**Text

Description automatically generated** **ISO/IEC JTC 1/SC 29/WG 7 N512**

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

**Document type:** Output Document

**Title:** Guidelines for Conducting AI Exploration Experiments for PCC

**Status:** Approved

**Date of document:** 2023-02-10

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

**Expected action:** None

**Action due date:** None

**No. of pages:** 9 (with cover page)

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

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

**INTERNATIONAL ORGANIZATION FOR STANDARDIZATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

**ISO/IEC JTC 1/SC 29/WG 7 MPEG CODING FOR 3D GRAPHICS AND HAPTICS**

**ISO/IEC JTC 1/SC 29/WG 7 N512**

**Jan 2023, Online**

|  |  |
| --- | --- |
| **Title** | **Guidelines for Conducting AI Exploration Experiments for PCC** |
| **Source** | **WG7, MPEG 3D Graphics and Haptics** |
| **Status** | **Approved** |
| **Serial Number** | **22429** |

# Introduction

Due to the data-driven nature of AI-based PCC, conducting experiments on AI-based PCC differs from experimentation with non-learning-based methods. In this document, we provide concrete guidelines when conducting experiments on AI-based PCC.

In Section 2, common guidelines on both benchmark approaches and AI-based approaches are provided. Section 3 is on experiments for benchmark approaches (G-PCC and V-PCC), while Section 4 provides detailed guidelines on AI-based PCC methods. Then general guidelines related to datasets and crosscheck methodologies are discussed in Section 5 and Section 6, respectively.

# Common Guidelines

This section attempts to summarize the common test conditions for V-PCC/G-PCC and AI-based PCC. For reference, the detailed test condition and evaluation methods of different types of AI-based PCC methods are provided in [1] [2] [3].

## Data Format

Input point clouds are classified into dense and sparse categories in [4]. However, in addition and in contrast to the 2D domain input format that is largely designed for YUV, volumetric data may be represented in several different formats such as voxel-based, point-based or mesh-based. It is however understood that depending on the application, the use of a particular representation may become preferable. Thus, it is required not to limit the input volumetric data to be voxel-based data or downsampled data.

Note that the downsampling/quantization step used by the existing G-PCC CTC [5] should not prevent an AI-based (or next generation) encoder to have full access to the original point cloud at every rate point.

## Anchor Selection

In general, the use case will indicate the anchor selection. We may have different use cases and depending on which is being considered, a specific anchor may be more appropriate.

* For LiDAR point clouds from automotive applications, G-PCC should be selected as the anchor.
* For surface point clouds, including solid, dense, and sparse surface point clouds, both G-PCC and V-PCC are suggested as anchors.

Note that the current focus of the group is on compression. Hence, a task-oriented codec is not considered in the anchor selection at this point.

# Guidelines on G-PCC & V-PCC

## G-PCC

This section attempts to summarize the test conditions, in particular for the G-PCC codec. Proponents should apply G-PCC octree geometry coding when using G-PCC as the anchor. The test conditions of applying G-PCC to different test point clouds follow the usage of the G-PCC anchor in the Python-based reporting template [6]. Particularly, the angular mode should be enabled when it is applied to LiDAR point clouds. The reporting template is available via the following repository: <http://mpegx.int-evry.fr/software/MPEG/PCC/ai/mpeg-pcc-ai-report>.

## V-PCC

This section attempts to summarize the test conditions, in particular for the V-PCC codec. Unless specifically stated in this document, the V-PCC CTC [7] should be applied. For dense dynamic point cloud datasets, V-PCC in the random-access configuration should be applied.

# Guidelines on AI-based PCC

## General

Similar to non-learning-based methods, it is necessary to evaluate the bit rates and the objective quality metrics for AI-based PCC methods. For this purpose, proponents of AI-based PCC are required to use the Python-based reporting template [6] to report the compression performance.

However, different from non-learning-based methods, AI-based PCC typically includes training and inference phases. Thus, additional aspects need to be considered when evaluating an AI-based PCC method. Detailed training and inference information to be reported is summarized in Table 1 of the Annex; while evaluation metrics to be used are included in Table 2 of the Annex. We briefly highlight some of them as follows.

## Model Architecture

The designed neural network model architecture needs to be reported in detail.

