ISO/IEC JTC 1/SC 29/WG 7 **N00094**

ISO/IEC JTC 1/SC 29/WG 7

MPEG 3D Graphics Coding

Convenorship: AFNOR (France)

Document type:	Approved WG 7 document
Title:	G-PCC test model v14 user manual
Status:	Final
Date of document:	2021-06-29
Source:	WG7
Expected action:	None
No. of pages:	19
Email of convenor:	marius.preda@imt.fr
Committee URL:	https://isotc.iso.org/livelink/livelink/open/jtc1sc29wg7

INTERNATIONAL ORGANISATION FOR STANDARDISATION
ORGANISATION INTERNATIONALE DE NORMALISATION
ISO/IEC JTC 1/SC 29/WG 7
MPEG 3D GRAPHICS CODING

ISO/IEC JTC 1/SC 29/WG 7 N00094
Online – April 2021

Title: G-PCC test model v14 user manual
Source: WG7
Serial Number: 20346

Abstract

This document is a user manual describing usage of reference software for the G-PCC project. It applies to version 14.0 of the software.

Contents

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.

`--outputSystem=0|1`

Controls the output scaling of the coded point cloud.

Value	Description
0	Conformance output
1	External co-ordinate system

The conformance output scales the coded point cloud to the sequence co-ordinate system. The output point positions are not offset by the sequence origin.

The external co-ordinate system output scales the point cloud to the defined external co-ordinate system (see `sequenceScale`, `externalScale`, and `outputUnitLength`). The output point positions are offset by the sequence origin, appropriately scaled.

`--outputUnitLength=REAL-VALUE`

The length of the output point cloud unit vector. Point clouds output by the encoder or decoder are rescaled to match this length.

For example, `outputUnitLength=1000` outputs a point cloud with integer point positions representing millimetres.

`--outputPrecisionBits=INT-VALUE`

The number of fractional bits to retain when scaling from the coding co-ordinate system to the sequence co-ordinate system. The fractional bits are further retained when converting to the external co-ordinate system.

The special value `outputPrecisionBits=-1` retains all fractional bits during the scaling process.

`--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.

Decoder-specific options

`--skipOctreeLayers=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.

`--decodeMaxPoints=INT-VALUE`

A value greater than zero controls the automatic derivation of `skipOctreeLayers` such that at most n points are decoded. This option only has an effect if the bitstream contains per octree level point count metadata (see `pointCountMetadata`).

Encoder-specific options

Co-ordinate systems and pre-scaling

`--srcUnit=0|1|metre`

The physical unit used to interpret values of `srcUnitLength`.

Value	Description
0	dimensionless

Value	Description
1,metre	metre

`--srcUnitLength=REAL-VALUE`

The length of the source point cloud unit vector. This value is used to define the unit vector length of the sequence co-ordinate system. It is not used to perform scaling by the encoder.

For example, `srcUnitLength=1000` and `srcUnit=metre` indicates that integer positions in the source point cloud represent millimetres.

`--inputScale=REAL-VALUE`

A scale factor applied to point positions in the source point cloud prior to integer conversion. The `inputScale` changes the length of the source unit vectors (as set by `srcUnitLength`).

For example, a point cloud may have a unit vector representing 1 metre (`srcUnitLength=1`) and contain points with a resolution of 1000 points per metre. Since the codec can only represent integer positions, without input scaling, it is coded with a precision of one metre. Setting `inputScale=1000` will increase the precision to 1 millimetre.

`--codingScale=REAL-VALUE`

A scale factor used to determine the length of the coding co-ordinate system unit vector. The scale factor is relative to `inputScale`. The input point cloud (after integer conversion) is scaled by `codingScale` and rounded to integer positions.

If `codingScale` is greater than `sequenceScale`, the encoder will set `codingScale=sequenceScale`.

A decoder will scale the coded point cloud by `sequenceScale/codingScale` prior to output.

`--sequenceScale=REAL-VALUE`

A scale factor used to determine the length of the sequence co-ordinate system unit vector. The scale factor is relative to `inputScale`. The input point cloud (after integer conversion) is scaled by the smallest of `sequenceScale` and `codingScale`.

`--externalScale=REAL-VALUE`

A scale factor used to define the length of the sequence co-ordinate system when `srcUnit` is dimensionless. The scale factor is relative to `inputScale`. The `externalScale` does not affect scaling of the input point cloud prior to coding.

For example, a point cloud coded with `sequenceScale=0.25` and `externalScale=0.5` specifies that:

- the input is scaled by 0.25 prior to coding, and
- the decoder is informed that 1 sequence unit is equal to 2 external units.

NB: a decoder is not required to scale the sequence co-ordinate system to an external co-ordinate system prior to output.

`--autoSeqBbox=0|1`

Automatically determine the sequence bounding box (`seqOrigin` and `seqSizeWhd`) using the first input frame.

