQ1, TQ2, TQ3, TQ4) as specified in Annex L.
- m-id: mesh Id as specified in Annex A.
- m-name: mesh name as in Annex A.

Figure 7 shows an example of folder structure for the encoding and decoding related files.

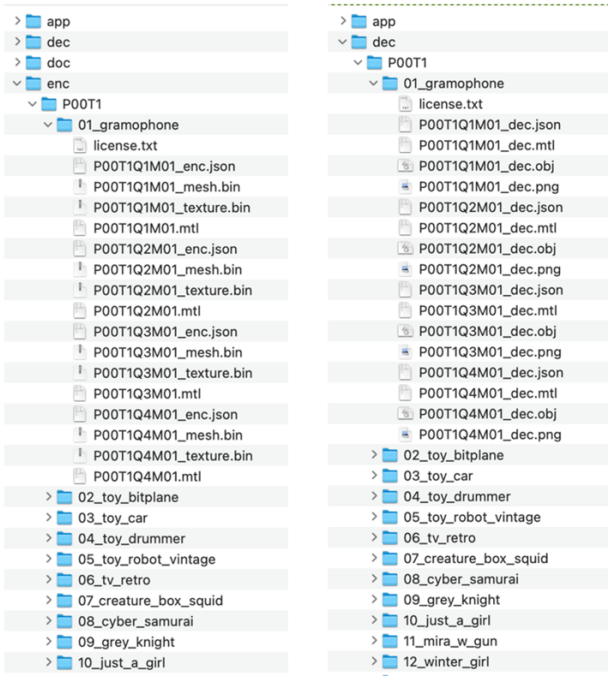

Figure 7: Folder structure and naming conventions for the lossy test condition.

8.4.5 Encoding and decoding statistics

Results of the objective tests shall be reported by using the reporting templates provided as part of this CfP. Proponents shall use the Excel spreadsheets lossless_reporting_template.xlsm, and lossy_reporting_template.xlsm for the lossless and lossy test conditions, respectively. The spreadsheets shall be the reference for information to be reported.

9 Subsequent provision of source code and IPR considerations

Participants in Alliance for Open Media Working Groups have adopted the Alliance for Open Media Patent License 1.0. This is intended to fulfill their commitments to make available their Essential Claims, as defined in the W3C Patent Policy, in Final Deliverables adopted by that Working Group under the W3C RF licensing requirements as if that Final Deliverable was a W3C Recommendation.

Software released by the Alliance for Open Media is made available under a combination of the following licenses: BSD 3-Clause Clear License and Alliance for Open Media Patent License 1.0.

Additional legal and software information can be found here: License | Alliance for Open Media (aomedia.org).

10 Contacts

VVM chairs:

- Shan Liu, Tencent (wg-vvm-chair@aomedia.org)
- Khaled Mammou, Apple (wg-vvm-chair@aomedia.org)

11 References

[1] https://en.wikipedia.org/wiki/Wavefront_.obj_file.
[2] https://registry.khronos.org/glTF/specs/2.0/glTF-2.0.html.
[3] https://graphics.pixar.com/usd/release/index.html.
[4] https://www.opengl.org.
[5] https://learn.microsoft.com/en-us/windows/win32/direct3d.
[6] https://www.vulkan.org.
[7] https://github.com/google/draco.
[8] G. Bjontegaard, Calculation of average PSNR differences between RD-curves, ITU-T SG16/Q6 VCEG-M33, 2001.
[9] S. Pateux, J. Jung, An excel add-in for computing Bjontegaard metric and its evolution, ITU-T SG16/Q6 VCEG-AE06, 2007.
[10] D. Tian, H. Ochimizu, C. Feng, R. Cohen and A. Vetro, "Geometric distortion metrics for point cloud compression," 2017 IEEE International Conference on Image Processing (ICIP), Beijing, China, 2017, pp. 3460-3464, doi: 10.1109/ICIP.2017.8296925.
[11] ITU-T P.910 (2008), Subjective video quality assessment methods for multimedia applications.
[12] Michael Garland and Paul S. Heckbert, “Surface simplification using quadric error metrics”, in Proceedings of the 24th annual conference on Computer graphics and interactive techniques (SIGGRAPH '97). ACM Press/Addison-Wesley Publishing Co., USA, 209–216. https://doi.org/10.1145/258734.258849.
[13] J. Jung, M. Wien, V. Baroncini, Guidelines for remote experts viewing sessions, ISO/IEC JTC 1/SC 29/AG 5 N00040, Online – October 2021.
[14] VVM test coordinators, “Test conditions for the subjective evaluation of the SPMC CfP”, VVM-2023-022i input document, March 2023.

Annex A: Detailed test content description

Table 4 reports the following information for each mesh:

- “Class”: the mesh class.
- “Id”: the mesh index.
- “Name”: the name of the mesh.
- "Type”: the kind of polygons that are present in the mesh.
- “Creation process”: the way the mesh has been produced (*i.e.* “DCC” for content obtained from DCC tools, and “3D Scan” for 3D human scans.
- “Face count”: the number of faces of the mesh.
- “Source texture resolution”: the resolution of the source texture of the mesh, in pixels.
- “C0 texture resolution”: the resolution of the C0 texture of the mesh, in pixels.

Class	**Id** (m-id)	**Name** (m-name)	**Type**	**Creation process**	**Face count**	**Source texture resolution**	**C0 texture resolution**
A1	1	gramophone	Quad	DCC	87384	-	-
	2	toy_bitplane	Quad	DCC	26496	-	-
	3	toy_car	Quad	DCC	29730	-	-
	4	toy_drummer	Quad	DCC	23370	-	-
	5	toy_robot_vintage	Quad	DCC	42212	-	-
	6	tv_retro	Quad	DCC	36640	-	-
A2	7	creature_box_squid	Tri	DCC	20140	4096×4096	4096×4096
	8	cyber_samurai	Tri-Quad	DCC	208944	12288×8192	4096×2730
	9	grey_knight	Poly	DCC	35771	4096×4096	4096×4096
	10	just_a_girl	Tri	DCC	77725	4096×3072	4096×3072

	11	mira_w_gun	Tri	DCC	77178	4096×6144	2730×4096
	12	winter_girl	Poly	DCC	92637	14338×12288	4096×3510
B	13	frederic_fr00001	Tri	3D Scan	49908	4096×4096	4096×4096
	14	levi_fr00000	Tri	3D Scan	40040	4096×4096	4096×4096
	15	mitch_fr00001	Tri	3D Scan	30000	4096×4096	4096×4096
	16	nathalie_fr00036	Tri	3D Scan	30000	4096×4096	4096×4096
	17	rafa_fr00001	Tri	3D Scan	40000	4096×4096	4096×4096
	18	thomas_fr00170	Tri	3D Scan	30000	4096×4096	4096×4096
C	19	chair_swan	Quad	DCC	24708	-	-
	20	cup_saucer_set	Quad	DCC	21396	-	-
	21	flower_tulip	Quad	DCC	62224	-	-
	22	motorcycle	Tri-Quad	DCC	30000	2048×3072	2048×3072
	23	teapot	Quad	DCC	103696	-	-
	24	wateringcan	Quad	DCC	36032	-	-
D1	25	apollo_11	Tri	3D Scan	721399	12288×8192	4096×2730
	26	apothecary_vase	Tri	3D Scan	60002	8192×8192	4096×4096
	27	orbiter_space_shutter	Tri	3D Scan	150000	4096×4096	4096×4096

