International Standard

ISO/IEC 23090-22

First edition
2024-09

Information technology — Coded representation of immersive media —

Part 22: Conformance for G-PCC

Technologies de l'information — Représentation codée de média immersifs —

Partie 22: Conformité pour G-PCC

Reference number
ISO/IEC 23090-22:2024(en)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

ISO and IEC draw attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO and IEC take no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO and IEC had not received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents and https://patents.iec.ch. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology* Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

A list of all parts in the ISO/IEC 23090 series series can be found on the ISO and IEC websites.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

Introduction

Advance in 3D capturing and rendering technologies is enabling new applications and services in the field of assisted and autonomous driving, maps, cultural heritage, industrial processes, immersive real-time communication, and Virtual/Augmented/Mixed reality (VR/AR/MR) content creation, transmission and communication. Point clouds have arisen as one of the main representations for such applications. A point cloud frame consists of a set of 3D points. Each point, in addition to having a 3D position may also be associated with numerous other attributes such as colour, transparency, reflectance, timestamp, surface normal, and classification. Such representations require a large amount of data, which can be costly in terms of storage and transmission. Therefore, ISO/IEC 23090-9 specifies Geometry-based Point Cloud Compression (G-PCC), which aims at efficiently compressing point cloud representations.

This document is the conformance testing specification for ISO/IEC 23090-9.

Information technology — Coded representation of immersive media —

Part 22: Conformance for G-PCC

1 Scope

This document specifies a set of tests and procedures designed to indicate whether encoders or decoders meet the normative requirements specified in ISO/IEC 23090-9.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 23090-9, *Information technology — Coded representation of immersive media — Part 9: Geometry-based point cloud compression*

ISO/IEC 23090-21, *Information technology — Coded representation of immersive media — Part 21: Reference software for Geometry-based Point Cloud Compression (G-PCC)*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 23090-9 and the following apply.

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

3.1
bitstream
sequence of bits

Note 1 to entry: An ISO/IEC 23090-9 G-PCC bitstream.

3.2
decoding process
process that restores information from a coded representation to the original form according to a given code

Note 1 to entry: An embodiment of the decoding process specified by ISO/IEC 23090-9.

3.3
decoder
embodiment of a *decoding process* (3.2)

Note 1 to entry: An ISO/IEC 23090-9 G-PCC decoder. The decoder does not include the rendering and display process, which is outside the scope of this document.

3.4
encoder
embodiment of a process that produces a *bitstream* (3.1)

3.5
reference software decoder
particular *decoder* (3.3) provided as a software package for use as an example available for study, as a potential starting basis for the development of other decoder, as a way of testing bitstreams for conformance to a decoding process specification, or as a reference for comparison with the behaviour of other decoder

Note 1 to entry: A software decoder is provided in ISO/IEC 23090-21.

3.6
reference software encoder
particular *encoder* (3.4) provided as a software package for use as an example available for study, as a potential starting basis for the development of other encoder, or as a reference for comparison with the behaviour of other encoder

Note 1 to entry: A software encoder is provided in ISO/IEC 23090-21.

4 Abbreviated terms

For the purposes of this document, relevant abbreviated terms are specified in ISO/IEC 23090-9:2023, Clause 4.

5 Conventions

For the purposes of this document, relevant conventions are specified in ISO/IEC 23090-9:2023, Clause 5.

6 Conformance testing for ISO/IEC 23090-9

6.1 General

The following subclauses specify the tests for verifying conformance of bitstreams as well as decoders. These tests shall use the test data (bitstream test suites) provided in https://standards.iso.org/iso-iec/23090/-22/ed-1/en and the reference software decoder is specified in Rec. ISO/IEC 23090-21.

6.2 Bitstream conformance

The bitstream conformance follows the specification text in ISO/IEC 23090-9. See Annex A.

6.3 Decoder conformance

The decoder conformance for ISO/IEC 23090-9, such as conformance point, is specified in ISO/IEC 23090-9:2023, 6.5.1.

6.4 Procedure to test bitstreams

A bitstream that claims conformance with ISO/IEC 23090-9 shall pass the following test.

The bitstream shall be decoded by processing it with the reference software decoder. When processed by the reference software decoder, the bitstream shall not cause any error or non-conformance messages to be reported by the reference software decoder. This test should not be applied to bitstreams that are known to contain errors introduced by transmission, as such errors are highly likely to result in bitstreams that lack conformance to ISO/IEC 23090-9.

Successfully passing the reference software decoder test provides only a strong presumption that the bitstream under test does indeed meet all the requirements (except Annex C) specified in ISO/IEC 23090-9 that are tested by the reference software decoder.

ISO/IEC 23090-9 contains several informative recommendations that are not an integral part of that International Standard. When testing a bitstream for conformance, it can also be useful to test whether or not the bitstream follows those recommendations.

To check correctness of a bitstream, it is necessary to parse the entire bitstream and to extract all the syntax elements and other values derived from those syntactic elements and used by the decoding process specified in ISO/IEC 23090-9.

A verifier may not necessarily perform all stages of the decoding process specified in ISO/IEC 23090-9 in order to verify bitstream correctness. Many tests can be performed on syntax elements in a state prior to their use in some processing stages.

6.5 Procedure to test decoder conformance

6.5.1 Conformance bitstreams

A bitstream has values of main_profile_compatibility_flag and level_idc corresponding to a set of specified constraints on a bitstream for which a decoder conforming to a specified profile, and level is required in ISO/IEC 23090-9:2023, Annex A to properly perform the decoding process.

6.5.2 Contents of the bitstream file

The conformance bitstreams are included in this document as an electronic attachment. The following information is included in a single folder for each such bitstream.

— *.bit – bitstream as described in subclause 6.6.2 (mandatory)

— readme.md – description (mandatory)

— *.cfg – config file used to generate bitstream with TMC13 encoder SW (optional, not applicable if TMC13 encoder release version not used)

— *.md5 – MD5sum of the bitstream file (mandatory)

— *_dec.ply – unordered decoded point cloud frames (optional).

— *_dec.ply.md5 – MD5 checksum for *_dec.ply (optional)

— Makefile – Script to regenerate the bitstream (optional)

NOTE Reference software decoder can be used to generate _*dec.ply file

6.5.3 Requirements on output of the decoding process

The output of the decoding process is specified in ISO/IEC 23090-9:2023, Clause 8.

A decoder shall be configured to output integer point position in accordance with ISO/IEC 23090-9:2023, subclause 6.5.1 so that that the decoder output data in conformance testing mode.

The rendering process, which may follow the output of the decoding process, is outside the scope of this document.

6.5.4 Recommendations

In addition to the requirements, it is desirable that conforming decoders implement various informative recommendations specified in ISO/IEC 23090-9 that are not an integral part of that International Standard. This clause discusses some of these recommendations.

It is recommended that a conforming decoder be able to resume the decoding process as soon as possible after the loss or corruption of part of a bitstream.

6.6 Specification of the test bitstreams

6.6.1 General

The bitstreams used for the decoder conformance testing specified in this document shall be those listed in Table 1. Characteristics of each bitstream are specified in this subclause.

Table 1 — List of reference bitstreams.

Categories	Feature Name	Features tested
Common functionality	EBS	Entropy bypass stream
	EC	Entropy continuation
	ST	Slice/Tile
	GPS	GPS (Geometry Parameter Set)
	APS	APS (Attribute Parameter Set)
Geometry coding	GTT	Geom tree coding type
	GSO	Geometry scaling in Occupancy tree
	OG	Occupancy tree geometry
	DPO	Duplicate points in Occupancy tree
	IDCM	IDCM
	NAV	Neighbour availability volume (N3/N6)
	QTBT	QTBT
	IOC	Intra occupancy contexts
	ACN	Adjacent child neighbours
	PLANAR	Planar
	AngIDCM	Angular-IDCM
	AngPLANAR	Angular-Planar
	PredGEO	Predictive geometry
	PredDUP	Predictive duplicate
	PredANG	Predictive geometry: Angular mode
	PredGS	Predictive geometry: Geometry Scaling
	CAPS	Change APS

Table 1 *(continued)*

Categories	Feature Name	Features tested
Attribute coding	GATT	Generic attribute
	QuantATT	Quantization
	LodRAHT	LoD and RAHT
	SLOD	Single LoD
	NumLOD	Number of LoDs
	DecLOD	LoD method: Decimation
	DisLOD	LoD method: Distance
	CentLOD	LoD method: Block-based
	SL	Scalable Lifting
	PSInterLOD	Predictor search: inter LoD
	PSIntraLOD	Predictor search: intra LoD
	NumPRED	Number of predictors
	DirectPRED	Predicting Transform: Direct predictors
	ACO	Axis coding order
	NB	Neighbour bias
	LCP	Lifting Transform: Last component pred
	ICP	Predicting Transform: Inter component pred
	RahtPRED	RAHT: prediction
	SDC	Spherical domain coding
	ANBF	Attribute neighbour blending filtering

6.6.2 Test bitstreams

6.6.2.1 Entropy bypass stream (EBS)

6.6.2.1.1 Test bitstream EBS_A_Panasonic[1)]

Specification: The bitstream exercise the entropy bypass stream On functionality.

— bypass_stream_enabled equal to 1.

— attr_coding_type is equal to 0 (RAHT).

Functional stage: Entropy bypass stream enabled and RAHT Attribute Coding on Simple profile.

Purpose: Check that the decoder can properly decode bitstreams in which entropy stream enabled.

6.6.2.1.2 Test bitstream EBS_B_Panasonic

Specification: The bitstream exercise the entropy bypass stream On functionality.

— bypass_stream_enabled equal to 1.

— attr_coding_type is equal to 1 (LoD with Predicting Transform).

Functional stage: Entropy bypass stream enabled and Predicting Attribute Coding on Predictive profile.

Purpose: Check that the decoder can properly decode bitstreams in which entropy stream enabled.

1) This information is given for the convenience of users of this document and does not constitute an endorsement by ISO or IEC of the product named.

6.6.2.1.3 Test bitstream EBS_C_Panasonic

Specification: The bitstream exercise the entropy bypass stream Off functionality.

— bypass_stream_enabled equal to 0.

— attr_coding_type is equal to 1 (LoD with Predicting Transform).

Functional stage: Entropy bypass stream disabled and Predicting Attribute Coding on Dense profile.

Purpose: Check that the decoder can properly decode bitstreams in which entropy stream disabled.

6.6.2.1.4 Test bitstream EBS_D_Panasonic

Specification: The bitstream exercise the entropy bypass stream On functionality.

— bypass_stream_enabled equal to 1.

— attr_coding_type is equal to 1 (LoD with Predicting Transform).

Functional stage: Entropy bypass stream enabled and Predicting Attribute Coding on Dense profile.

Purpose: Check that the decoder can properly decode bitstreams in which entropy stream enabled.

6.6.2.1.5 Test bitstream EBS_E_Panasonic

Specification: The bitstream exercise the entropy bypass stream On functionality.

— bypass_stream_enabled equal to 1.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

Functional stage: Entropy bypass stream enabled and Lifting Attribute Coding on Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which entropy stream enabled.

6.6.2.2 Entropy continuation (EC)

6.6.2.2.1 Test bitstream EC_A_Panasonic

Specification: The bitstream exercise the entropy continuation functionality on each coding method.

— slice_reordering_constraint is equal to 1.

