ISO/IEC JTC 1/SC 29/WG 11

Coding of moving pictures and audio

Convenorship: UNI (Italy)

Document type:	Approved WG 11 document
Title:	G-PCC test model v8 user manual
Status:	Final
Date of document:	2019-12-06
Source:	3DG
Expected action:	None
No. of pages:	12
Email of convenor:	leonardo@chiariglione.org
Committee URL:	mpeg.chiariglione.org

INTERNATIONAL ORGANISATION FOR STANDARDISATION
ORGANISATION INTERNATIONALE DE NORMALISATION
ISO/IEC JTC 1/SC 29/WG 11
CODING OF MOVING PICTURES AND AUDIO

Source: 3DG

Title: G-PCC test model v8 user manual

Abstract

This document is a user manual describing usage of reference software for the G-PCC project. It applies to version 8.0 of the software.

Contents

Date saved: 2019-12-06

General Information

Reference software is being made available to provide a reference implementation of the G-PCC standard being developed by MPEG (ISO/IEC SC29 WG11). One of the main goals of the reference software is to provide a basis upon which to conduct experiments in order to determine which coding tools provide desired coding performance. It is not meant to be a particularly efficient implementation of anything, and one may notice its apparent unsuitability for a particular use. It should not be construed to be a reflection of how complex a production-quality implementation of a future G-PCC standard would be.

This document aims to provide guidance on the usage of the reference software. It is widely suspected to be incomplete and suggestions for improvements are welcome. Such suggestions and general inquiries may be sent to the general MPEG 3DGC email reflector at mpeg-3dgc@gti.ssr.upm.es (registration required).

Bug reporting

Bugs should be reported on the issue tracker set up at http://mpegx.int-evry.fr/software/MPEG/PCC/TM/mpeg-pcc-tmc13/issues.

Obtaining the software

The authoritative location of the software is the following git repository: http://mpegx.int-evry.fr/software/MPEG/PCC/TM/mpeg-pcc-tmc13

Each released version may be identified by a version control system tag in the form `release-v${version}`.

An example:

```
$ git clone http://mpegx.int-evry.fr/software/MPEG/PCC/TM/mpeg-pcc-tmc13.git
$ cd mpeg-pcc-tmc13
$ git checkout release-v4.0
```

> It is strongly advised to obtain the software using the version control system rather than to download a zip (or other archive) of a particular release. The build system uses the version control system to accurately identify the version being built.

Building

The codec is supported on Linux, OSX and Windows platforms. The build configuration is managed using CMake.

> It is strongly advised to build the software in a separate build directory.

Linux

```
$ mkdir build
$ cd build
$ cmake ..
$ make
$ tmc3/tmc3 --help
```

OSX

```
$ mkdir build
$ cd build
$ cmake .. -G Xcode
$ xcodebuild
$ tmc3/tmc3 --help
```

As an alternative, the generated XCode project may be opened and built from XCode itself.

Windows

```
> md build
> cd build
> cmake .. -G "Visual Studio 15 2017 Win64"
```

Open the generated visual studio solution to build it.

Using the codec

```
./tmc3 [--help] [-c config.cfg] [--parameter=value]
```

The encoder takes as input one or more PLY files describing a point cloud sequence with integer positions and, optionally, per-point integer colour and reflectance attributes.

The output of the encoder is a binary bitstream encapsulated using the G-PCC annex-B format.

Conversely, the decoder takes as input a compressed bitstream file in G-PCC annex-B format and produces one or more reconstructed PLY file with position and any present attribute values.

The software may be configured using either command line arguments or from a configuration file specified using the -c|--config= option.

Sample configuration files are provided in the cfg/ directory. The utility <scripts/gen-cfg.sh> may be used to generate per sequence and per rate point configuration files for a variety of common test conditions.

Parameters are set by the last value encountered on the command line. Therefore if a setting is set via a configuration file, and then a subsequent command line parameter changes that same setting, the command line parameter value will be used.

General options

`--help`

Print a list of available command line (and configuration file) options along with their default values and exit.

`--config=FILE, -c`

This specifies a configuration file to be immediately loaded.

`--mode=VALUE`

This option selects the codec's mode of operation. A value of 0 enables encoding functionality. A value of 1 switches to decoding mode.