	28	piggy_bank	Tri	3D Scan	127393	8192×8192	4096×4096
	29	ware_bowl	Tri	3D Scan	64000	4096×4096	4096×4096
	30	zakopane_chair	Tri	3D Scan	141569	4096×4096	4096×4096
	31	heliostat	Tri-Quad	3D Scan	325374	8192×8192	4096×4096
	32	hussar_on_horseback	Tri-Quad	3D Scan	412947	4096×6144	2730×4096
	33	violin	Tri-Quad	3D Scan	459874	4096×4096	4096×4096
	34	electrodynamic	Poly	3D Scan	255964	8192×16384	2048×4096
	35	marble_mortar	Tri-Quad	3D Scan	29654	8192×16384	2048×4096
D2	36	butterflies_collection	Tri-Quad	3D Scan	445915	12288×8192	4096×2730
	37	armillary_sphere_1771	Tri-Quad	3D Scan	308898	8192×4096	4096×2048
	38	wooden_gramophone	Tri-Quad	3D Scan	643464	16384×16384	4096×4096
	39	stereoscopic_cam	Tri-Quad	3D Scan	318100	8192×8192	4096×4096
	40	buste_cuirasse_de_marc_aurele_age	Quad	3D Scan	489920	2048×2048	2048×2048
	41	candle_stick	Quad	3D Scan	271800	4096×4096	4096×4096
	42	promo_ashtray	Quad	3D Scan	54344	4096×4096	4096×4096
E	43	dead_rose_smallCCremoved	Tri	3D Scan	49940	8192×8192	4096×4096
	44	green_tree_frog	Tri	3D Scan	1333419	8192×8192	4096×4096

	45	japanese_spiny_lobster	Tri	3D Scan	864576	4096×8912	2048×4096
	46	luna_lionfish	Tri	3D Scan	546363	4096×8192	2048×4096
	47	sakura_cherry_blossom	Tri	3D Scan	1067755	8192×8192	4096×4096
	48	wiz_boots	Tri	3D Scan	300000	4096×8192	2048×4096
F	49	cela_ruins_of_a_nuns_cell	Poly	3D Scan	709020	8192×8192	4096×4096
	50	heilig_grab_kapelle	Tri	3D Scan	555924	8192×8192	4096×4096
	51	police_station	Tri	3D Scan	1538567	8192×8192	4096×4096
	52	the_great_drawing_room	Tri	3D Scan	999999	8192×8192	4096×4096
	53	the_serving_room	Tri	3D Scan	999999	8192×8192	4096×4096

Table 4: VVM dataset detailed description.

Annex B: Generation of the input categories C0, C1, and C2

The three input categories C0, C1, and C2 can be generated by using the python script "generate_input_categories.py" (see Annex I). The script generates meshes for input categories, downscales images if necessary, stores all images to one centralized location, and updates material files with correct pointers in input categories.

The input path should point to a local copy of the static meshes stored on S3 bucket under the following URI: s3://aom-vvm-datasets/StaticMeshes/. The dataset can be downloaded following the instructions from the Content repository at:

https://gitlab.com/AOMediaVVM/content-conditions/content

or by running the following commands (if aws CLI tools are already set up):

```
aws --profile aom-vvm s3 sync s3://aom-vvm-datasets/StaticMeshes datasets_local
```

The binary path should contain the following tools:

- Mesh Quantization (see Annex C)
- Draco (see Annex I)
- Lossless Compression Validation (see Annex G)
- Image Downscaling (see Annex D)

The config path should contain the configuration file used for image downscaling, as detailed Annex D.

After running the script, the following folders will appear under the output_path:

- C0-L1, C0-L2, C0-L3, C0-L4, C0-L5
- C1-L1, C1-L2, C1-L3, C1-L4, C1-L5
- C2-L1, C2-L2, C2-L3, C2-L4, C2-L5
- textures

Each Cx-Ly folder contains 53 mesh folders, named as m-id_m-name, where indices for meshes are detailed in Annex A. Figure 8 shows an example of file structures of an input category. Each mesh folder contains:

- m-name.obj: the input mesh for codecs.
- m-name.json: the quantization information for inverse-quantization.
- m-name.mtl: the material file for meshes with textures, it points to the corresponding image under the textures folder.
- license.txt/pdf: license information file.

The textures folder is the centralized location for all textures, which contains both images that need downscaling and those don't. The folder contains 42 mesh folders, named as m-id_m-name, where indices for meshes are detailed in Annex A. Each mesh folder contains:

- m-name.png: the texture image used by the mesh, either downscaled or not.
- license.txt/pdf: license information file.

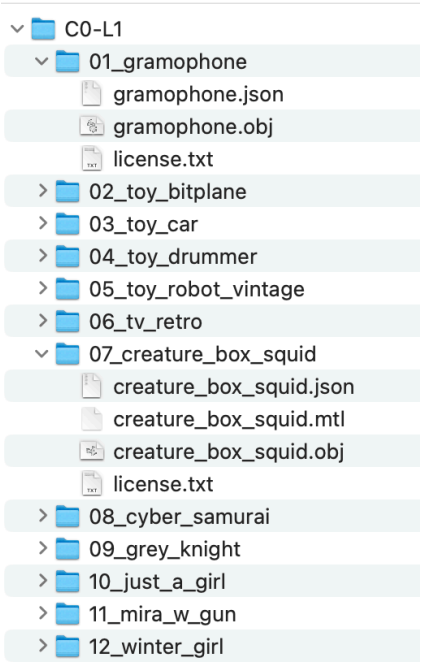

Figure 8: Example of file structures of input categories

Annex C: Mesh quantization

Quantization

The original dataset was quantized by using the mesh quantization tool "mesh-quantize.py" (see Annex I). The quantization tool converts the positions, the texture coordinates, and the normals of an input mesh to integer values by applying a uniform quantization. More precisely the quantized integer attribute values $(a_i)_{i \in \{0,\dots,A-1\}}$ (with A being the number of attributes) are obtained by processing the floating-point attribute values $(\alpha_i)_{i=0\dots A-1}$ as follows:

$$a_i = \left\lfloor \frac{\left(\alpha_i - \alpha^{min}\right) \times (2^q - 1)}{\max(\alpha^{max} - \alpha^{min})} + \frac{1}{2} \right\rfloor, \forall\, i \in \{0, \dots, A-1\},$$

where:

- $\lfloor x \rfloor$ is the floor function that takes as input a real number x, and gives as output the greatest integer less than or equal to x,
- q is the bit depth of the quantized values, and
- α^{max} and α^{min} are the maximum and minimum attribute values, respectively.

Both input and output meshes are stored in OBJ format. Besides the quantized output mesh, the tool dumps also quantization information in JSON format (see Figure 9).


```
{
  "attributes": [
    {
      "max": [
        194.98550415039063,
        317.681884765625,
        196.73941040039063
      ],
      "min": [
        -195.5240936279297,
        43.57830047607422,
        -195.44029235839844
      ],
      "num_components": 3,
      "scale": 10.441641807556152,
      "type": "POSITION",
      "vector_count": 382273
    },
    {
      "max": [
        0.9998329877853394,
        0.9995999932289124
      ],
      "min": [
        0.00016700000560376793,
        0.0002500000118743628
      ],
      "num_components": 2,
      "scale": 1023.341796875,
      "type": "TEX_COORD",
      "vector_count": 483572
```

```
      },
      {
        "max": [
          1.0,
          1.0,
          1.0
        ],
        "min": [
          -1.0,
          -1.0,
          -1.0
        ],
        "num_components": 3,
        "scale": 511.5,
        "type": "NORMAL",
        "vector_count": 493000
      }
    ]
}
```


Figure 9: Example of quantization information.