— entropy_continuation_enabled is equal to 1.

— slice_entropy_continuation is used.

— geom_tree_type is equal to 0 (occupancy tree).

— octree_bitwise_coding is equal to 1.

— attr_coding_type is equal to 0 (RAHT).

Functional stage: Entropy continuation on Occupancy tree Geometry Coding and RAHT Attribute Coding on Dense profile and Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which entropy continuation function is on each coding method.

6.6.2.2.2 Test bitstream EC_B_Panasonic

Specification: The bitstream exercise the entropy continuation functionality on each coding method.

— slice_reordering_constraint is equal to 1.

— entropy_continuation_enabled is equal to 1.

— slice_entropy_continuation is used.

— geom_tree_type is equal to 0 (occupancy tree).

— occtree_bitwise_coding is equal to 0 (using the dictionary encoding).

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

Functional stage: Entropy continuation on Occupancy tree Geometry Coding with dictionary encoding and Lifting Attribute Coding on Simple profile.

Purpose: Check that the decoder can properly decode bitstreams in which entropy continuation function is on each coding method.

6.6.2.2.3 Test bitstream EC_C_Panasonic

Specification: The bitstream exercise the entropy continuation functionality on each coding method.

— slice_reordering_constraint is equal to 1.

— entropy_continuation_enabled is equal to 1.

— slice_entropy_continuation is used.

— geom_tree_type is equal to 1 (geometry_predtree).

— attr_coding_type is equal to 1 (LoD with Predicting Transform).

Functional stage: Predictive Geometry Coding and LoD with Predicting Transform Attribute Coding on Predictive profile.

Purpose: Check that the decoder can properly decode bitstreams in which entropy continuation function is on each coding method.

6.6.2.2.4 Test bitstream EC_D_Panasonic

Specification: The bitstream exercise the entropy continuation functionality on each coding method.

— slice_reordering_constraint is equal to 1.

— entropy_continuation_enabled is equal to 0, 1, 1, 0, 1.

— slice_entropy_continuation is used.

Functional stage: Predictive Geometry Coding and LoD with Predicting Transform Attribute Coding on Predictive profile.

Purpose: Check that the decoder can properly decode bitstreams in which entropy continuation function is on each coding method.

6.6.2.3 Slice/Tile (ST)

6.6.2.3.1 Test bitstream ST_A_LGE

Specification: The bitstream consists of zero tile and one slice.

— Number of tiles is equal to 0.

— Number of slices is equal to 1.

Functional stage: Common functionality in Simple profile.

Purpose: Check that the decoder can properly decode bitstreams in which zero tile and one slice has.

6.6.2.3.2 Test bitstream ST_B_LGE

Specification: The bitstream consists of zero tile and one slice.

— Number of tiles is equal to 0.

— Number of slices is equal to 1.

Functional stage: Common functionality in Predictive profile.

Purpose: Check that the decoder can properly decode bitstreams in which zero tile and one slice has.

6.6.2.3.3 Test bitstream ST_C_LGE

Specification: The bitstream consists of zero tile and one slice.

— Number of tiles is equal to 0.

— Number of slices is equal to 1.

Functional stage: Common functionality in Main profile and Dense profile.

Purpose: Check that the decoder can properly decode bitstreams in which zero tile and one slice.

6.6.2.3.4 Test bitstream ST_D_LGE

Specification: The bitstream consists of zero tile and multiple slices.

— Number of tiles is equal to 0.

— Number of slices is equal to 4.

Functional stage: Common functionality in Simple profile.

Purpose: Check that the decoder can properly decode bitstreams in which zero tile and multiple slices have.

6.6.2.3.5 Test bitstream ST_E_LGE

Specification: The bitstream consists of zero tile and multiple slices.

— Number of tiles is equal to 0.

— Number of slices is equal to 4.

Functional stage: Common functionality in Predictive profile.

Purpose: Check that the decoder can properly decode bitstreams in which zero tile and multiple slices have.

6.6.2.3.6 Test bitstream ST_F_LGE

Specification: The bitstream consists of zero tile and multiple slices.

— Number of tiles is equal to 0.

— Number of slices is equal to 4.

Functional stage: Common functionality in Main profile and Dense profile.

Purpose: Check that the decoder can properly decode bitstreams in which zero tile and multiple slices have.

6.6.2.3.7 Test bitstream ST_G_LGE

Specification: The bitstream consists of one tile and one slice.

— Number of tiles is equal to 1.

— Number of slices is equal to 1.

Functional stage: Common functionality in Simple profile.

Purpose: Check that the decoder can properly decode bitstreams in which one tile and one slice has.

6.6.2.3.8 Test bitstream ST_H_LGE

Specification: The bitstream consists of one tile and one slice.

— Number of tiles is equal to 1.

— Number of slices is equal to 1.

Functional stage: Common functionality in Predictive profile.

Purpose: Check that the decoder can properly decode bitstreams in which one tile and one slice has.

6.6.2.3.9 Test bitstream ST_I_LGE

Specification: The bitstream consists of one tile and one slice.

— Number of tiles is equal to 1.

— Number of slices is equal to 1.

Functional stage: Common functionality in Main profile and Dense profile.

Purpose: Check that the decoder can properly decode bitstreams in which one tile and one slice has.

6.6.2.3.10 Test bitstream ST_J_LGE

Specification: The bitstream consists of one tile and multiple slices.

— Number of tiles is equal to 1.

— Number of slices is equal to 4.

Functional stage: Common functionality in Simple profile.

Purpose: Check that the decoder can properly decode bitstreams in which one tile and multiple slices have.

6.6.2.3.11 Test bitstream ST_K_LGE

Specification: The bitstream consists of one tile and multiple slices.

— Number of tiles is equal to 1.

— Number of slices is equal to 4.

Functional stage: Common functionality in Predictive profile.

Purpose: Check that the decoder can properly decode bitstreams in which one tile and multiple slices have.

6.6.2.3.12 Test bitstream ST_L_LGE

Specification: The bitstream consists of one tile and multiple slices.

— Number of tiles is equal to 1.

— Number of slices is equal to 4.

— Functional stage: Common functionality in Main profile and Dense profile.

Purpose: Check that the decoder can properly decode bitstreams in which one tile and multiple slices have.

6.6.2.3.13 Test bitstream ST_M_LGE

Specification: The bitstream consists of multiple tiles and multiple slices.

— Number of tiles is equal to 3.

— Number of slices is equal to 6.

— Tile inventory is signalled.

Functional stage: Common functionality in Simple profile.

Purpose: Check that the decoder can properly decode bitstreams in which multiple tiles and multiple slices have.

6.6.2.3.14 Test bitstream ST_N_LGE

Specification: The bitstream consists of multiple tiles and multiple slices.

— Number of tiles is equal to 3.

— Number of slices is equal to 6.

— Tile inventory is signalled.

Functional stage: Common functionality in Predictive profile.

Purpose: Check that the decoder can properly decode bitstreams in which multiple tiles and multiple slices have.

6.6.2.3.15 Test bitstream ST_O_LGE

Specification: The bitstream consists of multiple tiles and multiple slices.

— Number of tiles is equal to 3.

— Number of slices is equal to 6.

— Tile inventory is signalled.

Functional stage: Common functionality in Main profile and Dense profile.

Purpose: Check that the decoder can properly decode bitstreams in which multiple tiles and multiple slices have.

6.6.2.4 GPS (GPS)

6.6.2.4.1 Test bitstream GPS_A_Tencent[2)]

Specification: The bitstream includes one geometry parameter sets (GPS), which is sharing among multiple frames.

Functional stage: Common functionality in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which one GPS is sharing among multiple frames.

6.6.2.4.2 Test bitstream GPS_B_Tencent

Specification: The bitstream includes GPS. Difference slices in a frame uses alternative GPS.

— gps.geom_planar_mode_enabled_flag =0, 1 in two GPS, respectively.

Functional stage: Common functionality in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which slices in a frame employ different GPS.

6.6.2.4.3 Test bitstream GPS_C_Tencent

Specification: The bitstream includes one GPS, which repeats in front of each frame.

Functional stage: GPS in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which one GPS repeats in front of each frame

6.6.2.5 APS (APS)

6.6.2.5.1 Test bitstream APS_A_Tencent

Specification: The bitstream includes one APS, which is sharing among multiple frames.

Functional stage: Common functionality in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which one APS is sharing among multiple frames.

6.6.2.6 Geometry tree type (GTT)

6.6.2.6.1 Test bitstream GTT_A_Tencent

Specification: The bitstream includes two GPS, each with different geometry tree type. Different slices in a frame employ alternative GPS.

— gps.predgeom_enabled_flag = 0, 1.

Functional stage: Geometry coding in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which slices in a frame employ different geometry tree types.

---

2) This information is given for the convenience of users of this document and does not constitute an endorsement by ISO or IEC of the product named.

6.6.2.7 Geometry scaling in Occupancy tree (GSO)

6.6.2.7.1 Test bitstream GSO_A_Tencent

Specification: Geometry scaling in occupancy tree is turned on. Base QP, IDCM QP offset, slice QP offset, and node-level QP offset are all enabled. Note that the node-level QP offset is random in different nodes and node size is set to 9.

— geom_tree_type is equal to 0 (occupancy tree).

— geom_scaling_enabled equal to 1.

— geom_qp equal to 8.

— occtree_direct_node_qp_offset equal to -4.

— slice_ geom_qp_offset equal to 4.

— geom_qp_mul_log2 equal to 2.

Functional stage: geometry coding in Main profile.

Purpose: Check if the decoder can properly decode bitstreams in which a combination of base QP, IDCM QP offset, slice QP offset, and node QP offset is configured.

6.6.2.7.2 Test bitstream GSO_B_Tencent

Specification: Geometry scaling in occupancy tree is turned on. Only IDCM coded points are quantized and other points are not quantized.

— geom_tree_type is equal to 0 (occupancy tree).

— geom_scaling_enabled equal to 1.

— geom_qp equal to 0.

— occtree_direct_node_qp_offset equal to 8.

— slice_ geom_qp_offset equal to 0.

— geom_qp_mul_log2 equal to 2.

Functional stage: geometry coding in Main profile.

Purpose: Check if the decoder can properly decode bitstreams in which only IDCM QP offset is enabled.

6.6.2.7.3 Test bitstream GSO_C_Qualcomm[3)]

Specification: Geometry scaling in occupancy tree is turned on. The QP values used to derive the step sizes will be multiples of 2.

— geom_tree_type is equal to 0 (occupancy tree).

— geom_scaling_enabled is equal to 1.

— geom_qp is equal to 8.

— geom_qp_mul_log2 is equal to 1.

— occtree node size used for signaling position QP offsets is equal to 8.

3) This information is given for the convenience of users of this document and does not constitute an endorsement by ISO or IEC of the product named.

Functional stage: geometry coding in Simple profile, Main profile and Dense profile.

Purpose: Check that the decoder can properly decode bitstreams in which QP multiplier value of 2 is configured.

6.6.2.7.4 Test bitstream GSO_D_Qualcomm

Specification: Geometry scaling in occupancy tree is turned on. The QP values used to derive the step sizes will be multiples of 8.

— geom_tree_type is equal to 0 (occupancy tree).

— geom_scaling_enabled is equal to 1.

— geom_qp is equal to 0.

— geom_qp_mul_log2 is equal to 3.

— occtree node size used for signaling position QP offsets is equal to 4.