I/O parameters

`--firstFrameNum=INT-VALUE`

The initial frame number of the input or output sequence. The software replaces any instance of a '%d' printf format directive with the current frame number when evaluating the following options:

- uncompressedDataPath

- reconstructedDataPath

- postRecolourPath

- preInvScalePath

NB: When decoding, this option relates only to the output file names.

In order to have the decoder produce identically numbered output ply files as the encoder input, specify the same value of firstFrameNum for the decoder.

`--frameCount=INT-VALUE`

(Encoder only) The number of frames to be encoded.

`--uncompressedDataPath=FILE`

(Encoder only) The input source point cloud to be compressed. The first instance of '%d' in FILE will be expanded with the current frame number.

`--compressedStreamPath=FILE`

The compressed bitstream file output when encoding or input when decoding.

`--reconstructedDataPath=FILE`

The reconstructed point cloud file. When encoding, the output is the locally decoded picture. It is expected that the reconstructed output of the encoder and decoder match exactly.

The first instance of '%d' in FILE will be expanded with the current frame number.

`--postRecolourPath=FILE`

(Encoder only) As part of the encoding process, it may be necessary to re-colour the point cloud if the point geometry is altered. This diagnostic output file corresponds to the re-coloured point cloud prior to attribute coding without output geometry scaling.

The first instance of '%d' in FILE will be expanded with the current frame number.

`--preInvScalePath=FILE`

(Decoder only) This diagnostic output corresponds to the decoded point cloud (geometry and attributes) prior to output geometry scaling.

When compared to the output of `postRecolourPath`, the performance of attribute coding may be directly measured without being confounded by any geometry losses.

The first instance of '%d' in FILE will be expanded with the current frame number.

`--outputBinaryPly=0|1`

Sets the output format of PLY files (Binary=1, ASCII=0). Reading and writing binary PLY files is more efficient than the ASCII variant, but are less suited to simple scripts and direct human inspection.

If outputting non-integer point co-ordinates (eg, due to the output geometry scaling), the precision of the binary and ASCII versions are not identical.

Date saved: 2019-12-06

`--convertPlyColourspace=0|1`

Controls the conversion of ply RGB colour attributes to/from the colourspace set by an attribute's `colourMatrix` before attribute coding and after decoding. When disabled (0), or if there is no converter available for the requested `colourMatrix`, no conversion happens; however the `colourMatrix` value is still written to the bitstream.

`--hack.reflectanceScale=0|1`

Some input data uses 8-bit reflectance data scaled by 255 and represented using 16-bit attributes. This option enables a conversion of 16-bit to 8-bit at the encoder, and the corresponding conversion from 8-bit back to 16-bit at the decoder. If the original data has been scaled by 255, the conversion process is lossless.

Decoder-specific options

`--minGeomNodeSizeLog2=INT-VALUE`

The option indicates the number of skipped lod layers from leaf lod. If aps.scalable_enable_flag is 1, the option is valid. Otherwise, the option is ignored.

Encoder-specific options

`--positionQuantizationScale=REAL-FACTOR`

Prior to encoding, scale the point cloud geometry by multiplying each co-ordinate by the real *FACTOR* and rounding to integer precision. The scale factor is written to the bitstream and a decoder may use it to provide output at the original scale.

NB: when using trisoup geometry coding, use `triSoupIntToOrigScale` instead of this option.

`--positionQuantizationScaleAdjustsDist2=0|1`

This option simplifies the specification of the per-attribute `dist2` parameter.

The squared distance threshold used for generating levels-of-detail in attribute coding is dependent on the point cloud density and is therefore affected by geometry quantization. When this parameter is enabled, `dist2` values are scaled by `positionQuantizationScale` squared, thereby allowing `dist2` to be specified as an intrinsic property of the source sequence.

`--seq_bounding_box_xyz0=x,y,z`

Explicitly sets the origin of the sequence-level bounding box in unscaled integer coordinates.

NB: This option has no effect if `seq_bounding_box_whd=0,0,0`.

`--seq_bounding_box_whd=w,h,d`

Explicitly sets the size of the sequence-level bounding box in unscaled integer coordinates.