Inverse quantization

The inverse quantization process consists in converting the quantized integer attribute values $(a_i)_{i \in \{0,\dots,A-1\}}$ to reconstructed floating point attributes $(\hat{\alpha}_i)_{i=0\dots A-1}$ as follows:

$$\hat{\alpha}_i = \frac{a_i \times \max\left(\alpha^{max} - \alpha^{min}\right)}{(2^q - 1)} + \alpha^{min}, \forall\, i \in \{0, \dots, A-1\}.$$

The inverse quantization process requires the quantization information stored in the JSON file produced during the quantization process.

Annex D: Image downscaling

The original texture maps can be downscaled by using the Python script “downscale_images.py” (see Annex I). The “downscale_images.py” script leverages the HDRTool (see Annex I).

There is a total of 53 static meshes, out of which 42 contain textures. Among those meshes with textures, 36 of them require image downscaling. After running the script, a total of 36 folders will appear under the output path. Each folder is named as m-id_m-name, where indices for meshes are detailed in Annex A. Each mesh folder contains:

- m-name.png: the downscaled image.
- license.txt/pdf: license information.

Annex E: Lossless anchors for input categories C0, C1 and C2

The zlib-based mesh codec "mesh-zlib-codec" (see Annex I) was used to generate the lossless anchor bitstreams for the input category C0. The modified version of the Draco compression library (see Annex I) was used for input categories C1 and C2. The following changes to the Draco library were introduced to meet the lossless encoding requirements (see Section 7) and rules (see Section 8):

- Disable Draco internal quantization by loading all attributes as integer attributes instead of floating-point attributes.
- Increase the maximum corner count from 8 to 255 to support all the meshes included in the VVM evaluation dataset.
- Disable input values deduplication.
- Dump attribute reordering information as described in Annex G.
- Report encoding and decoding statistics (*e.g.* encoding and decoding time and compressed bitstream size) for each attributes separately.
- Expose the encode option *use_single_connectivity* to enforce single connectivity output for category C2.

Annex F: Lossless anchors generation process

The lossless anchor bitstreams for the three input categories C0, C1, and C2 can be generated by using the python script “generate_lossless_anchors.py" (see Annex I).

After running the script, the following folders will be generated under the output_path:

- “enc”: same structure described in Section 8.4.3,
- “dec”: same structure described in Section 8.4.3, and
- “stats”: performance statistics in csv format.

Annex G: Lossless compression validation

The Mesh Compare tool shall be used to validate whether the codec under evaluation is lossless or not. The tool takes as input:

- A mesh *M* belonging to one of the input categories Cx-Ly,
- A JSON file containing the encoding reordering information (see Figure 10),
- The decoded mesh *M'* obtained by decompressing the compressed bitstream associated with the input mesh *M.*

The codec under evaluation shall produce the encoding reordering information, which specifies for each attribute value the index of the corresponding attribute value in the input mesh. The reordering information shall be stored as a JSON file compliant with the scheme described in Figure 11. The corresponding input and decoded meshes are described in Figure 12 and Figure 13, respectively.

To compare the original mesh original.obj to the decoded mesh decoded.obj, while using the reordering information stored in reorder.json, use the following command line:

```
./mesh-compare \
    --original original.obj \
    --decoded decoded.obj \
    --reorder reorder.json  \
    --log cmp_log.json
```

The parameters of the tool are as follows:

- original: the path of the original mesh in OBJ format [Required]
- decoded: the path of the decoded mesh in OBJ format [Required]
- reorder: the path of the reordering info file in JSON format [Required]
- log: the path of the log file (see Figure 11) [Required]

```
{
  "attributes_reordering": [
    {
      "count": 8,
      "data_type": "integer",
      "indices": [
        6,
        4,
        3,
        2,
        1,
        7,
        0,
        5
```

```
      ],
      "num_components": 3,
      "type": "POSITION"
    },
    {
      "count": 24,
      "data_type": "integer",
      "indices": [
        22,
        23,
        21,
        20,
        11,
        10,
        8,
        9,
        6,
        7,
        5,
        4,
        19,
        18,
        16,
        17,
        14,
        15,
        13,
        12,
        3,
        2,
        0,
        1
      ],
      "num_components": 2,
      "type": "TEX_COORD"
    },
    {
      "count": 24,
      "data_type": "integer",
```

```
        "indices": [
          22,
          23,
          21,
          20,
          11,
          10,
          8,
          9,
          6,
          7,
          5,
          4,
          19,
          18,
          16,
          17,
          14,
          15,
          13,
          12,
          3,
          2,
          0,
          1
        ],
        "num_components": 3,
        "type": "NORMAL"
      }
}
```


Figure 10: Encoding reordering information.

```
{
  "decoded_mesh": {
    "face_count": 60002,
    "is_single_connectivity": false,
    "max_face_vertex_count": 3,
    "min_face_vertex_count": 3,
    "normal_count": 30666,
    "position_count": 29993,
    "shape_count": 1,
    "tex_coord_count": 31188
  },
  "lossless_coding": true,
  "original_mesh": {
    "face_count": 60002,
    "is_single_connectivity": false,
    "max_face_vertex_count": 3,
    "min_face_vertex_count": 3,
    "normal_count": 30666,
    "position_count": 29993,
    "shape_count": 1,
    "tex_coord_count": 31188
  },
  "parameters": {
    "decoded": "apothecary_vase_decoded.obj",
    "log": "apothecary_vase_cmp.json",
    "original": "apothecary_vase.obj",
    "reordering_info": "apothecary_vase_reorder.json"
  }
}
```


Figure 11: Example of comparison log.

```
v 0 0 65535
v 65535 0 0
v 65535 0 65535
v 65535 65535 65535
v 0 65535 65535
v 0 65535 0
v 65535 65535 0
v 0 0 0
vt 16383 16383
vt 0 0
vt 16383 0
vt 0 16383
vt 0 16383
vt 16383 0
vt 16383 16383
vt 0 0
vt 0 0
vt 16383 16383
vt 0 16383
vt 16383 0
vt 16383 16383
vt 0 0
vt 16383 0
vt 0 16383
vt 16383 0
vt 0 16383
vt 0 0
vt 16383 16383
vt 0 0
vt 16383 16383
vt 0 16383
vt 16383 0
vn 512 0 512
vn 512 0 512
vn 512 0 512
vn 512 0 512
vn 512 512 0
vn 512 512 0
vn 512 512 0
```

```
vn 512 512 0
vn 0 512 512
vn 0 512 512
vn 0 512 512
vn 0 512 512
vn 512 512 1023
vn 512 512 1023
vn 512 512 1023
vn 512 512 1023
vn 1023 512 512
vn 1023 512 512
vn 1023 512 512
vn 1023 512 512
vn 512 1023 512
vn 512 1023 512
vn 512 1023 512
vn 512 1023 512
f 8/4/4 2/2/2 3/3/3 1/1/1
f 8/8/8 6/6/6 7/7/7 2/5/5
f 5/12/12 6/10/10 8/11/11 1/9/9
f 3/16/16 4/14/14 5/15/15 1/13/13
f 7/20/20 4/18/18 3/19/19 2/17/17
f 7/23/23 6/21/21 5/24/24 4/22/22
```