Functional stage: geometry coding in Simple profile, Main profile and Dense profile.

Purpose: Check that the decoder can properly decode bitstreams in which QP multiplier value of 8 is configured.

6.6.2.7.5 Test bitstream GSO_E_Qualcomm

Specification: Geometry scaling in occupancy tree is turned on. The QP values used to derive the step sizes will be multiples of 1.

— geom_tree_type is equal to 0 (occupancy tree).

— geom_scaling_enabled is equal to 1.

— geom_qp is equal to 9.

— geom_qp_mul_log2 is equal to 0.

— occtree node size used for signaling position QP offsets is equal to 7.

Functional stage: geometry coding in Simple profile, Main profile and Dense profile.

Purpose: Check that the decoder can properly decode bitstreams in which QP multiplier value of 1 is configured.

6.6.2.7.6 Test bitstream GSO_F_Qualcomm

Specification: Geometry scaling in occupancy tree is turned on. The QP values used to derive the step sizes will be multiples of 4.

— geom_tree_type is equal to 0 (occupancy tree).

— geom_scaling_enabled is equal to 1.

— geom_qp is equal to 0.

— geom_qp_mul_log2 is equal to 2.

— occtree node size used for signaling position QP offsets is equal to 9.

Functional stage: geometry coding in Simple profile, Main profile and Dense profile.

Purpose: Check that the decoder can properly decode bitstreams in which QP multiplier value of 4 is configured.

6.6.2.8 Occupancy tree geometry (OG)

6.6.2.8.1 Test bitstream OG_A_LGE

Specification: The bitstream tests Occupancy geometry coding with a point cloud which has one point.

— geom_tree_type is equal to 0 (occupancy tree).

Functional stage: Geometry coding in Simple profile.

Purpose: Check that the decoder can properly decode bitstreams which Occupancy geometry coding is enabled and includes one point.

6.6.2.8.2 Test bitstream OG_B_LGE

Specification: The bitstream tests Occupancy geometry coding with a point cloud which has one point.

— geom_tree_type is equal to 0 (occupancy tree).

Functional stage: Geometry coding in Main profile and Dense profile.

Purpose: Check that the decoder can properly decode bitstreams which Occupancy geometry coding is enabled and includes one point.

6.6.2.8.3 Test bitstream OG_C_LGE

Specification: The bitstream tests Occupancy geometry coding with a point cloud which has two points.

— geom_tree_type is equal to 0 (occupancy tree).

Functional stage: Geometry coding in Simple profile.

Purpose: Check that the decoder can properly decode bitstreams which Occupancy geometry coding is enabled and includes two points.

6.6.2.8.4 Test bitstream OG_D_LGE

Specification: The bitstream tests Occupancy geometry coding with a point cloud which has two points.

— geom_tree_type is equal to 0 (occupancy tree).

Functional stage: Geometry coding in Main profile and Dense profile.

Purpose: Check that the decoder can properly decode bitstreams which Occupancy geometry coding is enabled and includes two points.

6.6.2.8.5 Test bitstream OG_E_LGE

Specification: The bitstream tests Occupancy geometry coding with a point cloud which has 21 bits depth.

— geom_tree_type is equal to 0 (occupancy tree).

— level_idc is equal to 80.

Functional stage: Geometry coding in Simple profile.

Purpose: Check that the decoder can properly decode bitstreams which Occupancy geometry coding is enabled and uses 21 bits depth.

6.6.2.8.6 Test bitstream OG_F_LGE

Specification: The bitstream tests Occupancy geometry coding with a point cloud which has 21 bits depth.

— geom_tree_type is equal to 0 (occupancy tree).

— level_idc is equal to 80.

Functional stage: Geometry coding in Main profile and Dense profile.

Purpose: Check that the decoder can properly decode bitstreams which Occupancy geometry coding is enabled and uses 21 bits depth.

6.6.2.9 Duplicate points in occupancy tree (DPO)

6.6.2.9.1 Test bitstream DPO_A_Panasonic

Specification: The bitstream tests duplicate point in Occupancy Tree. The bitstream has no duplicate points in single slice, and there are no constraints on duplicated points in a point cloud frame.

— geom_tree_type is equal to 0 (occupancy tree).

— geom_dup_point_counts_enabled is equal to 1.

— unique_point_positions_constraint is equal to 0.

— Number of slices is equal to 2.

Functional stage: Geometry occupancy tree coding in Main profile

Purpose: Check that the decoder can properly decode bitstreams in which the duplicate points count is enabled and a point cloud has duplicated points.

6.6.2.9.2 Test bitstream DPO_B_Panasonic

Specification: The bitstream tests duplicate point in Occupancy Tree. The bitstream has duplicate points in single slice, and there are no constraints on duplicated points in a point cloud frame.

— geom_tree_type is equal to 0 (occupancy tree).

— geom_dup_point_counts_enabled is equal to 0.

— unique_point_positions_constraint is equal to 0.

— Number of slices is equal to 1.

Functional stage: Geometry occupancy tree coding in Main profile

Purpose: Check that the decoder can properly decode bitstreams in which the duplicate points count is disabled and a point cloud has duplicated points.

6.6.2.9.3 Test bitstream DPO_C_Panasonic

Specification: The bitstream tests duplicate point in Occupancy Tree. The bitstream has no duplicate points in single slice.

— geom_tree_type is equal to 0 (occupancy tree).

— geom_dup_point_counts_enabled is equal to 1.

— Number of slices is equal to 1.

Functional stage: Geometry occupancy tree coding in Main profile

Purpose: Check that the decoder can properly decode bitstreams in which the duplicate points count is enabled.

6.6.2.10 IDCM (IDCM)

6.6.2.10.1 Test bitstream IDCM_A_Sony[4)]

Specification: The bitstream tests when Inferred direct coding mode is turned off in occupancy tree geometry coding.

— geom_tree_type is equal to 0 (occupancy tree).

— occtree_direct_coding_mode is equal to 0 (OFF test).

Functional stage: geometry occupancy tree coding in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which occtree_direct_coding_mode is configured.

6.6.2.10.2 Test bitstream IDCM_B_Sony

Specification: The bitstream tests when Inferred direct coding mode is turned on and the joint coding of two points is turned off occupancy tree geometry coding.

— geom_tree_type is equal to 0 (occupancy tree).

— occtree_direct_coding_mode is equal to 1.

— occtree_direct_joint_coding_enabled is equal to 0.

Functional stage: geometry occupancy tree coding in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which occtree_direct_coding_mode and is occtree_direct_joint_coding_enabled is configured.

6.6.2.10.3 Test bitstream IDCM_C_Sony

Specification: The bitstream tests when Inferred direct coding mode is turned on and the joint coding of two points is turned on occupancy tree geometry coding.

— geom_tree_type is equal to 0 (occupancy tree).

— occtree_direct_coding_mode is equal to 1.

— occtree_direct_joint_coding_enabled is equal to 1.

Functional stage: geometry occupancy tree coding in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which occtree_direct_coding_mode and is occtree_direct_joint_coding_enabled is configured.

6.6.2.10.4 Test bitstream IDCM_D_Sony

Specification: The bitstream tests when Inferred direct coding mode is turned on in occupancy tree geometry coding.

— geom_tree_type is equal to 0 (occupancy tree).

— occtree_direct_coding_mode is equal to 2.

4) This information is given for the convenience of users of this document and does not constitute an endorsement by ISO or IEC of the product named.

Functional stage: geometry occupancy tree coding in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which occtree_direct_coding_mode is configured.

6.6.2.10.5 Test bitstream IDCM_E_Sony

Specification: The bitstream tests when Inferred direct coding mode is turned on in occupancy tree geometry coding.

— geom_tree_type is equal to 0 (occupancy tree).

— occtree_direct_coding_mode is equal to 3.

Functional stage: geometry occupancy tree coding in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which occtree_direct_coding_mode is configured.

6.6.2.11 Neighbour availability volume (NAV)

6.6.2.11.1 Test bitstream NAV_A_Panasonic

Specification: The bitstream exercise occtree_neigh_window_log2_minus1.

— geom_tree_type is equal to 0 (occupancy tree).

— occtree_neigh_window_log2_minus1 equal to 0.

Functional stage: Occupancy tree Geometry Coding in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which geometry node occupancy of the current node is coded with the contexts determined from neighbouring nodes which is located inside the parent node of the current node.

6.6.2.11.2 Test bitstream NAV_B_Panasonic

Specification: The bitstream exercise the range of occtree_neigh_window_log2_minus1.

— geom_tree_type is equal to 0 (occupancy tree).

— occtree_neigh_window_log2_minus1 equal to 1.

Functional stage: Occupancy tree Geometry Coding in Dense profile and Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which geometry node occupancy of the current node is coded with the contexts determined from neighbouring nodes which is located inside or outside the parent node of the current node.

6.6.2.11.3 Test bitstream NAV_C_Panasonic

Specification: The bitstream exercise the range of occtree_neigh_window_log2_minus1.

— geom_tree_type is equal to 0 (occupancy tree).

— occtree_neigh_window_log2_minus1 equal to 2.

Functional stage: Occupancy tree Geometry Coding in Dense profile and Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which geometry node occupancy of the current node is coded with the contexts determined from neighbouring nodes which is located inside or outside the parent node of the current node.

6.6.2.11.4 Test bitstream NAV_D_Panasonic

Specification: The bitstream exercise the range of occtree_neigh_window_log2_minus1.

— geom_tree_type is equal to 0 (occupancy tree).

— occtree_neigh_window_log2_minus1 equal to 3.

Functional stage: Occupancy tree Geometry Coding in Dense profile and Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which geometry node occupancy of the current node is coded with the contexts determined from neighbouring nodes which is located inside or outside the parent node of the current node.

6.6.2.11.5 Test bitstream NAV_E_Panasonic

Specification: The bitstream exercise the range of occtree_neigh_window_log2_minus1.

— geom_tree_type is equal to 0 (occupancy tree).

— occtree_neigh_window_log2_minus1 equal to 4.

Functional stage: Occupancy tree Geometry Coding in Dense profile and Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which geometry node occupancy of the current node is coded with the contexts determined from neighbouring nodes which is located inside or outside the parent node of the current node.

6.6.2.11.6 Test bitstream NAV_F_Panasonic

Specification: The bitstream exercise the range of occtree_neigh_window_log2_minus1.

— geom_tree_type is equal to 0 (occupancy tree).

— occtree_neigh_window_log2_minus1 equal to 5.

Functional stage: Occupancy tree Geometry Coding in Dense profile and Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which geometry node occupancy of the current node is coded with the contexts determined from neighbouring nodes which is located inside or outside the parent node of the current node.

6.6.2.11.7 Test bitstream NAV_G_Panasonic

Specification: The bitstream exercise the range of occtree_neigh_window_log2_minus1.

— geom_tree_type is equal to 0 (occupancy tree).

— occtree_neigh_window_log2_minus1 equal to 6.

Functional stage: Occupancy tree Geometry Coding in Dense profile and Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which geometry node occupancy of the current node is coded with the contexts determined from neighbouring nodes which is located inside or outside the parent node of the current node.

6.6.2.11.8 Test bitstream NAV_H_Panasonic

Specification: The bitstream exercise the range of occtree_neigh_window_log2_minus1.

— geom_tree_type is equal to 0 (occupancy tree).

— occtree_neigh_window_log2_minus1 equal to 7.