`--seqOrigin=x,y,z`

Sets the origin of the sequence bounding box. The `seqOrigin` must be less than or equal to the lowest input point position. The origin is configured in the input co-ordinate system (after integer conversion). The encoder will adjust the origin according to `sequenceScale`.

This option has no effect when `autoSeqBbox=1`.

`--seqSizeWhd=w,h,d`

Sets the size of the sequence bounding box. The size is configured in the input co-ordinate system (after integer conversion). The encoder will adjust the size according to `sequenceScale`.

`seqSizeWhd=0,0,0` disables signalling the sequence bounding box size.

This option has no effect when `autoSeqBbox=1`.

`--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.

`--sortInputByAzimuth=0|1`

Pre-sort the input point cloud according to azimuth angle with the origin `lidarHeadPosition`. Pre-sorting occurs prior to tile/slice partitioning.

Input partitioning (slices & tiles)

`--partitionMethod=0|2|3|4|5`

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
4	uniform square partitions
5	n-point spans

Uniform longest edge partitioning slices the point cloud along the longest edge according to `partitionNumUniformGeom`.

Uniform octree partitioning generates slices with the same size based on an octree partitioning of the point cloud according to `partitionOctreeDepth`.

Uniform square partitioning generates cubic slices sized according to the shortest edge.

N-point span partitioning divides the input point list (after input pre-sorting) into `sliceMaxPoints`-point sublists. Input order (after pre-sorting) is maintained.

In all cases, a refinement process may merge or split slices in order to satisfy maximum or minimum points per slice constraints.

`--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 constraint 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.

General options

`--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
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.

`--enforceLevelLimits=0|1`

Controls the enforcement of level limits by the encoder. If a level limit is voilated, the encoder will abort.

`--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

`--entropyContinuationEnabled=0|1`

Controls the propagation of entropy coding state (context values) between slices in the same frame. When enabled, each slice (except the first) has a coding dependency on the previous slice.

Geometry coding

`--geomTreeType=0|1`

Selects the geometry tree coding method.

Value	Description
0	Octree
1	Predictive geometry tree

--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`

--positionQuantisationMethod=0|1|2

Selects the method used to determine the QP value for each quantised tree node.

Value	Description
0	Uniform
1	Random
2	By point density

The 'uniform' method sets every node QP to the slice QP.

The 'random' method picks a uniformly distributed random QP for each node from the range of permitted values. The seed for random number generation may be set using the environment variable `SEED`.

The 'point density' method varies the per-node qp according to the relative number of points in each node. The sparsest 5% of nodes use $sliceQp + qpPot$, the densest 40% of nodes use $sliceQp - qpPot$, and the remaining nodes use $sliceQp$, where qpPot is `8 >> positionQpMultiplierLog2`.

--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 0 results in a scale factor of 1.

--positionQpMultiplierLog2=0|1|2|3

Controls the granularity of quantisation step sizes by limiting the number of QP values per step size doubling interval. There are 2^n QPs per step size doubling interval.

--positionIdcmQp=INT-VALUE

The quantisation parameter used to quantise directly coded (IDCM) point positions prior to reaching the `positionQuantisationOctreeDepth`.

--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.

--positionQuantisationSizeLog2=INT-VALUE

Sets the depth at which per-node QP offsets are signalled. The depth is the tree level with the configured node size. This value, if greater than 0, overrides `positionQuantisationOctreeDepth`.

When non-cubic nodes are present, the depth is the tree level with the minimum node size dimension.

qtbtEnabled=0|1

Enables non-cubic geometry tree coding. When enabled, the geometry tree may have a cuboid bounding box. The partitioning of internal tree nodes at a particular depth are determined non-normatively by the encoder to be one of octree, quadtree or binary partitions.

`maxNumQtBtBeforeOt=INT-VALUE`

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

`minQtbtSizeLog2=INT-VALUE`

Specifies the minimum size of 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.

`--neighbourAvailBoundaryLog2=INT-VALUE`

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

A value less than 2 limits the use of neighbouring nodes to direct octree siblings.

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

`--inferredDirectCodingMode=0|1|2|3`

Controls the degree to which early termination of the geometry octree is used to code isolated points.

Value	Extent of qualifying node criteria
0	Disabled
1	Fully isolated parent and child
2	Partially isolated parent
3	Unconstrained

`--jointTwoPointIdcm=0|1`

Controls the method used to code the point positions of directly coded nodes containing two distinct points. When enabled, an implicit point order is used to improve coding efficiency.

`--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--32`

Controls the frequency in 1/32 percent of IDCM eligibility. Set to zero, IDCM is disabled. Set to 32, IDCM is unconstrained.

`--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.