Figure 12: Input mesh in OBJ format.

```
v 65535 65535 0
v 0 65535 65535
v 65535 65535 65535
v 65535 0 65535
v 65535 0 0
v 0 0 0
v 0 0 65535
v 0 65535 0
vt 0 16383
vt 16383 0
vt 16383 16383
vt 0 0
vt 16383 0
vt 0 16383
vt 0 0
vt 16383 16383
vt 16383 16383
vt 0 0
vt 16383 0
vt 0 16383
vt 16383 16383
vt 0 0
vt 16383 0
vt 0 16383
vt 16383 0
vt 0 16383
vt 0 0
vt 16383 16383
vt 0 16383
vt 16383 0
vt 16383 16383
vt 0 0
vn 512 1023 512
vn 512 1023 512
vn 512 1023 512
vn 512 1023 512
vn 0 512 512
vn 0 512 512
vn 0 512 512
```

```
vn 0 512 512
vn 512 512 0
vn 512 512 0
vn 512 512 0
vn 512 512 0
vn 1023 512 512
vn 1023 512 512
vn 1023 512 512
vn 1023 512 512
vn 512 512 1023
vn 512 512 1023
vn 512 512 1023
vn 512 512 1023
vn 512 0 512
vn 512 0 512
vn 512 0 512
vn 512 0 512
f 8/4/4 2/2/2 3/3/3 1/1/1
f 8/8/8 6/6/6 7/7/7 2/5/5
f 5/12/12 6/10/10 8/11/11 1/9/9
f 3/16/16 4/14/14 5/15/15 1/13/13
f 7/20/20 4/18/18 3/19/19 2/17/17
f 7/23/23 6/21/21 5/24/24 4/22/22
```

Figure 13: Decoded mesh in OBJ format.

Annex H: Lossy anchor generation process

The full framework is implemented as a Python script that runs sequentially the following modules:

- Decimation: the decimation is performed by MeshLab, using the python pyMeshLab library with a decimation ratio (decRatio). To make the C0-L5 mesh compatible with MeshLab decimation, C0-L5 is first dequantized and triangulated. The decimation algorithm applied is the "Quadric Edge Collapse Decimation (with texture)", by Garland and Heckbert [12].
- Draco coding and decoding: Draco is used to encode and decode the decimated mesh. The quantization bits for the position attribute (qpDraco), and for texture coordinate attribute (qtDraco) are set in a way to control the size of the bitstream file. The quantization bits for the normal vector attribute is set to -1 to avoid compression of normal vectors.
- Texture downsampling: C0-L5 texture is downsampled by a ratio ¾ in each horizontal and vertical direction, using HDRConvert.
- AV1 texture coding and decoding: the downsampled texture is encoded by AV1, using libaom, at different compression levels (cq-level).

Annex I: List of scripts, tools and parameters

Tools and parameters for the lossy anchor

The lossy anchor can be generated by the Python script vvmLossyAnchorGeneration.py and related files, that can be found at (tag spmc_cfp_1.0):

https://gitlab.com/AOMediaVVM/cfp/spmc

This script runs sequentially the different modules, depicted in Figure 4. Each module can be launched independently. The software to be used and the command line or configuration is provided below.

- PyMeshLab: library used for decimating the meshes

2022.2.post2	https://pymeshlab.readthedocs.io/en/latest/installation.html

- mesh_polygon2triangle: Python code used for triangulation of the meshes

tag: spmc_cfp_1.0	https://gitlab.com/AOMediaVVM/eval-tools/MeshProcessingTools
mesh_polygon2triangle.py \ -i {inputFile} \ -o {outputFile}	

- Draco: software used for encoding and decoding the meshes.

tag: spmc_cfp_lossy_1.0	https://gitlab.com/AOMediaVVM/eval-tools/draco
draco_encoder \ -i {inputObj} \ -o {outputFileBit} \ -qp {qp} -qt {qt} -qn -1 -qg 30 -cl 10	
draco_decoder \ -i {outputFileBit} \ -o {outputFile}	

- AOM encoder and decoder: software used to encode and decode the textures

tag: 3gpp-2021-10-15-5	https://aomedia.googlesource.com/aom/
aomenc \ --verbose --codec=av1 -v --psnr --obu \ --frame-parallel=0 --cpu-used=0 --limit=1 --passes=1 \ --end-usage=q --i420 --enable-tpl-model=0 --disable-kf \ --enable-keyframe-filtering=0 --fps=1/1 \ --input-bit-depth=10 --bit-depth=10 \ -w {widthOutput} -h {heightOutput} \ --cq-level = {cq-level} \ --tile-columns=0 --threads=1 --max-reference-frames=4 \ --min-gf-interval=32 --max-gf-interval=32 \ --gf-min-pyr-height=5 --gf-max-pyr-height=5 \ --kf-min-dist=1 --kf-max-dist=1 --lag-in-frames=0 \ --min-q= {cq-level} --max-q={cq-level}\	

```
                    --disable-warning-prompt --deltaq-mode=0 \
                    --enable-intrabc=0 --enable-palette=0 \
                    --target-seq-level-idx=0031 \
                    -o {outputTextureBin} \
                    {inputTexture}

aomdec \

                    --i420 –verbose \
                    --output= {outputYuv}\
                    " " {outputTextureBin}
```

- HDRConvert: software used for image down-sampling, image up-sampling, and colour space conversion

tag: v0.24	https://gitlab.com/standards/HDRTools

For RGB to YUV BT. 709 with potential image scaling:

```
HDRConvert \

                    -f {path/to/pngtoyuv420.cfg} \
                    -p SourceFile={path/to/png/file} \
                    -p OutputWidth={outputWidth} \
                    -p OutputHeight={outputHeight} \
                    -p OutputFile={path/to/yuv/file}
```

For YUV BT. 709 to RGB with potential image scaling:

```
HDRConvert \

                    -f {path/to/yuv420torgb444.cfg} \
                    -p SourceFile={path/to/yuv/file} \
                    -p SourceWidth={sourceWidth} \
                    -p SourceHeight={sourceHeight} \
                    -p OutputWidth={outputWidth} \
                    -p OutputHeight={outputHeight} \
                    -p OutputFile={path/to/png/file}
```

- mesh-quality-metric: C++ tool to compute objective quality metrics described in Annex K.

tag: spmc_cfp_1.0	https://gitlab.com/AOMediaVVM/eval-tools/MeshProcessingTools

```
mesh-quality-metric \

                    --mesh1 {referenceObj} --tex1 {referencePng} \
                    --mesh2 {decodedObj} --tex2 {decodedPng} \
                    --log {logFileJson}
```

Tools and parameters for the lossless anchor

- mesh-zlib-codec: C++ code to compress and decompress quantized meshes as the lossless anchor for C0.

tag: spmc_cfp_1.0	https://gitlab.com/AOMediaVVM/eval-tools/MeshProcessingTools

```
mesh-zlib-codec \

                    --mode 0 \
                    --input {inputObj} \
                    --output {outputFileBin} \
```