Functional stage: Occupancy tree Geometry Coding in Dense profile and Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which geometry node occupancy of the current node is coded with the contexts determined from neighbouring nodes which is located inside or outside the parent node of the current node.

6.6.2.12 QTBT (QTBT)

6.6.2.12.1 Test bitstream QTBT_A_Tencent

Specification: Both QTBT and planar mode are enabled.

— geom_tree_type is equal to 0 (occupancy tree).

— occtree_coded_axis_list_present equal to 1.

— occtree_planar_enabled equal to 1.

— occtree_coded_axis carries an array of 3-bit info about tree partition at each depth.

Functional stage: Geometry coding in Main profile.

Purpose: Check if the decoder can properly decode bitstreams in which both QTBT and planar mode are enabled.

6.6.2.12.2 Test bitstream QTBT_B_Tencent

Specification: Both QTBT and geometry scaling in occupancy tree are enabled.

— geom_tree_type is equal to 0 (occupancy tree).

— occtree_coded_axis_list_present equal to 1.

— occtree_planar_enabled equal to 1.

— occtree_coded_axis carries an array of 3-bit info about tree partition at each depth.

— geom_scaling_enabled equal to 1.

— geom_qp equal to 8.

— occtree_direct_node_qp_offset equal to 0.

— slice_geom_qp_offset equal to 0.

— geom_qp_mul_log2 equal to 2.

Functional stage: Geometry coding in Main profile.

Purpose: Check if the decoder can properly decode bitstreams in which both QTBT and geometry scaling in occupancy tree are enabled.

6.6.2.13 Intra occupancy contexts (IOC)

6.6.2.13.1 Test bitstream IOC_A_NPU

Specification: Turn off intra occupancy contexts when QTBT is on.

— geom_tree_type is equal to 0 (occupancy tree).

— occtree_intra_pred_max_nodesize_log2 is equal to 0.

— occtree_coded_axis_list_present is equal to 1.

Functional stage: Geometry coding in Main profile.

Purpose: Check if the decoder can properly decode bitstreams in which both IOC is disabled while QTBT is enabled.

6.6.2.13.2 Test bitstream IOC_B_NPU

Specification: Enable intra occupancy contexts when QTBT is on.

— geom_tree_type is equal to 0 (occupancy tree).

— octree_intra_pred_max_nodesize_log2 is equal to 6.

— octree_coded_axis_list_present is equal to 1.

Functional stage: Geometry coding in Main profile.

Purpose: Check if the decoder can properly decode bitstreams in which both intra occupancy contexts and QTBT are enabled.

6.6.2.13.3 Test bitstream IOC_C_NPU

Specification: Turn off intra occupancy contexts when QTBT is on.

— geom_tree_type is equal to 0 (occupancy tree).

— octree_intra_pred_max_nodesize_log2 is equal to 0.

— octree_coded_axis_list_present is equal to 1.

Functional stage: Geometry coding in Dense profile.

Purpose: Check if the decoder can properly decode bitstreams in which both IOC is disabled while QTBT is enabled.

6.6.2.13.4 Test bitstream IOC_D_NPU

Specification: Enable intra occupancy contexts when QTBT is on.

— geom_tree_type is equal to 0 (occupancy tree).

— octree_intra_pred_max_nodesize_log2 is equal to 6.

— octree_coded_axis_list_present is equal to 1.

Functional stage: Geometry coding in Dense profile.

Purpose: Check if the decoder can properly decode bitstreams in which both intra occupancy contexts and QTBT are enabled.

6.6.2.14 Adjacent child neighbours (ACN)

6.6.2.14.1 Test bitstream ACN_A_Xidian

Specification: Enable adjacent child neighbour contextualization.

— geom_tree_type is equal to 0 (occupancy tree).

— octree_adjacent_child_enabled is equal to 1.

— octree_neigh_window_log2_minus1 is equal to 7.

Functional stage: Geometry coding in Main profile.

Purpose: Check if the decoder can properly decode bitstreams in which adjacent child neighbour contextualization is enabled.

6.6.2.14.2 Test bitstream ACN_B_Xidian

Specification: Disable adjacent child neighbour contextualization.

— geom_tree_type is equal to 0 (occupancy tree).

— occtree_adjacent_child_enabled is equal to 0.

Functional stage: Geometry coding in Main profile.

Purpose: Check if the decoder can properly decode bitstreams in which adjacent child neighbour contextualization is disabled.

6.6.2.14.3 Test bitstream ACN_C_XidianUniv

Specification: Enable adjacent child neighbour contextualization.

— geom_tree_type is equal to 0 (occupancy tree).

— occtree_adjacent_child_enabled is equal to 1.

— occtree_neigh_window_log2_minus1 is equal to 7.

Functional stage: Geometry coding in Dense profile.

Purpose: Check if the decoder can properly decode bitstreams in which adjacent child neighbour contextualization is enabled.

6.6.2.14.4 Test bitstream ACN_D_XidianUniv

Specification: Disable adjacent child neighbour contextualization.

— geom_tree_type is equal to 0 (occupancy tree).

— occtree_adjacent_child_enabled is equal to 0.

Functional stage: Geometry coding in Dense profile.

Purpose: Check if the decoder can properly decode bitstreams in which adjacent child neighbour contextualization is disabled.

6.6.2.15 Planar (PLANAR)

6.6.2.15.1 Test bitstream PLANAR_A_Ofinno

Specification: Planar mode is not enabled to code geometry with occupancy tree.

— geom_tree_type is equal to 0 (occupancy tree).

— occtree_planar_enabled is equal to 0.

Functional stage: Geometry coding in Main profile and Dense profile.

Purpose: Check if the decoder can properly decode bitstream in which planar mode is not enabled.

6.6.2.15.2 Test bitstream PLANAR_B_Ofinno

Specification: Planar mode is enabled to code geometry with occupancy tree, and planar mode is always eligible for any axis.

— geom_tree_type is equal to 0 (occupancy tree).

— occtree_planar_enabled is equal to 1.

— occtree_planar_threshold[0] is equal to 0.

— occtree_planar_threshold[1] is equal to 0.

— occtree_planar_threshold[2] is equal to 0.

Functional stage: Geometry coding in Main profile and Dense profile.

Purpose: Check if the decoder can properly decode bitstream in which planar mode is used with a planar mode being always eligible for any axis.

6.6.2.15.3 Test bitstream PLANAR_C_Ofinno

Specification: Planar mode is enabled to code geometry with occupancy tree, and planar mode is always eligible for most probable axis, never eligible for less probable axis, and half eligible for remaining axis.

— geom_tree_type is equal to 0 (occupancy tree).

— occtree_planar_enabled is equal to 1.

— occtree_planar_threshold[0] is equal to 0.

— occtree_planar_threshold[1] is equal to 64.

— occtree_planar_threshold[2] is equal to 127.

Functional stage: Geometry coding in Main and Dense profile.

Purpose: Check if the decoder can properly decode bitstream in which planar mode is used with a planar mode being always eligible for most probable axis, never eligible for less probable axis, and half eligible for remaining axis.

6.6.2.16 Angular-IDCM (AngIDCM)

6.6.2.16.1 Test bitstream AngIDCM_A_Qualcomm

Specification: Both IDCM and in-tree quantization in occupancy tree are enabled.

— geom_tree_type is equal to 0 (occupancy tree).

— geom_scaling_enabled is equal to 1.

— occtree_direct_coding_mode equal to 3.

— geom_qp equal to 10.

— geom_qp_mul_log2 is equal to 1.

— ptree_qp_period_log2 is equal to 8.

Functional stage: Geometry coding in Main profile.

Purpose: Check if the decoder can properly decode bitstreams in which both IDCM and in-tree quantization in occupancy tree are enabled.

6.6.2.16.2 Test bitstream AngIDCM_B_Qualcomm

Specification: Both IDCM and in-tree quantization in occupancy tree are enabled.

— geom_tree_type is equal to 0 (occupancy tree).

— geom_scaling_enabled is equal to 1.

— occtree_direct_coding_mode equal to 3.

— geom_qp equal to 9.

— geom_qp_mul_log2 is equal to 0.

— ptree_qp_period_log2 is equal to 7.

Functional stage: Geometry coding in Main profile.

Purpose: Check if the decoder can properly decode bitstreams in which both IDCM and in-tree quantization in occupancy tree are enabled.

6.6.2.16.3 Test bitstream AngIDCM_C_Qualcomm

Specification: Both IDCM and angular mode in occupancy tree are enabled.

— geom_tree_type is equal to 0 (occupancy tree).

— occtree_direct_coding_mode is equal to 1.

Functional stage: Geometry coding in Main profile.

Purpose: Check if the decoder can properly decode bitstreams in which both IDCM and angular mode in occupancy tree are enabled.

6.6.2.16.4 Test bitstream AngIDCM_D_Qualcomm

Specification: Both IDCM and angular mode in occupancy tree are enabled.

— geom_tree_type is equal to 0 (occupancy tree).

— occtree_direct_coding_mode is equal to 2.

Functional stage: Geometry coding in Main profile.

Purpose: Check if the decoder can properly decode bitstreams in which both IDCM and angular mode in occupancy tree are enabled.

6.6.2.16.5 Test bitstream AngIDCM_E_Qualcomm

Specification: Both IDCM and angular mode in occupancy tree are enabled.

— geom_tree_type is equal to 0 (occupancy tree).

— occtree_direct_coding_mode is equal to 3.

Functional stage: Geometry coding in Main profile.

Purpose: Check if the decoder can properly decode bitstreams in which both IDCM and angular mode in occupancy tree are enabled.

6.6.2.16.6 Test bitstream AngIDCM_A_Ofinno

Specification: Angular mode is enabled to encode geometry with occupancy tree, point positions are coded by eligible direct nodes of the occupancy tree and direct nodes that code two points shall jointly code their positions.

— geom_tree_type is equal to 0 (occupancy tree).

— geom_angular_enabled is equal to 1.

— occtree_direct_coding_mode is equal to 3.

— occtree_direct_joint_coding_enabled is equal to 1.

Functional stage: Geometry coding in Main profile.

Purpose: Check if the decoder can properly decode bitstream in which direct nodes jointly code two points.

6.6.2.16.7 Test bitstream AngIDCM_B_Ofinno[5)]

Specification: Angular mode is enabled to encode geometry with occupancy tree, point positions are coded by eligible direct nodes of the occupancy tree and direct nodes that code two points shall not jointly code their positions.

— geom_tree_type is equal to 0 (occupancy tree).

— geom_angular_enabled is equal to 1.

— occtree_direct_coding_mode is equal to 3.

— occtree_direct_joint_coding_enabled is equal to 0.

Functional stage: Geometry coding in Main profile.

Purpose: Check if the decoder can properly decode bitstream in which direct nodes shall not jointly code two points.

6.6.2.17 Angular-Planar (AngPLANAR)

6.6.2.17.1 Test bitstream AngPLANAR_A_Qualcomm

Specification: Both Planar and angular mode in occupancy tree are enabled. Angular origin is not presented.

— geom_tree_type is equal to 0 (occupancy tree).

— geom_angular_enabled is equal to 1.

— slice_angular_origin_present is equal to 0.

Functional stage: Geometry coding in Main profile.

Purpose: Check if the decoder can properly decode bitstreams in which both Planar and angular mode in occupancy tree are enabled and angular origin is not included.