`--angularEnabled=0|1`

Controls the use of the angular coding mode in geometry occupancy contextualisation. Angular coding mode uses a LiDaR head model prior to improve the compression of a LiDaR acquired point cloud. The angular mode requires that planar mode is enabled.

`--lidarHeadPosition=x,y,z`

Specifies the LiDaR head position for use by the angular mode in terms of the input (unquantised) point cloud co-ordinate system.

`--numLasers=0--255`

The number of known laser angles and positions for use in angular mode.

`--lasersTheta=FLOAT-VALUE-LIST`

The elevation angle, theta, of each known laser used by the angular mode. Each elevation angle is expressed in radians relative to the x-y plane (range: $[-\pi, \pi]$. The zero angle describes a horizontal laser. Positive angles represent an elevation above the horizontal. Negative angles represent an elevation below the horizontal.

The software will convert the floating point angles to an 18 bit fixed point representation.

`--lasersZ=FLOAT-VALUE-LIST`

The vertical offset of each known laser used by the angular mode. Each offset is expressed along the z axis in the input point cloud co-ordinate system, corresponding to a vertical offset relative to the LiDAR head position.

The software will convert the floating point offsets to a three bit fixed point representation.

`--lasersNumPhiPerTurn=INT-VALUE-LIST`

The maximum number of samples that can be acquired during a full rotation of each known laser used by the angular mode.

`--planarBufferDisabled=0|1`

Controls the deactivation of the planar mode buffer for angular mode.

`--numOctreeEntropyStreams=INT-VALUE`

The number of geometry sub-streams (suitable for parallel coding) used to encode the geometry octree. For example, a value of eight generates eight sub-streams, one for the initial tree, then one for each of the last seven tree levels.

No parallel sub-streams are generated when *VALUE* is 1.

NB: the reference software does not attempt to exploit any opportunities for parallelism generated by this feature.

`--trisoupNodeSizeLog2=INT-VALUE|INT-VALUE-LIST`

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 less than 2 disables the use of trisoup.

When a list of values is used, the n-th entry in the list controls the configuration of the n-th slice. The last entry is mapped to all remaining slices.

--trisoup_sampling_value=INT-VALUE

Controls the number of points generated by the trisoup triangle voxelisation process. Larger values reduce the number of points generated per triangle.

Value	Description
0	automatic (default)
1	no sub-sampling
N >=1	point sampling period

The automatic mode will find the smallest sampling value that such that the number of generated points does not exceed the slice limit set by `sliceMaxPoints`.

--predGeomSort=INT-VALUE

Point order used to construct predictive geometry trees. Requires `geomTreeType=1`.

Value	Description
0	none
1	morton order
2	azimuth angle
3	radial distance
4	source azimuth angle (ply: laserangle)

--predGeomAzimuthSortPrecision=INT-VALUE

Controls the precision used in azimuthal sorting of points prior to predictive tree construction. A value of 0 represents full-precision, otherwise larger values represent increasing precision. Requires `predGeomSort=2`.

--predGeomTreePtsMax=INT-VALUE

Maximum number of points per predictive geometry tree. A slice may contain more than one predictive geometry tree. Requires `geomTreeType=1`.

--positionBaseQpFreqLog2=INT-VALUE

Controls the number of predictive geometry tree nodes scaled by the same QP offset instance. QP offsets are signalled every 2^n nodes in tree traversal order. This configuration applies to all slices. Requires `positionQuantisationEnabled=1`.

--positionSliceQpFreqLog2=INT-VALUE

Identical to `positionBaseQpFreqLog2`, but controls per-slice configuration.

--positionAzimuthScaleLog2=INT-VALUE

Number of additional bits used to represent predictive geometry azimuth angles. Requires `angularEnabled=1`.

--positionRadiusInvScaleLog2=INT-VALUE

Degree of quantisation applied in the representation of angular predictive geometry radial distances. Requires `angularEnabled=1`.

--positionAzimuthSpeed=INT-VALUE

Step size used to linearly model progression of per-laser azimuthal angles during angular predictive geometry coding. Requires `angularEnabled=1`.

`--predGeomAzimuthQuantization=1|0`

Controls the use of radius dependent azimuth quantization in predictive geometry coding. Requires `angularEnabled=1` and `geomTreeType=1`.

`--pointCountMetadata=0|1`

Controls the addition of per octree layer point count metadata to each geometry slice.

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.

`--defaultValue=INT-VALUE-LIST`

The default value to use for attribute data in case of data loss. If unset, the implicit default attribute value is 2**(bitdepth-1).

`--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.

`--attrScale=INT-VALUE` and `--attrOffset=INT-VALUE`

Scale and offset used to interpret coded attribute values.

The encoder derives the coded attribute value as $(attr - offset)/scale$.

The encoder and decoder scale coded attributes for output as $attr{\times}scale + offset$.

NB: these parameters are only supported for reflectance attributes.

