3GPP TSG-SA WG4 Meeting #133-eS4-251288

Online, 18 – 25 July 2025

**Source: InterDigital, Samsung, Sony Group Corporation, Nokia, Philips, Deutsche Telekom, Fraunhofer HHI, KDDI, Huawei**

**Title: pCR on corrections and completion in annexes for scenario 2**

**Spec: 3GPP TR 26.956 1.0.0**

**Agenda item: 9.6**

**Document for: Agreement**

**1. Introduction**

This pCR provides missing information, corrections and update of references in annexes for scenario 2.

**2. Reason for Change**

Provided information and corrections are essential for the completion of the TR.

**3. Proposal**

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

\* \* \* First Change \* \* \* \*

<Change as indicated>

### C.2.2 Juggle Soccer test sequence

#### C.2.2.1 Description

Soccer player with red shirt is showing soccer tricks with a ball. Particularity with the sequence is that a moving person and a ball are captured in one asset.

Figure C.2.2.1-1 Juggle Soccer - content courtesy XD Productions

#### C.2.2.2 Sequence properties

The tables C.2.2.2-1 and C.2.2.2-2 summarize the properties of the Joggle Soccer sequence

|  |  |
| --- | --- |
| Parameter | Value |
| Frame rate | 25 |
| #frames | 125 |
| Mean #point / frame | 1.883.637 |
| Attributes | RGB |
| Normals | Yes |
| Geometry Precision | 11 |
| Attribute Precision  | 8 |
| Normal Precision | Float |

Table C.2.2.2-1 Juggle Soccer sequence properties dense dynamic point cloud

|  |  |
| --- | --- |
| Parameter | Value |
| Frame rate | 25 |
| #triangles per frame | 80K |
| Texture resolution | 4K |
| #frames | 125 |

Table C2.2.2-2 Juggle Soccer sequence properties dynamic mesh

The sequence can be accessed: <https://aspera.pub/I4tSQ8k>

3GPP members can request credentials by sending a request per email to: 3GPP\_B2D\_Datasets@interdigital.com

#### C.2.2.3 Copyright and license information

XD Productions[Vol-22] kindly made this sequence freely available for 3GPP internal usage under license. License XD\_Productions\_-\_InterDigital\_Content\_license\_3GPP is provided in the directory with the sequence.

\* \* \* Next Change \* \* \* \*

<Change as indicated>

### C.2.3 Mitch test sequence

#### C.2.3.1 Description

Mitch is slacklining with slow movements allowing to check preserved details in tissue of the shirt and in the face.

Figure C.2.3.1-1 Mitch - content courtesy Volucap

#### 2.3.2 Sequence properties

The tables C.2.3.2-1 and C.2.3.2-2 summarize the properties of the Mitch sequence

|  |  |
| --- | --- |
| Parameter | Value |
| Frame rate | 25 |
| #frames | 475 |
| Mean #point / frame | 1.787.791 |
| Attributes | RGB |
| Normals | Yes |
| Geometry Precision | 11 |
| Attribute Precision  | 8 |
| Normal Precision | Float |

Table C.2.3.2-1 Mitch sequence properties dense dynamic point cloud

|  |  |
| --- | --- |
| Parameter | Value |
| Frame rate | 25 |
| #triangles per frame | 30K |
| Texture resolution | 4K |
| #frames | 475 |

Table C.2.3.2-2 Mitch sequence properties dynamic mesh

The sequence can be accessed: <https://aspera.pub/I4tSQ8k>

3GPP members can request credentials by sending a request per email to: 3GPP\_B2D\_Datasets@interdigital.com

#### C.2.3.3 Copyright and license information

Volucap [Vol-16] kindly made this sequence freely available for 3GPP internal usage under license. License “License\_Volucap\_T097\_Mitch2.1-05” is provided in the directory with the sequence.

\* \* \* Next Change \* \* \* \*

<Replace the last sentence as indicated, remainder unchanged>

#### C.2.4.2 Sequence properties

3GPP members can request credentials by sending a request per email to: 3GPP\_B2D\_Datasets@interdigital.com

\* \* \* Next Change \* \* \* \*

<Change as indicated>

### C.2.5 Nathalie test sequence

#### C.2.5.1 Description

Nathalie is performing a classic dance, as such the sequence is dynamic.