--reorder {outputReorderJson} \ --log {encoderLog}
mesh-zlib-codec \ --mode 1 \ --input {inputFileBin} \ --output {outputObj} \ --info {encoderLog} \ --log {decoderLog}

- Modified Draco for lossless coding: C++ code to compress and decompress quantized meshes as the lossless anchor for C1.

tag: spmc_cfp_lossless_1.0	https://gitlab.com/AOMediaVVM/eval-tools/draco
draco_encoder \ -i {inputObj} \ -o {outputFileBin} \ -r {outputReorderJson} \ -l {encoderLog} \ -cl 10 \ -preserve_polygons 1	
draco_decoder \ -i {inputFileBin} \ -o {outputObj} \ -l {decoderLog}	

- Modified Draco for lossless coding: C++ code to compress and decompress quantized meshes as the lossless anchor for C2.

tag: spmc_cfp_lossless_1.0	https://gitlab.com/AOMediaVVM/eval-tools/draco
draco_encoder \ -i {inputObj} \ -o {outputFileBin} \ -r {outputReorderJson} \ -l {encoderLog} \ -cl 10 \ -unify_indices	
draco_decoder \ -i {inputFileBin} \ -o {outputObj} \ -l {decoderLog}	

- Lossless compression evaluation tool: C++ code to compare if a mesh is lossless coded.

tag: spmc_cfp_1.0	https://gitlab.com/AOMediaVVM/eval-tools/MeshProcessingTools
mesh-compare \ --original {originalObj} \ --decoded {decodedObj} \ --reorder {reorderJson} \	

--log {compareLog}

- Lossless anchor generation:

tag: spmc_cfp_1.0	https://gitlab.com/AOMediaVVM/cfp/spmc
python3 generate_lossless_anchors.py \ --input_path {inputPath} \ --output_path {outputPath} \ --binary_path {binariesPath}	

Other tools

- Generation of input categories:

tag spmc_cfp_1.0	https://gitlab.com/AOMediaVVM/cfp/spmc
python3 generate_input_categories.py \ --input_path {inputPath} \ --output_path {outputPath} \ --binary_path {binariesPath} \ --config_path {configPath}	

- Quantization:

tag spmc_cfp_1.0	https://gitlab.com/AOMediaVVM/cfp/spmc
To quantize the input mesh input.obj with 11-bit precision for positions, 10-bit precision for texture coordinates, and 8-bit precision for normal: mesh-quantize \ --input {inputObj} \ --output {quantizedObj} \ --qp 11 --qt 10 --qn 8 \ --qinfo {quantizationInfoJson}	
To apply inverse quantization to a quantized mesh quantized_output.obj: mesh-quantize \ --mode 1 \ --input {quantizedObj} \ --output {reconstructedObj} \ --qinfo {quantizationInfoJson}	

- Downscaling:

tag spmc_cfp_1.0	https://gitlab.com/AOMediaVVM/cfp/spmc
python3 downscale_images.py \ --input_path {inputPath} \ --output_path {outputPath} \ --binary_path {binariesPath} \ --config_path {configPath}	

Annex J: Rendering scripts and tools

The rendering can be generated by the Python script vvmRenderer.py, available at (tag spmc_cfp_1.0):

https://gitlab.com/AOMediaVVM/cfp/spmc

The renderer is using open3D library to produce still images (PNG files) that are further converted to videos for subjective viewing (MP4 files). The software to be used and the command line or configuration are provided below.

This script can be launched independently, or from the lossy anchor generation script vvmLossyAnchorGeneration.py.

- vvmRenderer.py.

```
python vvmRenderer.py \
        --inputObjFile {inputObj} \
        --outputFolderPng {ouputPathPng}\
        --outputMp4 {outputMp4} \
        --initial_rotation_h {initial_rotation_h} \
        --initial_rotation_v {initial_rotation_v} \
        --initial_translation_h {initial_translation_h} \
        --initial_translation_v {initial_translation_v} \
        --scale {scale} \
        --rotation_speed {rotation_speed} \
        --grid_translate {grid_translate} \
        --inputFloor {inputFloorObj} \
        --nbFrames {nbFrames }
```

- Open3D: library used for rendering the meshes.

0.16.0	https://github.com/isl-org/Open3D/tags

- ffmpeg: library used to convert the list of PNG files produced by the renderer to MP4 videos. The crf value of 10 has been selected according to the recommendations made in [13].

v5.1.2	Executable files for Linux, Mac and Windows: https://ffmpeg.org/download.html

```
ffmpeg \
    -y -r 30 -f concat -safe 0 \
    -i {filesList} \
    -c:v libx265 -crf 10 -tag:v hvc1 -pix_fmt yuv420p {outputFile }
```

Annex K: Performance indicators

Sampling

The sampling process takes as input:

- A mesh *M*.
- A 3D bounding box *B* containing *M* and used to guide the sampling process.
- A user-defined parameter *G* controlling the precision of the sampling process.
- Optionally a texture map *T*.

The output of the sampling process is a point cloud *P* specified by a set of 3D points, with 3D positions, surface normal vectors, and colour information.

The sampling process proceeds as follows.

First, the bounding box *B* is uniformly sampled to generate a uniform grid of 3D points of size $G \times G \times G$. Then, a subset of these points is projected on the faces of the mesh *M* to generate the output point cloud *P* as described in the remainder of this section.

After triangulating the faces of the mesh M, the generated triangles are processed successively as follows. First, the normal to the current triangle is computed and used to choose between one of the three main projection planes, XY, YZ, or XZ. The plane with the normal direction closest to the triangle normal vector is selected. Then, the current triangle is projected on the selected plane according to the plane normal direction and all grid points located inside the projected triangle are added to the point cloud *P*. The positions of the added points are determined by projecting them back on the triangle according to the normal direction of the plane. The added points inherit the normal of the current triangle. The colours of the added points are determined by bilinear sampling of the texture map T. The texture coordinates used for the bilinear sampling are computed by barycentric interpolation of the texture coordinates of the triangle vertices.

Geometry quality metrics

The description of the performance indicators d1PSNR and d2PSNR is based on the reference document [10]. Let A and B denote the original and the compressed point cloud, respectively, obtained after the sampling stage described above. The compression errors in point cloud B relative to the original point cloud A are evaluated.

The d1 point-to-point error evaluates the distance between two points. The evaluation is achieved in a two-pass computation. In each pass, one point cloud is selected as reference, *e.g.* when computing d1(A,B), A serves as a reference, and vice versa. In the end, the lowest value among d1(A,B) and d1(B,A) is selected as the final measurement. The following algorithm is applied:

1. For each point a_j in point cloud A, identify a corresponding point b_j in point cloud B. The nearest neighbour is used to locate the corresponding point.
2. Take the unit normal vector N_j on point a_j in the reference point cloud A.
3. Compute the error vector $E(i,j)$ by connecting point a_j to point b_i. The length of the error vector leads to the d1 point-to-point error, *i.e.*

$$d1_{A,B} = \frac{1}{N_A} \sum_{\forall a_j \in A} \|E(i,j)\|_2^2 ,$$

where N_A is the number of points in point cloud A.

Both $d1_{A,B}$ and $d1_{B,A}$ are computed. The corresponding MSE is converted into PSNR, to produce the final d1PSNR metric, using the max value of $d1_{A,B}$ and $d1_{B,A}$:

$$d1PSNR = 10 \log_{10} \left(\frac{p^2}{max\left(d1_{A,B}, d1_{B,A}\right)} \right).$$

The diagonal distance of a bounding box of the point cloud A is used to define the peak value p.