When w, h, d not equal to 0,0,0, the sequence-level bounding box origin is set according to `seq_bounding_box_xyz0`. Otherwise, the sequence-level bounding box is determined by the encoder.

`--mergeDuplicatedPoints=0|1`

Controls the ability to code duplicate points. When duplicate point merging is enabled, bitstream syntax related to duplicate points is disabled and a pre-filtering process is used to remove co-located points.

`--geometry_axis_order=INT-VALUE`

Configures the order in which axes are internally coded. Changing the axis order does not change the orientation of the reconstructed point cloud.

Value	Coding order
0	z, y, x

Date saved: 2019-12-06

Value	Coding order
1	x, y, z
2	x, z, y
3	y, z, x
4	z, y, x
5	z, x, y
6	y, x, z
7	x, y, z

`--disableAttributeCoding=0|1`

This option instructs the encoder to ignore all options relating to attribute coding, as if they had never been configured.

`--partitionMethod=0|2|3`

Selects the partitioning method to map points to tiles and slices:

Value	Description
0	none (single slice)
2	uniform partitioning along longest edge
3	uniform octree partitions

`--partitionNumUniformGeom=INT-VALUE`

Sets the number of slices to generate using `partitionMethod=2`. If equal to zero, the number of slices is the integer ratio of the longest to shortest edges of the point cloud bounding box.

`--partitionOctreeDepth=INT-VALUE`

Sets the depth of the octree for slice generation using `partitionMethod=3`.

The input point cloud is decomposed using an octree with the configured depth. Each occupied leaf of the octree represents a single slice.

`--sliceMaxPoints=INT-VALUE`

Upper limit to the number of in each slice. Slices are split until this contraint is satisfied.

`--sliceMinPoints=INT-VALUE`

Minimum number of points in each slice. This soft limit is used to merge small slices together.

`--tileSize=INT-VALUE`

Tile dimension to use when performing initial partitioning. A value of zero disables tile partitioning.

`--cabac_bypass_stream_enabled_flag=0|1`

Controls the entropy coding method used for equi-probable (bypass) bins:

Value	Description
0	bypass bins coded using CABAC
1	bypass bins coded in bypass substream

Date saved: 2019-12-06

Geometry coding

`--positionQuantisationEnabled=0|1`

Enables in-loop quantisation and reconstruction of geometry positions.

NB: All in-loop quantisation is independent (and happens after) any position scaling due to `positionQuantizationScal`

`--positionBaseQp=INT-VALUE`

The quantisation parameter used to quantise geometry positions. The effective QP may be varied according to `positionSliceQpOffset` and `positionQuantisationOctreeDepth`. A QP equal to 4 results in a scale factor of 1.

`--positionSliceQpOffset=INT-VALUE`

A per-slice offset to be applied to `positionBaseQp`.

`--positionQuantisationOctreeDepth=INT-VALUE`

The depth in the octree at which per-node QP offsets are signalled. A non-normative encoder process determines the QP offset based upon the local density of the octree. A value of -1 disables signalling of per-node QP offsets.

`implicitQtBtEnabled=0|1`

Enables non-cubic geometry tree coding. When enabled, the geometry tree may have a cuboid bounding box. Internal tree nodes are implicitly determined to be octree, quadtree or binary tree nodes.

`max_num_implicit_qtbt_before_ot=INT-VALUE`

Limits the maximal number of implicit quadtree and binary tree partitions used before the first octree partition.

`min_implicit_qtbt_size_log2=INT-VALUE`

Specifies the minimum size of implicit quadtree and binary tree partitions

`--bitwiseOccupancyCoding=0|1`

In octree geometry coding, there are both byte-wise and bit-wise tools to encode the occupancy data. This option selects between the two methods.

`--neighbourContextRestriction=0|1`

Octree occupancy coding is contextualised in part by the occupancy of neighbouring octree nodes. The neighbour context restriction limits the use of neighbouring nodes to direct octree siblings.

NB: This option conflicts with `neighbourAvailBoundaryLog2`. It is necessary to set `neighbourAvailBoundaryLog2=` when `neighbourContextRestriction=1`.