To find out the best possible compression efficiency, different model architectures may be investigated for different types of point clouds during the exploration stage. The group may try to make the designs converge at a later stage, if possible, without compromising the performance.

## Dataset Choices

The datasets that are used for training, validation, and testing are specified in the dataset document [8]. However, it is also allowed to train and evaluate the AI model on proprietary datasets and report additional results.

The group will need to clarify for each public dataset about its availability, as well as whether it is allowed to be redistributed. For publicly available datasets, the group is considering copying them in the MPEG repository, if it is allowed by the license. If not allowed to be redistributed, an index of the public datasets with information on how to access them will be provided in the MPEG repository.

## Hyper-parameter Reporting

Hyper-parameters that are used to train the AI model need to be reported. Some common hyper-parameters include the learning rate policy, the batch size, the optimizer being used, loss functions, number of iterations, *etc.* If an AI model is obtained by multi-stage training, it is also necessary to report the details of each individual training stage.

## Test Conditions and Complexity Reporting

* **Platform information**: It is recommended to use PyTorch as the deep learning framework. PyTorch is easy to learn and work with, and suitable for developing prototypes rapidly. Additionally, it is recommended to include the version of PyTorch being used, the CUDA version, GPU type, CPU type, *etc*.
* **Training complexity**: training time (CPU and/or GPU) and memory consumption need to be reported. For different rate points, the model weights may be optimized per rate point. It is required to indicate whether the training is conducted for each rate point or a single model parameter set is shared for all rate points.
* **Inference complexity**: inference time (CPU and/or GPU), memory consumption, parameter number (model size).
* **Number of operations**: a metric based on the number of basic operations (*e.g.*, number of multiplications and accumulations) be also considered.

## Metrics

New metrics are being studied in EE 0.6 [9]. We may consider a two-phase approach: (1) Continue with current metrics where D1 and D2 are used, then (2) Introduce new metrics from the results of EE 0.6. A set of complete measurement metrics should consider the following aspects:

* **Point cloud geometry**:
  + **Metrics**: D1/D2 metrics should be included during evaluation.
  + **Scope**: The bitrate reporting (bpp) should be based on the original number of points of the input point cloud, for example, as defined in the current G-PCC CTC [5]. Non-normative procedures, such as upsampling/super-resolution should not be a mandatory part of the evaluation.
* **Point cloud attribute**: G-PCC CTC [5] should be followed to evaluate the attributes of point clouds.
* **Machine tasks**: AI-based PCC can be combined with machine tasks (*e.g.*, segmentation, detection) for end-to-end training [4]. Thus, when a machine vision task is of interest, it also becomes necessary to evaluate additional metrics related to the machine task. However, the current priority should be put on point cloud compression itself.
* **Generalization**: To understand the generalization ability of an AI-based model, it is recommended to report the quality metrics on a test dataset collected with a “different” source from the training dataset, though both of them should belong to the same application domain (*e.g.*, both are datasets for autonomous driving).

## Software Framework

It is recommended to implement the proposed method using the PccAI software framework [10]. PccAI is a testbed for AI-based PCC method. The software is available via the following repository: <http://mpegx.int-evry.fr/software/MPEG/PCC/ai/mpeg-pcc-pccai>.

## Dense Dynamic Point Clouds

Additional guidelines on the evaluation of dense dynamic point clouds using AI-based PCC are provided hereby. The guidelines on working with sparse dynamic point clouds will be made available in the future.

For evaluating the dense dynamic point clouds, the testing procedure that is described in [3] needs to be followed. Datasets are specified for training each method, as well as the datasets for obtaining the test results. The anchor to be used is V-PCC as described in section 3.2.

The proposed AI-based PCC methods need to use similar technologies for coding the intra frames as described in [3]. The intra frame specifications need to be reported, *e.g.*, per test rate point. Average bits per input point and average objective quality metric values need to be reported (Section 4.6), as well as additional information specified in this document.

## Point Cloud Attribute Coding

In general, the evaluation of AI-based point cloud attribute coding follows the CTC of G-PCC [5]. Detailed specifications can be found in [11]. Additional guidelines on working with point cloud attribute coding will be included in the future.

# Dataset Management

Please refer to the dataset document [8] for the recommended partitioning and categorization of the point cloud datasets. It is highly desirable to include additional point cloud data for MPEG. In the following, we provide guidelines on the usage of publicly available datasets and proprietary datasets as the training set.