Figure C.2.5.1-1 Nathalie - content courtesy Volucap

#### C.2.5.2 Sequence properties

The tables C.2.5.2-1 and C.2.5.2-2 summarize the properties of the Nathalie sequence

|  |  |
| --- | --- |
| Parameter | Value |
| Frame rate | 30 |
| #frames | 925 |
| Mean #point / frame | 1.641.098 |
| Attributes | RGB |
| Normals | Yes |
| Geometry Precision | 11 |
| Attribute Precision  | 8 |
| Normal Precision | Float |

Table C.2.5.2-1 Nathalie sequence properties dense dynamic point cloud

|  |  |
| --- | --- |
| Parameter | Value |
| Frame rate | 30 |
| #triangles per frame | 30K |
| Texture resolution | 4K |
| #frames | 925 |

Table C.2.5.2-2 Nathalie sequence properties dynamic mesh

The sequence can be accessed: <https://aspera.pub/I4tSQ8k>

3GPP members can request credentials by sending a request per email to: 3GPP\_B2D\_Datasets@interdigital.com

#### C.2.5.3 Copyright and license information

Volucap [Vol-16] kindly made this sequence freely available for 3GPP internal usage under license. License “AOM\_License Volucap\_rp\_nathalie\_4d\_001\_dancing-20211214\_Gsplats” is provided in the directory with the sequence.

\* \* \* Next Change \* \* \* \*

<Replace the last sentence as indicated, remainder unchanged>

#### C.2.6.2 Sequence properties

3GPP members can request credentials by sending a request per email to: 3GPP\_B2D\_Datasets@interdigital.com

\* \* \* Next Change \* \* \* \*

<Change as indicated>

### C.2.7 Aliyah test sequence

#### C.2.7.1 Description

Aliyah is performing a modern dance, as such the sequence is pretty dynamic.

Figure C.2.7.1-1 DancingAliyah - content courtesy Renderpeople

#### C.2.7.2 Sequence properties

The tables C.2.7.2-1 and C.2.7.2-2 summarize the properties of the Aliyah sequence

|  |  |
| --- | --- |
| Parameter | Value |
| Frame rate | 30 |
| #frames | 1112 |
| Mean #point / frame | 1.732.973 |
| Attributes | RGB |
| Normals | Yes |
| Geometry Precision | 11 |
| Attribute Precision  | 8 |
| Normal Precision | Float |

Table C.2.7.2-1 Aliyah sequence dense dynamic point cloud

|  |  |
| --- | --- |
| Parameter | Value |
| Frame rate | 30 |
| #triangles per frame | 30K |
| Texture resolution | 4K |
| #frames | 1112 |

Table C.2.7.2-2 Aliyah sequence properties dynamic mesh

Renderpeople [Vol-23] provides a free and publicly downloadable “4D People” source sequence under license. This source sequence is provided in file formats for 3ds Max, Maya, Blender, Cinema 4D and Alembic and can be stored or converted to mesh or dense point cloud format.

The sequence can be accessed: <https://renderpeople.com/free-3d-people/>

Select then the free sequence under 4D People

#### C.2.7.3 Copyright and license information

General terms and conditions can be found here: <https://renderpeople.com/general-terms-and-conditions/>

\* \* \* Next Change \* \* \* \*

<Change as indicated>

### C.2.8 Henry test sequence

#### C.2.8.1 Description

Henry is performing a stretching exercise, as such the sequence is dynamic.

Figure C.2.8.1-1 Henry - content courtesy Renderpeople

#### C.2.8.2 Sequence properties

The tables C.2.8.2-1 and C.2.8.2-2 summarizes the properties of the Henry sequence.

|  |  |
| --- | --- |
| Parameter | Value |
| Frame rate | 30 |
| #frames | 733 |
| Mean #point / frame | 1.773.110 |
| Attributes | RGB |
| Normals | Yes |
| Geometry Precision | 11 |
| Attribute Precision  | 8 |
| Normal Precision | Float |

Table C.2.8.2-1 Henry sequence properties dense dynamic point cloud

|  |  |
| --- | --- |
| Parameter | Value |
| Frame rate | 30 |
| #triangles per frame | 30K |
| Texture resolution | 4K |
| #frames | 733 |

Table C.2.8.2-2 Henry sequence properties dynamic mesh

Renderpeople[Vol-24] provides a catalogue of currently 130 “4D People” under license and the catalog is growing. These source sequences are provided in file formats for 3ds Max, Maya, Blender, Cinema 4D and Alembic and can be stored or converted to mesh or dense point cloud format. Sequences from the 4D catalog are not free and need to be purchased. Henry is one of the sequences in the catalog that has been picked up as it is dynamic and different from the other presented sequences.

The “4D People” shop is accessible here: <https://renderpeople.com/3d-people/?_product=4d-people>

#### C.2.8.3 Copyright and license information

General terms and conditions can be found here: <https://renderpeople.com/general-terms-and-conditions/>

\* \* \* Next Change \* \* \* \*

<Modify annex D.3 as follows:>

## D.3 Scenario 2 Processing

### D.3.1 Overview

The generation of objective metrics and 2D videos for subjective viewing for scenario 2 is supported by a software package provided in the repository: <https://github.com/5G-MAG/rt-beyond2d-evaluation-framework> in the folder “point\_cloud”.