6.6.2.17.2 Test bitstream AngPLANAR_B_Qualcomm

Specification: Both Planar and angular mode in occupancy tree are enabled. Angular origin is presented.

— geom_tree_type is equal to 0 (occupancy tree).

5) This information is given for the convenience of users of this document and does not constitute an endorsement by ISO or IEC of the product named.

— geom_angular_enabled is equal to 1.

— slice_angular_origin_present is equal to 1.

Functional stage: Geometry coding in Main profile.

Purpose: Check if the decoder can properly decode bitstreams in which both Planar and angular mode in occupancy tree are enabled and angular origin is included.

6.6.2.17.3 Test bitstream AngPLANAR_C_Qualcomm

Specification: Planar, angular mode and in-tree quantization in occupancy tree are enabled.

— geom_tree_type is equal to 0 (occupancy tree).

— geom_angular_enabled is equal to 1.

— geom_scaling_enabled is equal to 1.

— geom_qp equal to 10.

— geom_qp_mul_log2 is equal to 1.

— ptree_qp_period_log2 is equal to 8.

Functional stage: Geometry coding in Main profile.

Purpose: Check if the decoder can properly decode bitstreams in which Planar, angular mode and in-tree quantization in occupancy tree are enabled.

6.6.2.17.4 Test bitstream AngPLANAR_D_Qualcomm

Specification: Planar, angular mode and in-tree quantization in occupancy tree are enabled.

— geom_tree_type is equal to 0 (occupancy tree).

— geom_angular_enabled is equal to 1.

— geom_scaling_enabled is equal to 1.

— geom_qp equal to 9.

— geom_qp_mul_log2 is equal to 0.

— ptree_qp_period_log2 is equal to 7.

Functional stage: Geometry coding in Main profile.

Purpose: Check if the decoder can properly decode bitstreams in which Planar, angular mode and in-tree quantization in occupancy tree are enabled.

6.6.2.17.5 Test bitstream AngPLANAR_E_Qualcomm

Specification: Both Planar and angular mode in the occupancy tree are enabled. Planar buffer is enabled.

— geom_tree_type is equal to 0 (occupancy tree).

— occtree_planar_buffer_disabled is equal to 0.

Functional stage: Geometry coding in Main profile.

Purpose: Check if the decoder can properly decode bitstreams in which both Planar and angular mode in occupancy tree are enabled and planar buffer is enabled.

6.6.2.17.6 Test bitstream AngPLANAR_F_Qualcomm

Specification: Both Planar and angular mode in the occupancy tree are enabled. The number of lasers is 1.

— geom_tree_type is equal to 0 (occupancy tree).

— num_beams_minus1 is equal to 0.

Functional stage: Geometry coding in Main profile.

Purpose: Check if the decoder can properly decode bitstreams in which both Planar and angular mode in occupancy tree are enabled and number of lasers is specified.

6.6.2.17.7 Test bitstream AngPLANAR_G_Qualcomm

Specification: Both Planar and angular mode in the occupancy tree are enabled. The number of lasers is 16.

— geom_tree_type is equal to 0 (occupancy tree).

— num_beams_minus1 is equal to 15.

Functional stage: Geometry coding in Main profile.

Purpose: Check if the decoder can properly decode bitstreams in which both Planar and angular mode in occupancy tree are enabled and number of lasers is specified.

6.6.2.17.8 Test bitstream AngPLANAR_H_Qualcomm

Specification: Both Planar and angular mode in the occupancy tree are enabled. The number of lasers is 64.

— geom_tree_type is equal to 0 (occupancy tree).

— num_beams_minus1 is equal to 63.

Functional stage: Geometry coding in Main profile.

Purpose: Check if the decoder can properly decode bitstreams in which both Planar and angular mode in occupancy tree are enabled and number of lasers is specified.

6.6.2.17.9 Test bitstream AngPLANAR_I_Qualcomm

Specification: Both Planar and angular mode in the occupancy tree are enabled. The number of lasers is 255.

— geom_tree_type is equal to 0 (occupancy tree).

— num_beams_minus1 is equal to 254.

Functional stage: Geometry coding in Main profile.

Purpose: Check if the decoder can properly decode bitstreams in which both Planar and angular mode in occupancy tree are enabled and number of lasers is specified.

6.6.2.18 Predictive geometry (PredGEO)

6.6.2.18.1 Test bitstream PredGEO_A_LGE

Specification: The bitstream tests Predictive geometry coding with a point cloud which has one point.

— geom_tree_type is equal to 1 (predictive tree).

Functional stage: Geometry coding in Predictive profile.

Purpose: Check that the decoder can properly decode bitstreams which Predictive geometry coding is enabled and includes one point in Predictive profile.

6.6.2.18.2 Test bitstream PredGEO_B_LGE

Specification: The bitstream tests Predictive geometry coding with a point cloud which has two points.

— geom_tree_type is equal to 1 (predictive tree).

Functional stage: Geometry coding in Predictive profile.

Purpose: Check that the decoder can properly decode bitstreams which Predictive geometry coding is enabled and includes two points in Predictive profile.

6.6.2.18.3 Test bitstream PredGEO_C_LGE

Specification: The bitstream tests Predictive geometry coding with a point cloud which has 21 bits depth.

— geom_tree_type is equal to 1 (predictive tree).

— level_idc is equal to 80.

Functional stage: Geometry coding in Predictive profile.

Purpose: Check that the decoder can properly decode bitstreams which Predictive geometry coding is enabled and uses 21 bits depth in Predictive profile.

6.6.2.18.4 Test bitstream PredGEO_D_LGE

Specification: The bitstream tests multiple predictive trees per slice with Predictive geometry coding.

— geom_tree_type is equal to 1 (predictive tree).

— Number of predictive trees per slice is equal to 8.

— geom_angular_enabled is equal to 1.

Functional stage: Geometry coding in Predictive profile.

Purpose: Check that the decoder can properly decode bitstreams which Predictive geometry coding is enabled and includes multiple predictive trees per slice in Predictive profile.

6.6.2.19 Predictive duplicate (PredDUP)

6.6.2.19.1 Test bitstream PredDUP_A/B_LGE

Specification: The bitstream tests Predictive geometry coding with duplicate point count coding enabled.

— geom_tree_type is equal to 1 (predictive tree).

— geom_unique_points_flag is equal to 0.

— geom_dup_point_counts_enabled is equal to 1.

Functional stage: Geometry coding in Predictive profile.

Purpose: Check that the decoder can properly decode bitstreams which Predictive geometry coding is enabled and duplicate point count coding is enabled in Predictive profile.

6.6.2.19.2 Test bitstream PredDUP_C_LGE

Specification: The bitstream tests Predictive geometry coding with duplicate point count coding disabled.

— geom_tree_type is equal to 1 (predictive tree).

— geom_unique_points_flag is equal to 1.

— geom_dup_point_counts_enabled is equal to 0.

Functional stage: Geometry coding in Predictive profile.

Purpose: Check that the decoder can properly decode bitstreams which Predictive geometry coding is enabled and duplicate point count coding is disabled in Predictive profile.

6.6.2.20 Predictive geometry Angular mode (PredANG)

6.6.2.20.1 Test bitstream PredANG_A_LGE

Specification: The bitstream tests Predictive geometry coding with angular mode enabled. The bitstream tests with the point cloud which has one point.

— geom_tree_type is equal to 1 (predictive tree).

— geom_angular_enabled is equal to 1.

Functional stage: Geometry coding in Predictive profile.

Purpose: Check that the decoder can properly decode bitstreams which angular mode is enabled with Predictive geometry coding and includes one point.

6.6.2.20.2 Test bitstream PredANG_B_LGE

Specification: The bitstream tests Predictive geometry coding with angular mode enabled. The bitstream tests with the point cloud which has two points.

— geom_tree_type is equal to 1 (predictive tree).

— geom_angular_enabled is equal to 1.

Functional stage: Geometry coding in Predictive profile.

Purpose: Check that the decoder can properly decode bitstreams which angular mode is enabled with Predictive geometry coding and includes two points.

6.6.2.20.3 Test bitstream PredANG_C_LGE

Specification: The bitstream tests Predictive geometry coding with angular mode enabled. The bitstream tests with the point cloud which has 21 bits depth.

— geom_tree_type is equal to 1 (predictive tree).

— geom_angular_enabled is equal to 1.

— level_idc is equal to 80.

Functional stage: Geometry coding in Predictive profile.

Purpose: Check that the decoder can properly decode bitstreams which angular mode is enabled with Predictive geometry coding and uses 21 bits depth.

6.6.2.20.4 Test bitstream PredANG_D_LGE

Specification: The bitstream tests the number of lasers on Predictive geometry coding with angular mode enabled.

— geom_tree_type is equal to 1 (predictive tree).

— geom_angular_enabled is equal to 1.

— num_beams_minus1 is equal to 31.

Functional stage: Geometry coding in Predictive profile.

Purpose: Check that the decoder can properly decode bitstreams in which num_beams_minus1 is configured in Predictive geometry coding with angular mode enabled.

6.6.2.20.5 Test bitstream PredANG_E_LGE

Specification: The bitstream tests the number of lasers on Predictive geometry coding with angular mode enabled.

— geom_tree_type is equal to 1 (predictive tree).

— geom_angular_enabled is equal to 1.

— num_beams_minus1 is equal to 63.

Functional stage: Geometry coding in Predictive profile.

Purpose: Check that the decoder can properly decode bitstreams in which num_beams_minus1 is configured in Predictive geometry coding with angular mode enabled.

6.6.2.20.6 Test bitstream PredAng_A_Qualcomm

Specification: Test non-zero minimum value for radius in Mode 0 of geometry predictive coding.

— geom_tree_type is equal to 1 (predictive tree).

— ptn_radius_min is equal to 57.

— geom_angular_enabled is equal to 1.

Functional stage: Geometry coding in Predictive profile.

Purpose: Check if the decoder can properly decode bitstreams in which ptn_radius_min is configured in predictive coding.

6.6.2.20.7 Test bitstream PredAng_B_Qualcomm

Specification: Test non-zero minimum value for radius in Mode 0 of geometry predictive coding.

— geom_tree_type is equal to 1 (predictive tree).

— ptn_radius_min is equal to 0.

— geom_angular_enabled is equal to 1.

Functional stage: Geometry coding in Predictive profile.

Purpose: Check if the decoder can properly decode bitstreams in which ptn_radius_min is configured in predictive coding.

6.6.2.20.8 Test bitstream PredAng_C_Qualcomm

Specification: Test the number of lasers on geometry predictive coding.

— geom_tree_type is equal to 1 (predictive tree).

— num_beams_minus1 is equal to 0.

Functional stage: Geometry coding in Predictive profile.

Purpose: Check if the decoder can properly decode bitstreams in which num_beams_minus1 is configured in predictive coding.

6.6.2.20.9 Test bitstream PredAng_D_Qualcomm

Specification: Test the number of lasers on geometry predictive coding.

— geom_tree_type is equal to 1 (predictive tree).

— num_beams_minus1 = 15.

Functional stage: Geometry coding in Predictive profile.

Purpose: Check if the decoder can properly decode bitstreams in which num_beams_minus1 is configured in predictive coding.

6.6.2.20.10 Test bitstream PredAng_E_Qualcomm

Specification: Test the number of lasers on geometry predictive coding.