The d2PSNR point-to-plane metric computes the sum of the squared distance between a point and the tangent plane at its correspondence point. From the third step of the d1PSNR computation, the following algorithm is applied: From the third step of the d1PSNR computation, the following algorithm is applied:

1. Get point-to-plane errors by projecting the error vector $E(i,j)$ along the normal direction N_j to get a new error vector $E(i,j) \cdot N_j$. The point-to-plane error is finally computed as:

$$d2_{A,B} = \frac{1}{N_A} \sum_{\forall a_j \in A} \left\| E(i,j) \cdot N_j \right\|_2^2 \quad .$$

Both $d2_{A,B}$ and $d2_{B,A}$ are computed. The corresponding MSE is converted into PSNR, to produce the final d2PSNR metric, using the max value of $d2_{A,B}$ and $d2_{B,A}$:

$$d2PSNR = 10 \log_{10} \left(\frac{p^2}{max\left(d2_{A,B}, d2_{B,A}\right)} \right).$$

Texture quality metrics

After conversion of the point colours from the RGB space to the YUV BT.709 space, the MSE of each of the three colour components luma, cb, cr is computed, and noted lumaMSE, cbMSE and crMSE. The lumaPSNR is obtained with:

$$lumaPSNR = 10 \log_{10} \left(\frac{255^2}{lumaMSE} \right),$$

and the cbPSNR and crPSNR are obtained by replacing lumaMSE by cbMSE and crMSE, respectively.

Annex L: Quality targets and lossy anchor results

According to the rules defined in section 8, responses to the CfP must match pre-defined target qualities. This annex explains how the target qualities were selected, and what are the coding configurations that yield to these target qualities, for the anchor.

The encoding of the meshes is performed with different coding configurations, made of combinations of decRatio, qpDraco, qtDraco parameters. Multiple configurations were tested to find the best match with the requirements listed below. The tested parameters were the following:

- Decimation parameters: decRatio = [0.10, 0.20, 0.30, 0.40, 0.50, 0.60, 0.80, 1.00].
- Quantization parameters for Draco: [qpDraco, qtDraco] = [[15,14], [14, 13], [13, 12], [12,11], [11, 10], [10, 9], [9, 8], [8, 7], [7,6], [6,5], [5,4]].
- Compression levels for AV1: cq-level = [8, 13, 18, 23, 28, 33, 38, 43, 48, 53, 58, 63].

The selection depends on the type of meshes:

- For objects without texture, the selection is made using the point-to-plane d2PSNR metric. The four selected configurations meet the following constraints (in this order of priority): 1- they are located on the convex hull obtained from the different test points, 2- their d2PSNR quality is close to target qualities 65, 72, 79, and 86.
- For objects with texture, that are part of the viewing set, the four selected configurations meet the following constraints (in this order of priority): 1- they are located on the convex hull obtained from the different test points, 2- their quality is in a range of acceptable visual quality, for practical mesh-based services, 3- their quality is reasonably spread in the range of acceptable visual quality.
- For objects with texture, that are not part of the viewing test, the selection relies on the following principles: 1- the configuration for the lowest quality is set to decRatio=0.10, qpDraco=9, qtDraco=8, cq-level=63. This configuration corresponds to the average configuration selected for TQ4 during the subjective viewing dry-run, applied on the viewing set. 2- the configuration for the highest quality (TQ1) is set to decRatio=1.00, qpDraco=12, qtDraco=11, cq-level=38. This configuration corresponds to the average configuration selected for TQ1 during the subjective viewing dry-run, applied on the viewing set.

Table 5 reports for each object without texture, the selected configuration for each target quality, as well as the d2PSNR quality of the anchor that becomes the target quality for the CfP responses. The achieved file sizes of the anchor (size of geometry and connectivity information) are available in the VVM template for lossy scenario excel sheet, in the “anchor” tab.

Table 6 reports for each object with texture, the selected configurations, the target lumaPSNR quality, the achieved lumaPSNR quality of the anchor that becomes the target for the CfP responses, the achieved file size of the anchor (size of geometry, connectivity, mapping information, and the size of the texture attribute map).

Mesh Name	TQ	d2PSNR	decRatio	qpDraco	qtDraco
chair_swan	TQ4	83.4	0.20	12	11
chair_swan	TQ3	76.3	0.10	11	10
chair_swan	TQ2	69.3	0.10	8	7
chair_swan	TQ1	63.7	0.10	7	6
cup_saucer_set	TQ4	83.6	0.30	11	10
cup_saucer_set	TQ3	78.9	0.20	10	9
cup_saucer_set	TQ2	70.3	0.10	8	7
cup_saucer_set	TQ1	63.6	0.10	7	6
flower_tulip	TQ4	85.0	0.20	12	11
flower_tulip	TQ3	76.6	0.10	10	9
flower_tulip	TQ2	73.0	0.10	9	8
flower_tulip	TQ1	62.3	0.10	7	6
gramophone	TQ4	83.5	0.10	12	11
gramophone	TQ3	80.1	0.10	10	9
gramophone	TQ2	73.0	0.10	9	8
gramophone	TQ1	62.9	0.10	7	6
teapot	TQ4	85.7	0.20	11	10
teapot	TQ3	78.4	0.10	10	9
teapot	TQ2	70.2	0.10	8	7
teapot	TQ1	64.9	0.10	7	6
toy_bitplane	TQ4	84.1	0.40	11	10
toy_bitplane	TQ3	76.3	0.20	11	10
toy_bitplane	TQ2	69.4	0.10	10	9
toy_bitplane	TQ1	62.6	0.10	7	6
toy_car	TQ4	83.6	0.40	12	11

toy_car	TQ3	77.0	0.30	10	9
toy_car	TQ2	69.1	0.10	11	10
toy_car	TQ1	65.7	0.10	8	7
toy_drummer	TQ4	83.5	0.40	12	11
toy_drummer	TQ3	76.6	0.30	10	9
toy_drummer	TQ2	71.9	0.20	9	8
toy_drummer	TQ1	65.0	0.10	8	7
toy_robot_vintage	TQ4	83.3	0.30	11	10
toy_robot_vintage	TQ3	77.8	0.20	10	9
toy_robot_vintage	TQ2	70.7	0.10	9	8
toy_robot_vintage	TQ1	66.7	0.10	8	7
tv_retro	TQ4	85.6	0.10	11	10
tv_retro	TQ3	81.4	0.10	10	9
tv_retro	TQ2	69.4	0.10	8	7
tv_retro	TQ1	62.1	0.10	7	6
wateringcan	TQ4	83.3	0.20	11	10
wateringcan	TQ3	80.4	0.20	10	9
wateringcan	TQ2	72.3	0.10	9	8
wateringcan	TQ1	64.6	0.10	7	6

Table 5: Selected configurations, including the initial target quality, the achieved quality of the anchor, the achieved file size of the anchor, for each object without texture.