`--neighbourAvailBoundaryLog2=INT-VALUE`

Defines the volume within which octree nodes are considered available for use in occupancy contextualisation and intra occupancy prediction.

A value of 0 indicates that no constraint is applied.

The software currently supports a maximum value of 8 or 9 when intra occupancy prediction prediction is enabled or disabled respectively.

`--inferredDirectCodingMode=0|1`

Controls the use of early termination of the geometry octree for isolated points.

`--adjacentChildContextualization=0|1`

Controls the contextualization of occupancy bits according to the state of adjacent children of neighbouring nodes.

`--intra_pred_max_node_size_log2=INT-VALUE`

Intra occupancy prediction uses an octree node's neighbours to predict its occupancy. The prediction mode is enabled for octree nodes smaller than or equal to the configured size. A value of 0 disables intra occupancy prediction.

`--planarEnabled=0|1`

Controls the use of planar coding mode for geometry occupancy.

`--planarModeIdcmUse=0--127`

Controls the frequency of IDCM eligibility. A value of zero prevents IDCM being eligible for use, while the value 127 does not further constrain IDCM use.

`--planarModeThreshold0=0--127`

Controls the eligibility threshold of the first planar mode based upon local child node density.

`--planarModeThreshold1=0--127`

Controls the eligibility threshold of the second planar mode based upon local child node density.

`--planarModeThreshold2=0--127`

Controls the eligibility threshold of the third planar mode based upon local child node density.

`--ctxOccupancyReductionFactor=INT-VALUE`

Adjusts the number of contexts used in bit-wise occupancy coding. The total number of contexts used is 256 » *VALUE*.

NB: the final standard is expected to define this factor as a constant.

`--trisoup_node_size_log2=INT-VALUE`

Controls the use of trisoup by setting the node size for triangle based surface reconstruction. The trisoup method terminates the octree coding at the given node size and continues by encoding triangles which are subsequently voxelised to produce points.

A value of 0 disables the use of trisoup.

Attribute coding

The codec may be configured to represent one or more attributes. The configuration of each attribute is independent from all others. To configure coding of an attribute, first set the attribute options, then save the configuration using the `attribute` option.

`--attribute=NAME`

Saves the current attribute configuration for coding the named attribute.

Name	Description
colour	r, g, and b properties as a tri-stimulus attribute
reflectance	refc or reflectance property as a single-stimulus attribute

This option must be specified after the options corresponding to the attribute.

`--colourMatrix=INT-VALUE`

Indicates the colourspace of the coded attribute values according to the ISO/IEC 23001-8 Codec Independent Code Points for ColourMatrix. When used in conjunction with `convertPlyColourspace=1`, a colourspace conversion will be performed at the input/output of the encoder and decoder if supported.

Value	RGB converter	Description
0	n/a	Direct coding (eg, RGB, XYZ)
1	Yes	YCbCr ITU-R BT.709
2	n/a	Unspecified
3	n/a	Reserved
4	No	USA Title 47 CFR 73.682 (a)(20)
5	No	YCbCr ITU-R BT.601
6	No	YCbCr SMPTE 170M
7	No	YCbCr SMPTE 240M
8	Yes (YCgCoR)	YCgCo / YCgCoR
9	No	YCbCr ITU-R BT.2020
10	No	YCbCr ITU-R BT.2020 (constant luminance)
11	No	YDzDx SMPTE ST 2085

NB: the use of YCgCoR and `bitdepth=N` implies that the bitdepth of the chroma component bitdepth is N + 1.

`--bitdepth=INT-VALUE`

The bitdepth of the attribute data. NB, this is not necessarily the same as the bitdepth of the PLY property.

`--transformType=0|1|2`

Coding method to use for the current attribute:

Value	Description
0	Hierarchical neighbourhood prediction
1	Region Adaptive Hierarchical Transform (RAHT)
2	Hierarchical neighbourhood prediction as lifting transform

`--rahtDepth=INT-VALUE`

Number of bits for Morton representation of RAHT co-ordinate components.

`--rahtPredictionEnabled=0|1`

Controls the use of transform domain prediction of RAHT coefficients from spatially upsampling the DC values of neighbouring parent nodes in the transform tree.

`--numberOfNearestNeighboursInPrediction=INT-VALUE`

Attribute's maximum number of nearest neighbours to be used for prediction.