— geom_tree_type is equal to 1 (predictive tree).

— num_beams_minus1 = 63.

Functional stage: Geometry coding in Predictive profile.

Purpose: Check if the decoder can properly decode bitstreams in which num_beams_minus1 is configured in predictive coding.

6.6.2.20.11 Test bitstream PredAng_F_Qualcomm

Specification: Test the number of lasers on geometry predictive coding.

— geom_tree_type is equal to 1 (predictive tree).

— num_beams_minus1 = 254.

Functional stage: Geometry coding in Predictive profile.

Purpose: Check if the decoder can properly decode bitstreams in which num_beams_minus1 is configured in predictive coding.

6.6.2.21 Predictive geometry Scaling (PredGS)

6.6.2.21.1 Test bitstream PredGS_A_LGE

Specification: The bitstream tests QP combination on Predictive tree coding. Geometry scaling in occupancy tree is enabled. Base QP, slice QP, slice QP offset, and in-tree-level QP offset are all enabled.

— geom_tree_type is equal to 1 (predictive tree).

— geom_scaling_enabled is equal to 1.

— geom_qp is equal to 0.

— geom_qp_mul_log2 is equal to 0.

— slice_geom_qp_offset is equal to 0.

— ptree_qp_period_log2 is equal to 8.

— slice_ptree_qp_period_log2_offset is equal to 0.

Functional stage: Geometry coding in Predictive profile

Purpose: Check if the decoder can properly decode bitstreams which a QP combination on Predictive tree coding is enabled.

6.6.2.21.2 Test bitstream PredGS_B_LGE

Specification: The bitstream tests QP combination on Predictive tree coding. Geometry scaling in occupancy tree is enabled. Base QP, slice QP, slice QP offset, and in-tree-level QP offset are all enabled.

— geom_tree_type is equal to 1 (predictive tree).

— geom_scaling_enabled is equal to 1.

— geom_qp is equal to 8.

— geom_qp_mul_log2 is equal to 0.

— slice_geom_qp_offset is equal to 8.

— ptree_qp_period_log2 is equal to 8.

— slice_ptree_qp_period_log2_offset is equal to 0.

Functional stage: Geometry coding in Predictive profile

Purpose: Check if the decoder can properly decode bitstreams which a QP combination on Predictive tree coding is enabled.

6.6.2.21.3 Test bitstream PredGS_C_LGE

Specification: The bitstream tests QP combination on Predictive tree coding. Geometry scaling in occupancy tree is enabled. Base QP, slice QP, slice QP offset, and in-tree-level QP offset are all enabled.

— geom_tree_type is equal to 1 (predictive tree).

— geom_scaling_enabled is equal to 1.

— geom_qp is equal to 8.

— geom_qp_mul_log2 is equal to 0.

— slice_geom_qp_offset is equal to 8.

— ptree_qp_period_log2 is equal to 0.

— slice_ptree_qp_period_log2_offset is equal to 8.

Functional stage: Geometry coding in Predictive profile

Purpose: Check if the decoder can properly decode bitstreams which a QP combination on Predictive tree coding is enabled.

6.6.2.21.4 Test bitstream PredGS_A_Qualcomm

Specification: Test QP multiplier when geometry quantization is used in predictive geometry coding.

— geom_tree_type is equal to 1 (predictive tree).

— geom_scaling_enabled is equal to 1.

— geom_qp is equal to 5.

— geom_qp_mul_log2 is equal to 0.

— ptree_qp_period_log2 is equal to 8.

— slice_ptree_qp_period_log2_offset is equal to 0.

Functional stage: Geometry coding in Predictive profile.

Purpose: Check if the decoder can properly decode bitstreams in which QP multiplier in predictive coding.

6.6.2.21.5 Test bitstream PredGS_B_Qualcomm

Specification: Test QP multiplier when geometry quantization is used in predictive geometry coding.

— geom_tree_type is equal to 1 (predictive tree).

— geom_scaling_enabled is equal to 1.

— geom_qp is equal to 7.

— geom_qp_mul_log2 is equal to 1.

— ptree_qp_period_log2 is equal to 8.

— slice_ptree_qp_period_log2_offset is equal to 0.

Functional stage: Geometry coding in Predictive profile.

Purpose: Check if the decoder can properly decode bitstreams in which QP multiplier in predictive coding.

6.6.2.21.6 Test bitstream PredGS_C_Qualcomm

Specification: Test QP multiplier when geometry quantization is used in predictive geometry coding.

— geom_tree_type is equal to 1 (predictive tree).

— geom_scaling_enabled is equal to 1.

— geom_qp is equal to 5.

— geom_qp_mul_log2 is equal to 2.

— ptree_qp_period_log2 is equal to 8.

— slice_ptree_qp_period_log2_offset is equal to 0.

Functional stage: Geometry coding in Predictive profile.

Purpose: Check if the decoder can properly decode bitstreams in which QP multiplier in predictive coding.

6.6.2.21.7 Test bitstream PredGS_D_Qualcomm

Specification: Test QP multiplier when geometry quantization is used in predictive geometry coding.

— geom_tree_type is equal to 1 (predictive tree).

— geom_scaling_enabled is equal to 1.

— geom_qp is equal to 3.

— geom_qp_mul_log2 is equal to 3.

— ptree_qp_period_log2 is equal to 8.

— slice_ptree_qp_period_log2_offset is equal to 0.

Functional stage: Geometry coding in Predictive profile.

Purpose: Check if the decoder can properly decode bitstreams in which QP multiplier in predictive coding.

6.6.2.22 Change APS (CAPS)

6.6.2.22.1 Test bitstream CAPS_A_Tencent

Specification: The bitstreams includes four APS, each with different attribute encoding types. Each slice in a frame employs one of the four GPS.

— aps.attr_encoding = 0, 1, 2, 3.

Functional stage: Attribute coding in Dense profile.

Purpose: Check if the decoder can properly decode bitstreams in which different attribute coding type is employed in different slices in a frame.

6.6.2.23 General attribute (GATT)

6.6.2.23.1 Test bitstream GATT_A_LGE

Specification: The bitstream tests attribute Region Adaptive Hierarchical Transform with a point cloud which has one attribute with three components.

— attr_coding_type is equal to 0 (RAHT).

Functional stage: Attribute coding in Simple profile.

Purpose: Check that the decoder can properly decode bitstreams which includes one attribute with three components and is used attribute Region Adaptive Hierarchical Transform.

6.6.2.23.2 Test bitstream GATT_B_LGE

Specification: The bitstream tests attribute Region Adaptive Hierarchical Transform with a point cloud which has one attribute with three components.

— attr_coding_type is equal to 0 (RAHT).

Functional stage: Attribute coding in Main profile and Dense profile.

Purpose: Check that the decoder can properly decode bitstreams which includes one attribute with three components and is used attribute Region Adaptive Hierarchical Transform.

6.6.2.23.3 Test bitstream GATT_C_LGE

Specification: The bitstream tests attribute LoD with Lifting Transform with a point cloud which has one attribute with three components.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

Functional stage: Attribute coding in Simple profile.

Purpose: Check that the decoder can properly decode bitstreams which includes one attribute with three components and is used attribute LoD with Lifting Transform.

6.6.2.23.4 Test bitstream GATT_D_LGE

Specification: The bitstream tests attribute LoD with Lifting Transform with a point cloud which has one attribute with three components.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

Functional stage: Attribute coding in Main profile and Dense profile.

Purpose: Check that the decoder can properly decode bitstreams which includes one attribute with three components and is used attribute LoD with Lifting Transform.

6.6.2.23.5 Test bitstream GATT_E_LGE

Specification: The bitstream tests attribute Region Adaptive Hierarchical Transform with a point cloud which has one attribute with one component.

— attr_coding_type is equal to 0 (RAHT).

Functional stage: Attribute coding in Simple profile.

Purpose: Check that the decoder can properly decode bitstreams which includes one attribute with one component and is used attribute Region Adaptive Hierarchical Transform.

6.6.2.23.6 Test bitstream GATT_F_LGE

Specification: The bitstream tests attribute Region Adaptive Hierarchical Transform with a point cloud which has one attribute with one component.

— attr_coding_type is equal to 0 (RAHT).

Functional stage: Attribute coding in Main profile and Dense profile.

Purpose: Check that the decoder can properly decode bitstreams which one attribute with one component and is used attribute Region Adaptive Hierarchical Transform.

6.6.2.23.7 Test bitstream GATT_G_LGE

Specification: The bitstream tests attribute LoD with Lifting Transform with a point cloud which one attribute with one component.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

Functional stage: Attribute coding in Simple profile.

Purpose: Check that the decoder can properly decode bitstreams which includes one attribute with one component and is used attribute LoD with Lifting Transform in Simple profile.

6.6.2.23.8 Test bitstream GATT_H_LGE

Specification: The bitstream tests attribute LoD with Lifting Transform with a point cloud which has one attribute with one component.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

Functional stage: Attribute coding in Main profile and Dense profile.

Purpose: Check that the decoder can properly decode bitstreams which includes one attribute with one component and is used attribute LoD with Lifting Transform.

6.6.2.23.9 Test bitstream GATT_I_LGE

Specification: The bitstream tests attribute Region Adaptive Hierarchical Transform with a point cloud which has two attributes.

— attr_coding_type is equal to 0 (RAHT).

Functional stage: Attribute coding in Simple profile.

Purpose: Check that the decoder can properly decode bitstreams which includes two attributes and is used attribute Region Adaptive Hierarchical Transform.

6.6.2.23.10 Test bitstream GATT_J_LGE

Specification: The bitstream tests attribute Region Adaptive Hierarchical Transform with a point cloud which has two attributes.

— attr_coding_type is equal to 0 (RAHT).

Functional stage: Attribute coding in Main profile and Dense profile.

Purpose: Check that the decoder can properly decode bitstreams which includes two attributes and is used attribute Region Adaptive Hierarchical Transform.

6.6.2.23.11 Test bitstream GATT_K_LGE

Specification: The bitstream tests attribute LoD with Lifting Transform with a point cloud which has two attributes.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

Functional stage: Attribute coding in Simple profile.

Purpose: Check that the decoder can properly decode bitstreams which includes two attributes and is used attribute LoD with Lifting Transform.

6.6.2.23.12 Test bitstream GATT_L_LGE

Specification: The bitstream tests attribute LoD with Lifting Transform with a point cloud which has two attributes.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

Functional stage: Attribute coding in Main profile and Dense profile.

Purpose: Check that the decoder can properly decode bitstreams which includes two attributes and is used attribute LoD with Lifting Transform.

6.6.2.23.13 Test bitstream GATT_M_LGE

Specification: The bitstream tests on setting default attribute value.

— attr_default_value is equal to [255, 0, 0].

Functional stage: Attribute coding in Simple profile.

Purpose: Check that the decoder can properly decode bitstreams which set default attribute value.

6.6.2.23.14 Test bitstream GATT_N_LGE

Specification: The bitstream tests on setting default attribute value.

— attr_default_value is equal to [255, 0, 0].

Functional stage: Attribute coding in Main and Dense profile.

Purpose: Check that the decoder can properly decode bitstreams which set default attribute value.

6.6.2.23.15 Test bitstream GATT_O_LGE

Specification: The bitstream tests raw attribute data.

— attr_coding_type is equal to 3 (Raw attribute data).

Functional stage: Attribute coding in Simple profile.