Mesh Name	TQ	lumaPSNR	decRatio	qpDraco	qtDraco	cq-level
apollo_11	TQ4	32.1	1.00	13	12	33
apollo_11	TQ3	30.5	1.00	13	12	43
apollo_11	TQ2	26.6	1.00	11	10	53
apollo_11	TQ1	20.1	0.10	9	8	63
apollo_11	TQ0	15.6	0.10	7	6	63
apothecary_vase	TQ4	31.1	1.00	12	11	53
apothecary_vase	TQ3	26.3	0.10	12	11	58
apothecary_vase	TQ2	24.1	0.10	10	9	63
apothecary_vase	TQ1	19.2	0.10	8	7	63
apothecary_vase	TQ0	16.9	0.10	7	6	63
armillary_sphere_1771	TQ4	33.6	1.00	12	11	38
armillary_sphere_1771	TQ3	30.9	0.20	13	12	48
armillary_sphere_1771	TQ2	28.9	0.10	11	10	53
armillary_sphere_1771	TQ1	25.8	0.10	9	8	63
buste_cuirasse_de_marc_aurele_age	TQ4	35.1	1.00	11	10	28
buste_cuirasse_de_marc_aurele_age	TQ3	31.2	0.30	10	9	43
buste_cuirasse_de_marc_aurele_age	TQ2	29.4	0.10	10	9	53
buste_cuirasse_de_marc_aurele_age	TQ1	27.4	0.10	9	8	58
buste_cuirasse_de_marc_aurele_age	TQ0	22.1	0.10	7	6	63
butterflies_collection	TQ4	28.9	1.00	12	11	38
butterflies_collection	TQ3	27.3	0.10	12	11	58
butterflies_collection	TQ2	22.0	0.10	10	9	63
butterflies_collection	TQ1	18.7	0.10	9	8	63
candle_stick	TQ4	34.9	1.00	12	11	38

candle_stick	TQ3	33.1	0.30	13	12	48
candle_stick	TQ2	30.2	0.10	11	10	58
candle_stick	TQ1	27.5	0.10	9	8	63
cela_ruins_of_a_nuns_cell	TQ4	33.9	1.00	12	11	38
cela_ruins_of_a_nuns_cell	TQ3	32.1	0.20	13	12	48
cela_ruins_of_a_nuns_cell	TQ2	29.4	0.10	11	10	58
cela_ruins_of_a_nuns_cell	TQ1	26.2	0.10	9	8	63
creature_box_squid	TQ4	30.6	1.00	12	11	38
creature_box_squid	TQ3	29.1	1.00	12	11	48
creature_box_squid	TQ2	25.7	1.00	9	8	63
creature_box_squid	TQ1	23.3	0.10	9	8	63
cyber_samurai	TQ4	28.5	1.00	12	11	38
cyber_samurai	TQ3	25.6	0.40	12	11	48
cyber_samurai	TQ2	21.4	0.20	11	10	63
cyber_samurai	TQ1	17.8	0.10	9	8	63
dead_rose_smallCCremoved	TQ4	33.6	1.00	12	11	43
dead_rose_smallCCremoved	TQ3	30.3	0.60	12	11	53
dead_rose_smallCCremoved	TQ2	26.9	0.30	10	9	58
dead_rose_smallCCremoved	TQ1	23.1	0.20	8	7	63
dead_rose_smallCCremoved	TQ0	17.5	0.10	6	5	63
electrodynamic	TQ4	34.9	1.00	12	11	38
electrodynamic	TQ3	33.2	0.20	13	12	48
electrodynamic	TQ2	30.8	0.10	10	9	53
electrodynamic	TQ1	28.3	0.10	9	8	63
frederic_fr00001	TQ4	36.1	1.00	11	10	48
frederic_fr00001	TQ3	34.7	0.40	12	11	53

frederic_fr00001	TQ2	31.1	0.30	11	10	63
frederic_fr00001	TQ1	28.3	0.10	9	8	63
frederic_fr00001	TQ0	23.5	0.10	7	6	63
green_tree_frog	TQ4	26.6	1.00	12	11	38
green_tree_frog	TQ3	22.5	0.20	12	11	63
green_tree_frog	TQ2	19.6	0.20	10	9	63
green_tree_frog	TQ1	17.1	0.10	9	8	63
grey_knight	TQ4	30.8	1.00	12	11	43
grey_knight	TQ3	28.3	0.60	12	11	58
grey_knight	TQ2	26.1	0.60	11	10	63
grey_knight	TQ1	22.7	0.10	9	8	63
grey_knight	TQ0	19.7	0.10	6	5	63
heilig_grab_kapelle	TQ4	31.4	1.00	12	11	38
heilig_grab_kapelle	TQ3	29.4	0.20	12	11	48
heilig_grab_kapelle	TQ2	25.8	0.20	10	9	63
heilig_grab_kapelle	TQ1	24.6	0.10	9	8	63
heliostat	TQ4	36.0	1.00	12	11	38
heliostat	TQ3	34.4	0.20	12	11	48
heliostat	TQ2	29.8	0.10	10	9	58
heliostat	TQ1	26.9	0.10	9	8	63
hussar_on_horseback	TQ4	32.0	1.00	12	11	43
hussar_on_horseback	TQ3	28.7	0.60	11	10	58
hussar_on_horseback	TQ2	26.9	0.10	11	10	63
hussar_on_horseback	TQ1	24.6	0.10	8	7	63
hussar_on_horseback	TQ0	21.3	0.10	7	6	63
japanese_spiny_lobster	TQ4	26.3	1.00	12	11	38

japanese_spiny_lobster	TQ3	23.8	1.00	12	11	58
japanese_spiny_lobster	TQ2	20.8	1.00	11	10	63
japanese_spiny_lobster	TQ1	16.1	0.10	9	8	63
just_a_girl	TQ4	27.5	1.00	12	11	38
just_a_girl	TQ3	24.2	0.80	12	11	63
just_a_girl	TQ2	16.8	0.40	12	11	58
just_a_girl	TQ1	14.2	0.10	9	8	63
levi_fr00000	TQ4	41.2	1.00	12	11	38
levi_fr00000	TQ3	37.2	0.80	11	10	53
levi_fr00000	TQ2	32.8	0.60	11	10	63
levi_fr00000	TQ1	25.7	0.10	9	8	63
luna_lionfish	TQ4	28.7	1.00	13	12	48
luna_lionfish	TQ3	25.4	1.00	12	11	58
luna_lionfish	TQ2	23.7	1.00	11	10	53
luna_lionfish	TQ1	21.7	1.00	11	10	63
luna_lionfish	TQ0	18.6	1.00	10	9	63
marble_mortar	TQ4	34.7	1.00	12	11	38
marble_mortar	TQ3	33.9	0.40	11	10	38
marble_mortar	TQ2	31.8	0.10	10	9	53
marble_mortar	TQ1	30.3	0.10	9	8	63
mira_w_gun	TQ4	29.3	1.00	12	11	58
mira_w_gun	TQ3	26.9	1.00	11	10	63
mira_w_gun	TQ2	24.3	0.30	10	9	63
mira_w_gun	TQ1	23.3	0.20	10	9	63
mira_w_gun	TQ0	18.4	0.10	7	6	63
mitch_fr00001	TQ4	32.4	1.00	12	11	38