`--adaptivePredictionThreshold=INT-VALUE`

Neighbouring attribute value difference that enables choice of single|multi predictors. Applies to transformType=2 only. A value of -1 is replaced by 2**(bitdepth-2).

`--interComponentPredictionEnabled=0|1`

Controls the use of an in-loop inter-component prediction of attribute residuals. Applies to `transformType=1` and `attribute=color` only.

`--attributeSearchRange=INT-VALUE`

Range for nearest neighbour search.

`--max_num_direct_predictors=INT-VALUE`

Maximum number of nearest neighbour candidates used in direct attribute prediction.

`--lodDecimation=0|1`

Controls the level-of-detail generation method:

Value	Description
0	Euclidean distance thresholding
1	Decimation by 1:3

`--intraLodPredictionEnabled=0|1`

Part of LoD attribute coding. Permits (1) points to be predicted from previously reconstructed points within the same LoD.

`--aps_scalable_enabled_flag=0|1`

Enable spatially scalable attribute encoding. The option is only valid when `transformType=2`, `lodDecimation=0`, and `trisoup_node_size_log2=0`.

`--levelOfDetailCount=INT-VALUE`

Attribute's number of levels of detail.

`--dist2=INT-VALUE|INT-VALUE-LIST`

Attribute's list of squared distances, or initial value for automatic derivation.

`--lod_neigh_bias=INT-VALUE-LIST`

A set of three bias factors corresponding to the first, second and third geometry axes used to weight nearest neighbours during the LoD generation and weighting processes. The value `1,1,1` implies no bias.

`--qp=INT-VALUE`

Attribute's luma quantization parameter.

`--qpChromaOffset=INT-VALUE`

Attribute's chroma quantization quantization parameter relative to luma. Only applies when `attribute=colour`.

`--aps_slice_qp_deltas_present_flag=0|1`

Enables signalling of per-slice QP values.

`--qpLayerOffsetsLuma=INT-VALUE-LIST`

Attribute's per layer luma QP offsets. A layer is corresponds to a level-of-detail or RAHT transform block.

`--qpLayerOffsetsChroma=INT-VALUE-LIST`

Attribute's per layer chroma QP offsets. A layer is corresponds to a level-of-detail or RAHT transform block. Only applies when `attribute=colour`.

Attribute recolouring (encoder only)

The following options configure the recolouring module, used when resampling a point cloud, or if the geometry coding process invents new points.

`--recolourSearchRange=INT-VALUE`

Attribute space search range for optimal attribute transfer.

`--recolourNumNeighboursFwd=INT-VALUE`

Number of source points used at the neighborhood of a target point to create the forward points list.

`--recolourNumNeighboursBwd=INT-VALUE`

Number of target points used at the neighborhood of a source point to create the backward points list.

Date saved: 2019-12-06

`--recolourUseDistWeightedAvgFwd=0|1`

Use distance-weighted average for forward list.

`--recolourUseDistWeightedAvgBwd=0|1`

Use distance-weighted average for backward list.

`--recolourSkipAvgIfIdenticalSourcePointPresentFwd=0|1`

Do not use forward points list if an identical source point exists.

`--recolourSkipAvgIfIdenticalSourcePointPresentBwd=0|1`

Do not use backward points list if an identical source point exists.

`--recolourDistOffsetFwd=REAL-VALUE`

Distance offset to avoid infinite weight when distance between a forward list point and the target is zero.

`--recolourDistOffsetBwd=REAL-VALUE`

Distance offset to avoid infinite weight when distance between a backward list point and target is zero.

`--recolourMaxGeometryDist2Fwd=REAL-VALUE`

Maximum allowed squared distance of a source point from target to get into the forward list.

`--recolourMaxGeometryDist2Bwd=REAL-VALUE`

Maximum allowed squared distance of a source point from target to get into the backward list.

`--recolourMaxAttributeDist2Fwd=REAL-VALUE`

Maximum allowed squared attribute value difference of a source point for inclusion in the forward list.

`--recolourMaxAttributeDist2Bwd=REAL-VALUE`

Maximum allowed squared attribute value difference of a source point for inclusion in the backward list.