## Public Datasets

Based on the contents of the public datasets, they are loosely categorized as generic and application-specific datasets. Depending on the use cases, both generic datasets and application-specific datasets can be used for training:

* **Generic datasets**: examples include ModelNet [12] and ShapeNet [13]. These generic datasets represent typical surfaces/geometric structures and are provided in the form of meshes. It has the flexibility to resample to a desired bit-depth and density.
* **Application-specific datasets**: examples contain the Ford sequence [5] and the KITTI dataset [14]. These datasets contain specific types of point clouds, such as sparse and high bit-depth LiDAR point clouds. They are targeted for specific applications such as autonomous driving.

We note that the pre-processing of a dataset should be related to particular use cases and the setup of the learning-based method. The pre-processing of the KITTI dataset is specified in the dataset document [8] while there are no common pre-processing steps for the rest of the datasets at this point.

## Proprietary Datasets

It is also allowed to use proprietary datasets for training the neural network models. In this case, the proponent should provide as many details about the proprietary dataset as possible, *e.g.*, size of the dataset, its format, attributes, *etc*.

However, performance of the models trained from proprietary datasets is solely for information purposes. Decisions will be made only based on the models trained from public datasets. This is mainly to prevent the misuse of public test data for training and reporting as training on proprietary datasets.

# Crosscheck Methodologies

A two-step crosscheck procedure is adopted:

* **Step 1**: partial crosscheck (inference only):
  1. Proponent provides their learning-based PCC codec, their trained models, and the corresponding R-D performance.
  2. Crosschecker performs inference with the given codec and the models on the public test data, then compares the results to the R-D performance provided by the proponent.
* **Step 2**: full crosscheck (train & inference):
  1. In addition to resources to perform partial crosscheck, proponent provides full instructions and code/scripts on training their models.
  2. Crosschecker trains new models with the provided instructions and code.
  3. Crosschecker performs inference on the trained models, then compares the results to the R-D performance provided by the proponent.

The 1st step, partial crosscheck, helps to screen out problematic proposals in the first place without consuming too many resources. Then the 2nd step, the full crosscheck, validates a proposal thoroughly. Adoption of any tool is subject to a full crosscheck.

We note that randomness exists in the training process and even during inference. Hence, it is difficult for the crosschecker to fully reproduce the results of the proponent. There are no particular solutions to this reproducibility issue. However, if there is a substantial mismatch between the results of the crosschecker and the proponent, it should be investigated whether the discrepancy comes from the randomness of training and inference.

# Annex

## Template for Reporting Training/Inference Information

Table 1. Information to be reported for evaluation.

|  |  |  |
| --- | --- | --- |
| Stage | Reporting Type | Parameters |
| Training stage  platform  information | Mandatory | GPU Type  CPU Type  Framework: PyTorch  Number of GPUs  Supporting software  RAM Size  Operating system |
|
| Training stage | Mandatory | Batch size  Loss functions  Learning rate policy  Training time (CPU/GPU)  Dataset choices  Epochs (or number of iterations)  Configuration per RD point  Pre-processing  Network Model:   * Network Visualization * Number of parameters * Parameter precision   Total Memory (MB)   * Peak Memory Usage (Total) * Peak Memory Usage (per Model) |
|
|
| Optional | Optimizer  Patch size  Mini-batch selection process  Network Model:   * Total Conv. Layers * Total FC Layers |
|
| Inference stage  platform  information | Mandatory | GPU Type  CPU Type  Framework: PyTorch  Number of GPUs  Supporting software  RAM Size  Operating system |
|
| Inference stage | Mandatory | Number of operations  CPU enc/dec times  GPU enc/dec times  Network Model (if different from training):   * Network Visualization * Number of parameters * Parameter precision   Total Memory (MB)   * Peak Memory Usage (Total) * Peak Memory Usage (per Model) |
|
| Optional | Patch size  Mini-batch selection process  Border handling  Network Model (if different from training):   * Total Conv. Layers * Total FC Layers |
| Other information |  |  |

Table 2. Evaluation metrics.

|  |  |
| --- | --- |
| Type | Metrics |
| Geometry | D1 point-to-point distance  D2 point-to-plane distance  Geometry bitrate |
| Attribute | G-PCC CTC |
| Other | New metrics studied in EE 0.6 [7] |

## Parameter Definitions

Definitions of the parameters are provided as follows:

**Batch size**: Number of samples processed before the model is updated (training) or the number of samples processed in parallel during inference (inference).