mitch_fr00001	TQ3	30.8	0.60	13	12	48
mitch_fr00001	TQ2	28.7	0.20	11	10	58
mitch_fr00001	TQ1	26.6	0.10	9	8	63
motorcycle	TQ4	37.1	1.00	12	11	38
motorcycle	TQ3	33.3	0.60	11	10	58
motorcycle	TQ2	29.0	0.40	9	8	63
motorcycle	TQ1	22.8	0.10	9	8	63
nathalie_fr00036	TQ4	28.8	1.00	11	10	58
nathalie_fr00036	TQ3	27.9	0.20	11	10	58
nathalie_fr00036	TQ2	26.8	0.10	10	9	63
nathalie_fr00036	TQ1	25.5	0.10	8	7	63
nathalie_fr00036	TQ0	23.6	0.10	7	6	63
orbiter_space_shutter	TQ4	27.5	1.00	13	12	53
orbiter_space_shutter	TQ3	23.7	0.80	11	10	63
orbiter_space_shutter	TQ2	22.4	0.30	11	10	63
orbiter_space_shutter	TQ1	21.0	0.20	10	9	63
orbiter_space_shutter	TQ0	19.4	0.20	9	8	63
piggy_bank	TQ4	34.5	1.00	12	11	38
piggy_bank	TQ3	33.2	0.40	12	11	38
piggy_bank	TQ2	30.9	0.20	13	12	53
piggy_bank	TQ1	28.2	0.10	9	8	63
police_station	TQ4	27.0	0.80	12	11	38
police_station	TQ3	23.7	0.20	11	10	53
police_station	TQ2	22.8	0.10	11	10	58
police_station	TQ1	19.6	0.10	9	8	63
police_station	TQ0	16.9	0.10	8	7	63

promo_ashtray	TQ4	41.1	1.00	12	11	38
promo_ashtray	TQ3	38.0	0.20	12	11	48
promo_ashtray	TQ2	34.6	0.10	11	10	58
promo_ashtray	TQ1	30.0	0.10	9	8	63
rafa_fr00001	TQ4	38.0	1.00	12	11	38
rafa_fr00001	TQ3	36.2	0.80	12	11	48
rafa_fr00001	TQ2	31.8	0.30	11	10	58
rafa_fr00001	TQ1	27.5	0.10	9	8	63
sakura_cherry_blossom	TQ4	28.6	1.00	12	11	38
sakura_cherry_blossom	TQ3	26.1	1.00	12	11	58
sakura_cherry_blossom	TQ2	20.7	0.20	11	10	58
sakura_cherry_blossom	TQ1	14.6	0.10	9	8	63
stereoscopic_cam	TQ4	32.0	1.00	12	11	43
stereoscopic_cam	TQ3	30.2	0.20	12	11	53
stereoscopic_cam	TQ2	29.1	0.10	11	10	58
stereoscopic_cam	TQ1	27.7	0.10	9	8	63
stereoscopic_cam	TQ0	25.0	0.10	7	6	63
the_great_drawing_room	TQ4	26.5	1.00	12	11	38
the_great_drawing_room	TQ3	24.5	0.30	12	11	43
the_great_drawing_room	TQ2	22.4	0.10	12	11	53
the_great_drawing_room	TQ1	19.6	0.10	9	8	63
the_serving_room	TQ4	22.1	1.00	12	11	38
the_serving_room	TQ3	19.5	0.20	13	12	48
the_serving_room	TQ2	16.6	0.10	11	10	58
the_serving_room	TQ1	12.4	0.10	9	8	63
thomas_fr00170	TQ4	34.4	1.00	12	11	38

thomas_fr00170	TQ3	32.7	0.40	12	11	48
thomas_fr00170	TQ2	30.5	0.20	10	9	58
thomas_fr00170	TQ1	28.5	0.10	9	8	63
violin	TQ4	38.3	1.00	12	11	38
violin	TQ3	36.8	0.30	13	12	43
violin	TQ2	33.8	0.10	11	10	58
violin	TQ1	31.2	0.10	9	8	63
ware_bowl	TQ4	36.6	1.00	12	11	38
ware_bowl	TQ3	33.4	0.20	12	11	48
ware_bowl	TQ2	30.3	0.10	12	11	58
ware_bowl	TQ1	25.3	0.10	9	8	63
winter_girl	TQ4	26.8	1.00	12	11	38
winter_girl	TQ3	23.6	0.80	11	10	58
winter_girl	TQ2	20.4	0.60	10	9	63
winter_girl	TQ1	14.5	0.10	9	8	63
wiz_boots	TQ4	35.7	1.00	12	11	38
wiz_boots	TQ3	34.5	0.30	12	11	43
wiz_boots	TQ2	30.8	0.10	10	9	53
wiz_boots	TQ1	28.4	0.10	9	8	63
wooden_gramophone	TQ4	34.9	1.00	12	11	38
wooden_gramophone	TQ3	34.0	0.30	12	11	43
wooden_gramophone	TQ2	31.0	0.10	10	9	53
wooden_gramophone	TQ1	28.8	0.10	9	8	63
zakopane_chair	TQ4	29.6	0.20	13	12	53
zakopane_chair	TQ3	26.9	0.10	10	9	63
zakopane_chair	TQ2	25.6	0.10	8	7	63

zakopane_chair	TQ1	24.3	0.10	7	6	63
zakopane_chair	TQ0	22.2	0.10	6	5	63

Table 6: Selected configurations, including the initial target quality, the achieved quality of the anchor, the achieved file size of the anchor, for each object with texture.

Annex M: Encoding and decoding logs

Figure 14 and Figure 15 show the format and the information reported in the encoding and decoding logs.

The total byte count (*i.e.* "total_byte_count") corresponds to the total compressed bitstream size in bytes. The total encoding (*i.e.* "total_encoding_time") and the total decoding time (*i.e.* "total_decoding_time") correspond to the time needed to encode and decode (excluding any IO operations) all the mesh components in milliseconds, respectively. The triangle count (*i.e.* "triangle_count") corresponds to the number of triangles in the decoded mesh. If the decoded mesh contains non-triangular faces, the number of triangles obtained after the non-triangular faces are triangulated is reported.

The array attributes shall contain the statistics associated with each mesh component (e.g. positions, texture coordinates, normal, and texture map):

- byte count (*i.e.* "byte_count") corresponds to the compressed bitstream size in bytes needed to encode the considered component.
- encoding (*i.e.* "encoding_time") and the decoding time (*i.e.* "decoding_time") correspond to the time needed to encode and decode (excluding any IO operations) the considered mesh component in milliseconds, respectively.

For corner attributes, the byte count and encoding and decoding time consider both the attribute values and indices.

```
{
  "encoding_stats": {
    "attributes": [
      {
        "byte_count": 268388,
        "encoding_time": 150.262,
        "type": "POSITION"
      },
      {
        "byte_count": 113496,
        "encoding_time": 18.419,
        "type": "TEX_COORD"
      },
      {
        "byte_count": 324575,
        "encoding_time": 14.609,
        "type": "NORMAL"
      },
      {
        "byte_count": 0,
        "encoding_time": 0,
        "type": "TEX_MAP"
      }
    ],
    "total_byte_count": 706459,
    "total_encoding_time": 183.365,
  }
```


Figure 14: Encoding log format and required information.

```
{
  "decoding_stats": {
    "attributes": [
      {
        "decoding_time": 0.6125,
        "type": "NORMAL"
      },
      {
        "decoding_time": 1.1925,
        "type": "TEX_COORD"
      },
      {
        "decoding_time": 2.860,
        "type": "POSITION"
      },
      {
        "decoding_time": 0,
        "type": "TEX_MAP"
      }
    ],
    "total_decoding_time": 4.665
    "triangle_count": 15001
  }
}
```


Figure 15: Decoding log format and required information.