Purpose: Check that the decoder can properly decode bitstreams which raw attribute data is enabled.

6.6.2.23.16 Test bitstream GATT_P_LGE

Specification: The bitstream tests raw attribute data.

— attr_coding_type is equal to 3 (Raw attribute data).

Functional stage: Attribute coding in Main profile and Dense profile.

Purpose: Check that the decoder can properly decode bitstreams which raw attribute data is enabled.

6.6.2.24 Quantization (QuantATT)

There are 19 streams for each attr_coding_type.

— QuantATT_A01_Panasonic ~ QuantATT_A19_Panasonic: test on attr_coding_type = 0 (RAHT)

— QuantATT_B01_Panasonic ~ QuantATT_B19_Panasonic: test on attr_coding_type = 1 (LoD with Predicting Transform)

— QuantATT_C01_Panasonic ~ QuantATT_C19_Panasonic: test on attr_coding_type = 2 (LoD with Lifting Transform)

6.6.2.24.1 Test bitstream QuantATT_A01(B01, C01)_Panasonic

Specification: The bitstream test colour attribute QP combination on RAHT (Predicting, Lifting) and secondary QP offset is also tested.

— attr_primary_qp_minus4 equal to 2

— attr_secondary_qp_offset equal to -1

Functional stage: Attribute coding in Dense profile.

Purpose: Check that the decoder can properly decode bitstreams which is used attribute quantize mechanism (quantize first/second attribute component).

6.6.2.24.2 Test bitstream QuantATT_A02(B02, C02)_Panasonic

Specification: The bitstream test colour attribute maximum QP combination on RAHT (Predicting, Lifting) and secondary QP offset is also tested.

— attr_primary_qp_minus4 equal to 47.

— attr_secondary_qp_offset equal to 0.

Functional stage: Attribute coding in Dense profile.

Purpose: Check that the decoder can properly decode bitstreams which is used attribute quantize mechanism (quantize first/second attribute component).

6.6.2.24.3 Test bitstream QuantATT_A03(B03, C03)_Panasonic

Specification: The bitstream test colour attribute maximum and minimum QP combination on RAHT (Predicting, Lifting) and secondary QP offset is also tested.

— attr_primary_qp_minus4 equal to 47.

— attr_secondary_qp_offset equal to -47.

Functional stage: Attribute coding in Dense profile.

Purpose: Check that the decoder can properly decode bitstreams which is used attribute quantize mechanism (quantize first/second attribute component).

6.6.2.24.4 Test bitstream QuantATT_A04(B04, C04)_Panasonic

Specification: The bitstream test colour attribute maximum and minimum QP combination on RAHT (Predicting, Lifting) and secondary QP offset is also tested.

— attr_primary_qp_minus4 equal to 0.

— attr_secondary_qp_offset equal to 47.

Functional stage: Attribute coding in Dense profile.

Purpose: Check that the decoder can properly decode bitstreams which is used attribute quantize mechanism (quantize first/second attribute component).

6.6.2.24.5 Test bitstream QuantATT_A05(B05, C05)_Panasonic

Specification: The bitstream test colour attribute maximum QP combination on RAHT (Predicting, Lifting) and secondary QP offset is also tested. QP clipping is exercised.

— attr_primary_qp_minus4 equal to 95.

— attr_secondary_qp_offset equal to 95.

Functional stage: Attribute coding in Dense profile.

Purpose: Check that the decoder can properly decode bitstreams which is used attribute quantize mechanism (quantize first/second attribute component).

6.6.2.24.6 Test bitstream QuantATT_A06(B06, C06)_Panasonic

Specification: The bitstream test colour attribute minimum QP combination on RAHT (Predicting, Lifting) and secondary QP offset is also tested. QP clipping is exercised.

— attr_primary_qp_minus4 equal to 0.

— attr_secondary_qp_offset equal to -95.

Functional stage: Attribute coding in Dense profile.

Purpose: Check that the decoder can properly decode bitstreams which is used attribute quantize mechanism (quantize first/second attribute component).

6.6.2.24.7 Test bitstream QuantATT_A07(B07, C07)_Panasonic

Specification: The bitstream test colour attribute maximum and minimum QP combination with slice QP enabled on RAHT (Predicting, Lifting) and secondary QP offset is also tested. QP clipping is exercised.

— attr_primary_qp_minus4 equal to 10.

— attr_secondary_qp_offset equal to 10.

— attr_qp_offsets_present equal to 1.

— attr_qp_offset[0] equal to 0, 2, -2, 95, 0, -95.

— attr_qp_offset[1] equal to 0, -2, 2, -47, 95, -95.

Functional stage: Attribute coding in Dense profile

Purpose: Check that the decoder can properly decode bitstreams which is used attribute quantize mechanism (quantize first/second attribute component).

6.6.2.24.8 Test bitstream QuantATT_A08(B08, C08)_Panasonic

Specification: The bitstream test colour attribute maximum and minimum QP combination with layer QP enabled on RAHT (Predicting, Lifting) and secondary QP offset is also tested. QP clipping is exercised.

— attr_primary_qp_minus4 equal to 10.

— attr_secondary_qp_offset equal to 10.

— attr_qp_layers_present equal to 1.

— attr_qp_layer_cnt_minus1 equal to 5.

— attr_qp_layer_offset[6][2] equal to {{0, 0}, {2, 2}, {-2, 2}, {95, -47}, {0, 95}, {-95, -95}}.

Functional stage: Attribute coding in Dense profile.

Purpose: Check that the decoder can properly decode bitstreams which is used attribute quantize mechanism (quantize first/second attribute component).

6.6.2.24.9 Test bitstream QuantATT_A09(B09, C09)_Panasonic

Specification: The bitstream test colour attribute QP combination with layer QP enabled on RAHT (Predicting, Lifting) and secondary QP offset is also tested.

— attr_primary_qp_minus4 equal to 10.

— attr_secondary_qp_offset equal to 10.

— attr_qp_layers_present equal to 1.

— attr_qp_layer_cnt_minus1 equal to 0.

— attr_qp_layer_offset[0][2] equal to {{0, 0}}.

Functional stage: Attribute coding in Dense profile.

Purpose: Check that the decoder can properly decode bitstreams which is used attribute quantize mechanism (quantize first/second attribute component).

6.6.2.24.10 Test bitstream QuantATT_A10(B10, C10)_Panasonic

Specification: The bitstream test colour attribute QP combination with maximum layer QP enabled on RAHT (Predicting, Lifting) and secondary QP offset is also tested.

— attr_primary_qp_minus4 equal to 10.

— attr_secondary_qp_offset equal to 10.

— attr_qp_layers_present equal to 1.

— attr_qp_layer_cnt_minus1 equal to 20.

— attr_qp_layer_offset[21][2] equal to {{5, 5}, {5, 5},}.

Functional stage: Attribute coding in Dense profile.

Purpose: Check that the decoder can properly decode bitstreams which is used attribute quantize mechanism (quantize first/second attribute component).

6.6.2.24.11 Test bitstream QuantATT_A11(B11, C11)_Panasonic

Specification: The bitstream test colour attribute QP combination with region QP enabled on RAHT (Predicting, Lifting) and secondary QP offset is also tested. Bit-dependent QP Region Bits is exercised.

— attr_primary_qp_minus4 equal to 10.

— attr_secondary_qp_offset equal to 10.

— attr_qp_region_cnt equal to 1.

— attr_qp_region_origin_xyz[1][3] equal to {0,0,0}.

— attr_qp_region_size_minus1_xyz[1][3] to {1,1,1}.

— attr_qp_region_offset[2] equal to {0,5}.

Functional stage: Attribute coding in Dense profile.

Purpose: Check that the decoder can properly decode bitstreams which is used attribute quantize mechanism (quantize first/second attribute component).

6.6.2.24.12 Test bitstream QuantATT_A12(B12, C12)_Panasonic

Specification: The bitstream test colour attribute QP combination with region QP enabled on RAHT (Predicting, Lifting) and secondary QP offset is also tested. Bit-dependent QP Region Bits is exercised.

— attr_primary_qp_minus4 equal to 10.

— attr_secondary_qp_offset equal to 10.

— attr_qp_region_cnt equal to 1.

— attr_qp_region_origin_xyz[1][3] equal to {0,0,0}.

— attr_qp_region_size_minus1_xyz[1][3] is equal to $\{2^{20}-1,\ 2^{20}-1,\ 2^{20}-1\}$.

— attr_qp_region_offset[2] equal to {-5,5}.

Functional stage: Attribute coding in Dense profile.

Purpose: Check that the decoder can properly decode bitstreams which is used attribute quantize mechanism (quantize first/second attribute component).

6.6.2.24.13 Test bitstream QuantATT_A13(B13, C13)_Panasonic

Specification: The bitstream test colour attribute minimum QP combination with region QP enabled on RAHT (Predicting, Lifting) and secondary QP offset is also tested. QP clipping is exercised.

— attr_primary_qp_minus4 equal to 47.

— attr_secondary_qp_offset equal to 10.

— attr_qp_region_cnt equal to 1.

— attr_qp_region_origin_xyz[1][3] equal to {0,0,0}.

— attr_qp_region_size_minus1_xyz[1][3] is equal to {199, 199, 199}.

— attr_qp_region_offset[2] equal to {-47,-95}.

Functional stage: Attribute coding in Dense profile.

Purpose: Check that the decoder can properly decode bitstreams which is used attribute quantize mechanism (quantize first/second attribute component).

6.6.2.24.14 Test bitstream QuantATT_A14(B14, C14)_Panasonic

Specification: The bitstream test colour attribute maximum and minimum QP combination with region QP enabled on RAHT (Predicting, Lifting) and secondary QP offset is also tested.

— attr_primary_qp_minus4 equal to 0.

— attr_secondary_qp_offset equal to 0.

— attr_qp_region_cnt equal to 1.

— attr_qp_region_origin_xyz[1][3] equal to {0,0,0}.

— attr_qp_region_size_minus1_xyz[1][3] is equal to {199, 199, 199}.

— attr_qp_region_offset[2] equal to {47,-47}.

Functional stage: Attribute coding in Dense profile.

Purpose: Check that the decoder can properly decode bitstreams which is used attribute quantize mechanism (quantize first/second attribute component).

6.6.2.24.15 Test bitstream QuantATT_A15(B15, C15)_Panasonic

Specification: The bitstream test colour attribute maximum and minimum QP combination with region QP enabled on RAHT (Predicting, Lifting) and secondary QP offset is also tested. QP clipping is exercised.

— attr_primary_qp_minus4 equal to 10.

— attr_secondary_qp_offset equal to 10.

— attr_qp_region_cnt equal to 1.

— attr_qp_region_origin_xyz[1][3] equal to {0,0,0}.

— attr_qp_region_size_minus1_xyz[1][3] equal to {199, 199, 199}.

— attr_qp_region_offset[2] equal to {-95,47}.

Functional stage: Attribute coding in Dense profile.

Purpose: Check that the decoder can properly decode bitstreams which is used attribute quantize mechanism (quantize first/second attribute component).

6.6.2.24.16 Test bitstream QuantATT_A16(B16, C16)_Panasonic

Specification: The bitstream test colour attribute maximum QP combination with region QP enabled on RAHT (Predicting, Lifting) and secondary QP offset is also tested. QP clipping is exercised.

— attr_primary_qp_minus4 equal to 10.