**Border handling**: Description on boundary handling if inference operates on a block.

**Configuration per rate-distortion point:** Any changes in the requested information used to generate different rate-distortion points.

**CPU/GPU enc/dec times:** inference encoding and decoding runtimes.

**Epoch:** Number of complete passes through the training data.

**Framework:** Neural network development framework (PyTorch).

**Loss function:** Function used to calculate the model error during training and optimization.

**Learning rate:** Amount that the weights are updated during training.

**Mini-batch Selection Procedure:** Description of mini-batch selection procedure.

**Network Visualization:** Graphical representation of the neural network

**Number of Iterations:** Number of gradient updates within an epoch.

**Number of parameters:** Total number of parameters of the model.

**Optimizer:** Algorithm used to change the attributes of proposed neural networks.

**Parameter Precision:** Precision of the parameters (*e.g.*, integer 8 bits, float 32 bits)

**Patch size:** Size of input to the neural networks during training and inference.

**Pre-processing:** Preprocessing procedure, normalization, cropping method, rotation, zoom *etc*.

**Number of operations:** Number of multiply–accumulate (MAC) operations per sample in the inference stage. A sample corresponds to one luma value or one chroma value.

**Total Conv. Layers:** Total numbers of convolutional layers in the network structure. If there is no convolutional layer, just fill in 0.

**Total FC Layers:** Total number of fully-connected layers in the network structure. If there is no fully-connected layer, just fill in 0.

**Total memory (MB):** Temporary memory used for inference. The temporary memory shall be reported as (i) the total memory size required for inference and (ii) the maximum memory size per model, if the proposal employs multiple network models in its design.

**Training time:** CPU and/or GPU training runtimes.

# References

|  |  |
| --- | --- |
| [1] | WG7, MPEG 3DGH, *EE 5.1 Deep Octree Coding,* Mainz: MPEG w22102, Oct 2022. |
| [2] | WG7, MPEG 3DGH, *EE 5.2 Deep feature based PC Coding,* Mainz: MPEG w22103, Oct 2022. |
| [3] | WG7 MPEG 3DGH, *EE 5.3 AI-based Dynamic PC Coding,* Online: MPEG w22104, Oct 2022. |
| [4] | J. Pang, M. A. Lodhi, G. Martin-Cocher and D. Tian, *[PCC] AI-based Point Cloud Compression for New PCC,* Online: MPEG m56776, Apr 2021. |
| [5] | WG7, MPEG 3DG, *Common Test Conditions for G-PCC,* Online: MPEG w20358, Apr 2021. |
| [6] | M. A. Lodhi, J. Wang, J. Pang, A. Zaghetto and A. Akhtar, *[AI-3DGC][EE5.0 related] On Reporting Template for AI-based PCC,* Online: MPEG m61197, Oct 2022. |
| [7] | WG7, MPEG 3DG, *Common Test Conditions for V-PCC,* Online: MPEG w19613, Oct 2020. |
| [8] | WG7, MPEG 3DGH, *Preliminary data set collection for AI experiments,* Online: MPEG w22431, Jan 2023. |
| [9] | WG7, MPEG 3DG, *EE 0.6 on Evaluating Alternative Metrics for Point Cloud Coding,* Online: MPEG w20760, July 2021. |
| [10] | J. Pang, M. A. Lodhi and D. Tian, *[AI-3DGC][EE13.54-related] PccAI: A Software Framework for AI-based PCC,* Online: MPEG m58961, Jan 2021. |
| [11] | WG7, MPEG 3DGH, *EE 5.4 AI-based attribute coding,* Mainz: MPEG w22189, Oct 2022. |
| [12] | Z. Wu, S. Song and et.al., "3D shapenets: A deep representation for volumetric shapes," in *CVPR*, 2015. |
| [13] | A. X. Chang, T. Funkhouse and et.al., "ShapeNet: An Information-Rich 3D Model Repository," in *ICCV*, 2017. |
| [14] | KITTI, "The KITTI Vision Benchmark Suite," [Online]. Available: http://www.cvlibs.net/datasets/kitti/. |