3GPP TSG-SA WG4 Meeting #132S4-250942

Japan, Fukuoka, 19 – 23 May 2025

**Source: InterDigital**

**Title: Pseudo-CR on annex for software packages**

**Spec: 3GPP TR 26.956 V0.4.0**

**Agenda item: 9.7**

**Document for: Agreement**

**1. Introduction**

This contribution describes software packages for scenario 2 that are used to generate objective and subjective test results. Interested users can use these packages to understand the processing or to reproduce results.

**2. Reason for Change**

Without these software packages the processing steps would not be visible and interested users would not be able to reproduce the results provided.

**3. Proposal**

It is proposed to agree the following changes to 3GPP TR 26.956 V0.4.0.

\* \* \* First Change (all new) \* \* \* \*

# Annex D Software Package

## D.X Scenario 2 Processing

### D.X.1 Overview

The generation of objective metrics and 2D videos for subjective viewing for scenario 2 is supported by a software package provided in this repository: <https://github.com/XXX/Beyond2D>

The software package permits the following functionalities:

- Test sequence preparation

- Bitstream generation and objective metric generation

- 2D video generation using a camera path for subjective viewing

### D.X.2 Installation

#### D.X.2.1 Cloning

git clone <https://github.com/XXX/Beyond2D>

cd Beyond2D

Please use a [python virtual environment](https://docs.python.org/3/library/venv.html#creating-virtual-environments) to install dependencies and run the scripts. A requirements.txt file is provided such that a suitable virtual environment can be set-up as follows:

python3 -m venv venv

venv\Scripts\activate # on Windows

. venv/bin/activate # on Linux

python -m pip install –upgrade pip

pip install -r requirements.txt

#### D.X.2.2 Working Directory

The scripts assume that the current directory is a local working directory, at the root of the repository.

### D.X.3 Test sequence preparation

#### D.X.3.1 Dense dynamic point cloud

This clause describes how reference sequences provided in dynamic mesh format are converted to the dense dynamic point cloud format with the target quality (vox11, approximately 2M points/frame). Please follow instructions in annex C.2 for downloading the sequences Mitch, Joggle Soccer, Nathalie, Aliyah and Henry in dense dynamic mesh format. The sequences Aliyah and Henry are provided as Blender project and generation of dense dynamic mesh is described in the complementary document doc/point\_cloud/readme\_ply\_generation.md in the repository.

####  D.X.3.1.1 Generation of target dense dynamic point clouds

To proceed with the generation, the user needs to navigate to the point\_cloud/ply\_generation/ directory, which contains:

* \*.py: Python scripts for generating PLY (point cloud) files.
* output\_info/: Directory containing all expected md5sum result files for meshes (\*\_mesh\_md5.txt) and PLY files (\*\_output.log) for each sequence.
* jsons/: Directory with an example of input configuration files.

A JSON file named 3gpp\_selection.json is provided as input and is located in the jsons/ directory. It contains all information listed in Table Y. This JSON file needs to be updated for each sequence with the correct paths to the meshes for your environment (MeshObjPath and MeshTxtPath).

Table Y conversion parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sequence** | **Geo Quantization Bitdepth** | **Ratio** | **1st Frame Index** | **Frame Number** |
| **Mitch** | 11 | 0.70 | 1 | 475 |
| **JuggleSoccer** | 11 | 1 | 0 | 125 |
| **Henry** | 11 | 0.75 | 1 | 733 |
| **Nathalie** | 11 | 1 | 1 | 925 |
| **Aliyah** | 11 | 0.88 | 1 | 1112 |

Once the JSON file is updated with the correct mesh paths, the PLY generation can be launched using the script exec\_ply\_generation.py which goes through the following steps:

* The MPEG mmetric software [DM-18] is automatically downloaded to the output directory within the dependencies directory.
* A sampling pass gathers information on the sequence for quantifying the number of expected points. A ratio is provided via the JSON file to ensure each sequence generates point clouds with approximately 2M points/frame.
* Quantization pass.
* Cleaning pass: This step removes all duplicate points using PyntCloud in Python.

The script is launched from the python environment with the following command:

python3 point\_cloud/ply\_generation/exec\_ply\_generation.py -i point\_cloud/ply\_generation/jsons/3gpp\_selection.json -o $YOUR\_OUTPUT\_PATH

For help on the script see complementary document readme\_ply\_generation in the doc folder installed by Git.

In the output directory, you will find the generated PLY files and corresponding log files for each sequence.

To ensure the PLY generation proceeded as expected, md5 checksums for meshes, the number of points and the md5 checksums for point clouds are provided for each frame of each sequence. These details are compiled into a single file per sequence and stored in ply\_generation/output\_info.

### D.X.4 Bitstream and objective metric generation

#### D.X.4.1 Dense dynamic point cloud

This clause assumes that all test sequences are available in the dense point cloud representation format as described in clause D.X.3.1. This clause describes how to execute the text environment using the provided scripts. Deeper information on the functioning of the scripts is given the documentation installed via Git. Interested users are referred to the document readme\_ply\_to\_bin in the doc folder.

##### D.X.4.1.1 Executing tests

Python scripts are provided to:

* Build the test environment under the output “dependencies” directory. The MPEG V-PCC test model [VOL-26] will be used to encode and decode test sequences. The MPEG mmetric software [DM-18] will be used to compute metrics. These tools are automatically downloaded and built by the script.
* Perform tests, including:
	+ Encode each sequence for each condition, rate point and profile.
	+ Decode the corresponding sequence.
	+ Compute the objective metrics.
	+ Generate CSV tables and graph worksheets.