`--transformType=0|1|2`

Coding method to use for the current attribute:

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

`--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.

`--rahtPredictionThreshold0=0--19`

Controls a per-block threshold used to enable or disable the transform domain prediction of RAHT coefficients. This threshold specifies the number of parent neighbour points that must be present.

`--rahtPredictionThreshold1=0--19`

Controls a per-block threshold used to enable or disable the transform domain prediction of RAHT coefficients. This threshold specifies the number of neighbour points that must be present.

`--numberOfNearestNeighboursInPrediction=INT-VALUE`

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

`--adaptivePredictionThreshold=INT-VALUE`

Neighbouring attribute value difference that enables the use of direct predictor selection over the weighted average. If bitdepth is greater than 8, the threshold is scaled by 2**(bitDepth - 8).

Applies to `transformType=0` only.

`--predWeightBlending=0|1`

When enabled, blends the distance derived weights of the three-neighbour predictor according to the relative distances between the neighbours.

Applies to `transformType=0` only.

`--direct_avg_predictor_disabled_flag=0|1`

Controls the use of the neighbour average predictor when direct prediction is invoked.

`--interComponentPredictionEnabled=0|1`

Controls the use of an in-loop inter-component prediction of attribute residuals. When enabled, the secondary attribute residuals (e.g. red/blue) are predicted from the primary component (e.g. green).

Applies to `transformType=0` and `attribute=color` only.

`--lastComponentPredictionEnabled=0|1`

Controls the use of an in-loop inter-component prediction of attribute coefficients. When enabled, the coefficient of the last component (e.g. Cr) of the secondary attribute is predicted from the corresponding first component (e.g. Cb) according to a simple model.

Applies to `transformType=2` and `attribute=color` only.

`--intraLoDSearchRange=INT-VALUE`

Buffer range searched for nearest neighbours within the same level of detail. The value -1 configures a full-range search.

`--interLoDSearchRange=INT-VALUE`

Buffer range searched for nearest neighbours between different levels of detail. The value -1 configures a full-range search.

`--max_num_direct_predictors=INT-VALUE`

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

`--lodDecimator=0|1|2`

Controls the level-of-detail generation method:

Value	Description
0	No decimation is performed
1	Decimation by periodic lodSubsamplingPeriod
1	Decimation by distance to lodSubsamplingPeriod centroid

`--intraLodPredictionSkipLayers=INT-VALUE`

The number of detail levels where intra prediction is disabled, starting from the finest detail level. Applies to `transformType=0` only.

Value	Description
-1	Disabled in all detail levels
0	Enabled in all detail levels
n	Disabled in n finest detail levels

`--aps_scalable_enabled_flag=0|1`

Enable spatially scalable attribute encoding. The option is only valid when `transformType=2`, `positionQpMultiplier` `lodDecimator=0`, and `trisoupNodeSizeLog2=0`.

`--max_neigh_range=INT-VALUE`

Limits the distance between a point and the neighbours used for its prediction. The maximum distance is expressed in units of node diagonals and is scaled according to the current level of detail.

`--levelOfDetailCount=INT-VALUE`

Attribute's number of levels of detail.

`--dist2=INT-VALUE`

An initial squared distances used to generate successive levels of detail. When equal to zero, an initial value is automatically determined.

`--dist2PercentileEstimate=FLOAT-VALUE`

Percentile of per-point nearest neighbour distances used to estimate `dist2`.

`--positionQuantizationScaleAdjustsDist2=0|1`

Adjusts `dist2` according to `sequenceScale`. 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 `sequenceScale` squared, thereby allowing `dist2` to be specified as an intrinsic property of the source sequence.

`--lodSubsamplingPeriod=INT-VALUE|INT-VALUE-LIST`

A list of sampling periods used to generate successive levels of detail.

`--canonical_point_order_flag=0|1`

Controls the order used for attribute coding. The canonical (geometry decoding order) is usable only with LoD attribute coding and `levelOfDtailCount=0`.

Value	Description
0	Morton order
1	Decoded geometry (canonical) order

`--spherical_coord_flag=0|1`

Controls the conversion of point co-ordinates used in attribute coding from the Cartesian domain to a scaled spherical domain.

`--attrSphericalMaxLog2=INT-VALUE`

Override spherical co-ordinate normalization factor. This may be used to compensate any increased azimuth resolution when `predGeomAzimuthQuantization=1`.

Value	Description
0	Automatic calculation
1	Override max value

Applies when `angularEnabled=1` and `predGeomAzimuthQuantization=1`.

`--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`.

`--quantNeighWeight=INT-VALUE-LIST`

Three factors used to derive quantization weights when `transformType=1`. The quantization weights are determined by recursively distributing each coefficient's weight to each of its neighbours, i, scaled by $\texttt{quantNeighWeight}[i] \div 256$.

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.

`--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.