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.3.2 Installation

#### D.3.2.1 Cloning

git clone https://github.com/5G-MAG/rt-beyond2d-evaluation-framework

cd rt-beyond2D-evaluation-framework/point\_cloud

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.3.2.2 Working Directory

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

### D.3.3 Test sequence preparation

#### D.3.3.1 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/readme\_ply\_generation.md in the repository.

####  D.3.3.1.1 Generation of target dense dynamic point clouds

To proceed with the generation, the user needs to navigate to the /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 D.3.3.1.1-1. This JSON file needs to be updated for each sequence with the correct paths to the meshes for your environment (MeshObjPath and MeshTxtPath).

**Table D.3.3.1.1-1 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 ply\_generation/exec\_ply\_generation.py -i ply\_generation/jsons/3gpp\_selection.json -o $YOUR\_OUTPUT\_PATH

For help on the script see complementary document doc/readme\_ply\_generation.md 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.3.4 Bitstream and objective metric generation

#### D.3.4.1 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.3.3.1. 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 doc/readme\_ply\_to\_bin.md in the repository.

##### D.3.4.1.1 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 D.3.4.1.1-1.

**Table D.3.3.1.1-1 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 | 25 | 39 | 4 | 28 | 39 | 4 |
| **R02** | 10 | 23 | 29 | 2 | 19 | 35 | 2 | 15 | 30 | 2 | 24 | 30 | 4 | 20 | 32 | 4 |
| **R03** | 20 | 19 | 25 | 2 | 11 | 28 | 2 | 8 | 26 | 2 | 20 | 26 | 4 | 20 | 26 | 4 |
| **R04** | 30 | 15 | 23 | 2 | 9 | 24 | 2 | 7 | 23 | 2 | 18 | 24 | 2 | 18 | 24 | 2 |
| **R05** | 50 | 11 | 21 | 2 | 5 | 21 | 2 | 6 | 20 | 2 | 17 | 21 | 2 | 7 | 23 | 2 |

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.3.4.1.2 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.3.5 Video generation

#### D.3.5.1 Dense dynamic point cloud

This clause describes how to generate 2D videos with a predefined camera path. It is assumed that test sequences are available either in raw dense point cloud format or as bitstream encoded with V-PCC. Please check clauses D.3.3 and D.3.4 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.

To proceed with the video generation, the user needs to navigate to the /bin\_to\_video/ directory, which contains:

* \*.py: Python scripts for generating PLY (point cloud) files.
* jsons/: Directory with an example of input configuration files.

Multiple JSON files are available in the jsons/ directory:

* 3gpp\_selection\_src.json provides the information for the sources dense point clouds sequences. This JSON file needs to be updated for each sequence with the correct paths to the source .PLY files for your environment (PathDec parameter).
* 3gpp\_selection\_dec.json  provides the information for the encoded dense point cloud sequences. This JSON file needs to be updated for each sequence with the correct paths to the V-PCC encoded .BIN files for your environment (PathEnc parameter).
* 3gpp\_test\_configuration.json contains an example configuration to generate multiple videos with different MPEG Representative Renderer [VOL-19] settings for each sequence. The provided rendering settings are described in the Table D3

**Table D.3.5.1-1 with rendering settings for video generation**

|  |  |  |  |
| --- | --- | --- | --- |
| **Rendering Job Name**  | **Point Primitive**  | **Renderer Arguments**  | **Background**  |
| Cube size 1  | Cube  | --floor=1 --type=0 --size=1  | No  |
| Blend size 2.4 alpha 1.8 linear  | Linear blended splat  | --floor=1 --type=3 --alphaFalloff=1.8 --size=2.4 --blendMode=1  | No  |
| Bck blend size 2.4 alpha 1.8 linear  | Linear blended splat  | --type=3 --alphaFalloff=1.8 --size=2.4 --blendMode=1  | Yes  |

The JSON directory also contains the /camerapath/ and /background/ folders, providing additional configuration files used by the MPEG Representative Renderer [VOL-19].

* /camerapath/ contains files describing pre-recorded camera trajectories for each content
* /background/ contains files describing the position, orientation and scale of external 3D assets used as background for each content. These files need to be updated for each sequence with the correct path to the assets.

To generate the video of the sources:

python3 bin\_to\_video/exec\_binToVideo.py \
-c bin\_to\_video/jsons/3gpp\_test\_configuration.json \
-i bin\_to\_video/jsons/3gpp\_selection\_src.json \
-o $YOUR\_OUTPUT\_DIR -v

To generate the video of the encoded content:

python3 bin\_to\_video/exec\_binToVideo.py \
-c bin\_to\_video/jsons/3gpp\_test\_configuration.json \
-i bin\_to\_video/jsons/3gpp\_selection\_dec.json \
-o $YOUR\_OUTPUT\_DIR

The scripts generate uncompressed .RGB videos. For delivery purposes, the videos were compressed with an external tool to lossless HEVC. No such feature is delivered with this package.

Detailed information on the functioning of the scripts is given in the document doc/readme\_bin\_to\_video.md in the repository.

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