To execute the tests, the user should navigate to the “ply\_to\_bin/” directory, which contains:

* \*.py: Python scripts to encode, decode, compute metrics and generate CSV and XLSM workbooks.
* templates/: Directory with template XLSM sheet used for graph generation.
* jsons/: Directory with configurations
	+ sequences.json: Describes the list of input sequences to test. It contains information on the location of point cloud sequences and has to be set by the user to point to the right location. It also has information on the name of the configuration file used for the encoding step (${test\_sequence}.cfg).
	+ 3gpp\_test\_configuration.json: Describes the test lists to perform. For each profile, it defines the encoding parameters ("—profileToolsetIdc, --profileReconstructionIdc, --mapCountMinus1), the number of frames to test (typically 300) and the list of sequences to be tested. This list includes:
		- The “id” corresponding to the one set in the sequences.json file.
		- The condition to test, here, random access.
		- A list of 5 rate points as defined in Table X.

Table X with QP selection for obtaining the fixed target bitrates

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Rate** | **Target Bitrate (mbps)** | **S01 Mitch** | **S02 Juggle Soccer** | **S03 Henry** | **S04 Nathalie** | **S05 Aliyah** |
|
|  |  | QPGeo | QPAtt | OccPrec | QPGeo | QPAtt | OccPrec | QPGeo | QPAtt | OccPrec | QPGeo | QPAtt | OccPrec | QPGeo | QPAtt | OccPrec |
| **R01** | 5 | 29 | 33 | 4 | 30 | 39 | 4 | 23 | 34 | 4 |  |  |  |  |  |  |
| **R02** | 10 | 23 | 29 | 2 | 19 | 35 | 2 | 15 | 30 | 2 |  |  |  |  |  |  |
| **R03** | 20 | 19 | 25 | 2 | 11 | 28 | 2 | 8 | 26 | 2 |  |  |  |  |  |  |
| **R04** | 30 | 15 | 23 | 2 | 9 | 24 | 2 | 7 | 23 | 2 |  |  |  |  |  |  |
| **R05** | 50 | 11 | 21 | 2 | 5 | 21 | 2 | 6 | 20 | 2 |  |  |  |  |  |  |

Editor’s note: Table to be completed for Nathalie and Aliyah

A script “exec\_binGenerator.py” is provided to automate all steps including encoding, decoding, objective metrics computation and output generation. It can be launched from your Python environment with the following command:

python exec\_binGenerator.py -o $YOUR\_OUTPUT\_DIR -i jsons/sequences.json -t jsons/test\_configuration.json

For help on the script see the complementary document readme\_ply\_to\_bin in the doc folder installed by Git.

The output directory structure is:

- cmd: Directory with job command and logs.

- dependencies: Compilation of TMC2 and mmetric software used to perform the test.

- A list “Fyy\_ProfileName” directories with Fyy corresponds to the number of tested frames, ProfileName corresponds to the tested profile and includes generated bitstreams.

- A list of CSV files with extracted metric information per profile for a given number of frames.

- Excel worksheets with graphs per profile for a given number of frames.

##### D.X.4.1.2 Objective results

CSV and workbook files are automatically generated by the scripts. The output log containing all metrics information is used to extract metrics and a build CSV files. Each CSV file concatenates metrics information for each condition and selected profile and is generated for all sequences and rate points.

The following information is stored in a CSV file:

* SeqId: identifier of the sequence
* CondId: tested condition (RA)
* RateId: tested rate number [R1..R5]
* nbFrame: number of tested frames
* NbInputPoints: number of points in the source sequence
* NbOutputPoints: number of points in the candidate test sequence
* MeanOutputPoints: mean number of points in the candidate test sequence
* MeanDuplicatePoints: mean number of duplicated points (with same geometry) in the candidate test sequence
* TotalBitstreamBits: size of the bistream in bits
* geometryBits: size of the geometry stream in bits
* metadataBits: size of the metadata stream in bits
* attributeBits: size of the attribute stream in bits
* D1Mean: mseF,PSNR (p2point)
* D2Mean: mseF,PSNR (p2plane)
* LumaMean: c[0],PSNRF
* CbMean: c[1],PSNRF
* CrMean: c[2],PSNRF
* PCQM: PCQM PSNR
* SelfEncoderRuntime: encoder time for current process
* ChildEncoderRuntime: encoder time for child processes
* SelfDecoderRuntime: decoder time for current process
* ChildDecoderRuntime: decoder time for child processes

From the CSV file, an excel spreadsheet is generated from the template xlsm sheet (in the “templates” directory) to get tables and graphs for interpretation of the results.

### D.X.5 Video generation

This clause describes how to generate 2D videos with a predefined camera path.

#### D.X.5.1 Dense dynamic point cloud

This clause assumes that test sequences are available either in raw dense point cloud format or as bitstream encoded with V-PCC. Please check clauses D.X.3 and D.X.4 on how to generate these inputs.

The provided scripts use the MPEG V-PCC test model [VOL-26] for decoding V-PCC bitstreams and the MPEG Representative Renderer [VOL-19] to generate videos from PLY files. Both are automatically cloned and built when running the scripts for the first time.

Detailed information on the functioning of the scripts is given the documentation installed via Git and users are referred to the document readme\_bin\_to\_video in the doc folder.

\* \* \* End of Changes \* \* \* \*