— attr_secondary_qp_offset equal to 0.

— attr_qp_region_cnt equal to 1.

— attr_qp_region_origin_xyz[1][3] equal to {0,0,0}.

— attr_qp_region_size_minus1_xyz[1][3] equal to {200, 200, 200}.

— attr_qp_region_offset[2] equal to {95,95}.

Functional stage: Attribute coding in Dense profile.

Purpose: Check that the decoder can properly decode bitstreams which is used attribute quantize mechanism (quantize first/second attribute component).

6.6.2.24.17 Test bitstream QuantATT_A17(B17, C17)_Panasonic

Specification: The bitstream test colour attribute QP combination with slice QP, layer QP and region QP enabled on RAHT (Predicting, Lifting) and secondary QP offset is also tested.

— attr_primary_qp_minus4 equal to 5.

— attr_secondary_qp_offset equal to 2.

— attr_qp_offsets_present equal to 1.

— attr_qp_offset[0] equal to 1,2.

— attr_qp_offset[1] equal to 3,-1.

— attr_qp_layers_present equal to 1.

— attr_qp_layer_cnt_minus1 equal to 43.

— attr_qp_layer_offset[44][2] equal to {{0, 0}, {0, 0}, {-1, 1}, {1, -1}}.

— attr_qp_region_cnt equal to 1.

— attr_qp_region_origin_xyz[1][3] equal to {0,0,0}.

— attr_qp_region_size_minus1_xyz[1][3] equal to {199, 199, 199}

— attr_qp_region_offset[2] equal to {47,0}.

Functional stage: Attribute coding in Dense profile.

Purpose: Check that the decoder can properly decode bitstreams which is used attribute quantize mechanism (quantize first/second attribute component).

6.6.2.24.18 Test bitstream QuantATT_A18(B18, C18)_Panasonic

Specification: The bitstream test colour attribute QP combination with slice QP on RAHT (Predicting, Lifting) and secondary QP offset is also tested. The bit-dependent QP range is exercised. QP clipping is exercised.

— attr_primary_qp_minus4 equal to 10.

— attr_secondary_qp_offset equal to 10.

— attr_bitdepth_minus1 equal to 8.

— attr_qp_offsets_present equal to 1.

— attr_qp_offset[0] equal to 47.

— attr_qp_offset[1] equal to 0.

Functional stage: Attribute coding in Dense profile.

Purpose: Check that the decoder can properly decode bitstreams which is used attribute quantize mechanism (quantize first/second attribute component).

6.6.2.24.19 Test bitstream QuantATT_A19(B19, C19)_Panasonic

Specification: The bitstream test reflectance attribute QP combination with slice QP, layer QP and region QP enabled on RAHT (Predicting, Lifting) and secondary QP offset is also tested.

— attr_primary_qp_minus4 equal to 5.

— attr_qp_offsets_present equal to 1.

— attr_qp_offset[0] equal to 1.

— attr_qp_offset[1] equal to 3.

— attr_qp_layers_present equal to 1.

— attr_qp_layer_cnt_minus1 equal to 4.

— attr_qp_layer_offset[4][2] equal to {{0, 0}, {0, 0}, {-1, 1}, {1, -1}}.

— attr_qp_region_cnt equal to 1.

— attr_qp_region_origin_xyz[1][3] equal to {0,0,0}.

— attr_qp_region_size_minus1_xyz[1][3] equal to {3999, 3999, 3999}.

— attr_qp_region_offset[2] equal to {3,-3}.

Functional stage: Attribute coding in Main profile.

Purpose: Check that the decoder can properly decode bitstreams which is used attribute quantize mechanism (quantize first/second attribute component).

6.6.2.25 LoD and RAHT (LodRAHT)

6.6.2.25.1 Test bitstream LodRAHT_A_LGE

Specification: The bitstream tests LoD with Lifting Transform with the point cloud which has one point.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

Functional stage: Attribute coding in Main profile.

Purpose: Check that the decoder can properly decode bitstreams which includes one point and is used attribute Lifting Transform.

6.6.2.25.2 Test bitstream LodRAHT_B_LGE

Specification: The bitstream tests Region Adaptive Hierarchical Transform with the point cloud which has one point.

— attr_coding_type is equal to 0 (RAHT).

Functional stage: Attribute coding in Main profile.

Purpose: Check that the decoder can properly decode bitstreams which includes one point and is used attribute Region Adaptive Hierarchical Transform.

6.6.2.25.3 Test bitstream LodRAHT_C_LGE

Specification: The bitstream tests LoD with Lifting Transform with the point cloud which has one point.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

Functional stage: Attribute coding in Simple profile.

Purpose: Check that the decoder can properly decode bitstreams which includes one point and is used attribute Lifting Transform.

6.6.2.25.4 Test bitstream LodRAHT_D_LGE

Specification: The bitstream tests Region Adaptive Hierarchical Transform with the point cloud which has one point.

— attr_coding_type is equal to 0 (RAHT).

Functional stage: Attribute coding in Simple profile.

Purpose: Check that the decoder can properly decode bitstreams which includes one point and is used attribute Region Adaptive Hierarchical transform.

6.6.2.25.5 Test bitstream LodRAHT_E_LGE

Specification: The bitstream tests LoD with Lifting Transform with the point cloud which has two points.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

Functional stage: Attribute coding in Main profile.

Purpose: Check that the decoder can properly decode bitstreams which includes two points and is used attribute Lifting Transform.

6.6.2.25.6 Test bitstream LodRAHT_F_LGE

Specification: The bitstream tests Region Adaptive Hierarchical Transform with the point cloud which has two points.

— attr_coding_type is equal to 0 (RAHT).

Functional stage: Attribute coding in Main profile.

Purpose: Check that the decoder can properly decode bitstreams which includes two points and is used attribute Region Adaptive Hierarchical Transform.

6.6.2.25.7 Test bitstream LodRAHT_G_LGE

Specification: The bitstream tests LoD with Lifting Transform with the point cloud which has two points.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

Functional stage: Attribute coding in Simple profile.

Purpose: Check that the decoder can properly decode bitstreams which includes two points and is used attribute Lifting Transform.

6.6.2.25.8 Test bitstream LodRAHT_H_LGE

Specification: The bitstream tests Region Adaptive Hierarchical Transform with the point cloud which has two points.

— attr_coding_type is equal to 0 (RAHT).

Functional stage: Attribute coding in Simple profile.

Purpose: Check that the decoder can properly decode bitstreams which includes two points and is used attribute Region Adaptive Hierarchical transform.

6.6.2.25.9 Test bitstream LodRAHT_I_LGE

Specification: The bitstream tests LoD with Lifting Transform with the point cloud which has one point with multiple duplicate points.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

Functional stage: Attribute coding in Main profile.

Purpose: Check that the decoder can properly decode bitstreams which includes one point with multiple duplicate points and is used attribute Lifting Transform.

6.6.2.25.10 Test bitstream LodRAHT_J_LGE

Specification: The bitstream tests Region Adaptive Hierarchical Transform with the point cloud which has one point with multiple duplicate points.

— attr_coding_type is equal to 0 (RAHT).

Functional stage: Attribute coding in Main profile.

Purpose: Check that the decoder can properly decode bitstreams which includes one point with multiple duplicate points and is used attribute Region Adaptive Hierarchical Transform.

6.6.2.25.11 Test bitstream LodRAHT_K_LGE

Specification: The bitstream tests LoD with Lifting Transform with the point cloud which has one point with multiple duplicate points.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

Functional stage: Attribute coding in Simple profile.

Purpose: Check that the decoder can properly decode bitstreams which includes one point with multiple duplicate points and is used attribute Lifting Transform.

6.6.2.25.12 Test bitstream LodRAHT_L_LGE

Specification: The bitstream tests Region Adaptive Hierarchical Transform with the point cloud which has one point with multiple duplicate points.

— attr_coding_type is equal to 0 (RAHT).

Functional stage: Attribute coding in Simple profile.

Purpose: Check that the decoder can properly decode bitstreams which includes one point with multiple duplicate points and is used attribute Region Adaptive Hierarchical transform.

6.6.2.25.13 Test bitstream LodRAHT_M_LGE

Specification: The bitstream tests LoD with Lifting Transform with the point cloud which has 16 bits attributes.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

Functional stage: Attribute coding in Main profile.

Purpose: Check that the decoder can properly decode bitstreams which includes 16 bits attributes and is used attribute Region Adaptive Hierarchical Transform.

6.6.2.25.14 Test bitstream LodRAHT_N_LGE

Specification: The bitstream tests Region Adaptive Hierarchical Transform with the point cloud which has 16 bits attributes.

— attr_coding_type is equal to 0 (RAHT).

Functional stage: Attribute coding in Main profile.

Purpose: Check that the decoder can properly decode bitstreams which includes 16 bits attributes and is used attribute Region Adaptive Hierarchical Transform.

6.6.2.25.15 Test bitstream LodRAHT_O_LGE

Specification: The bitstream tests LoD with Lifting Transform with the point cloud which has 16 bits attributes.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

Functional stage: Attribute coding in Simple profile.

Purpose: Check that the decoder can properly decode bitstreams which includes 16 bits attributes and is used attribute Region Adaptive Hierarchical transform.

6.6.2.25.16 Test bitstream LodRAHT_P_LGE

Specification: The bitstream tests Region Adaptive Hierarchical Transform with the point cloud which has 16 bits attributes.

— attr_coding_type is equal to 0 (RAHT).

Functional stage: Attribute coding in Simple profile.

Purpose: Check that the decoder can properly decode bitstreams which includes 16 bits attributes and is used attribute Region Adaptive Hierarchical transform.

6.6.2.26 Single LoD (SLOD)

6.6.2.26.1 Test bitstream SLOD_A_LGE

Specification: The bitstream tests single detail level in LoD with Predicting Transform with Morton order-based sorting enabled.

— attr_coding_type is equal to 1 (LoD with Predicting Transform).

— lod_max_levels_minus1 is equal to 0.

— attr_canonical_order_enabled is equal to 1.

— pred_intra_lod_search_range is equal to -1.

— pred_intra_min_lod is equal to 0.

Functional stage: Attribute coding in Simple profile.

Purpose: Check that the decoder can properly decode bitstreams which has single detail level and Morton sorting enabled.

6.6.2.26.2 Test bitstream SLOD_B_LGE

Specification: The bitstream tests single detail level in LoD with Predicting Transform with Morton order-based sorting enabled.

— attr_coding_type is equal to 1 (LoD with Predicting Transform).

— lod_max_levels_minus1 is equal to 0.

— attr_canonical_order_enabled is equal to 1.

— pred_intra_lod_search_range is equal to -1.

— pred_intra_min_lod is equal to 0.

Functional stage: Attribute coding in Main profile and Dense profile.

Purpose: Check that the decoder can properly decode bitstreams which has single detail level and Morton sorting enabled.

6.6.2.26.3 Test bitstream SLOD_C_LGE

Specification: The bitstream tests single detail level in LoD with Predicting Transform with Morton order-based sorting disabled.

— attr_coding_type is equal to 1 (LoD with Predicting Transform).

— lod_max_levels_minus1 is equal to 0.

— attr_canonical_order_enabled is equal to 0.

— pred_intra_lod_search_range is equal to -1.

— pred_intra_min_lod is equal to 0.

Functional stage: Attribute coding in Simple profile.

Purpose: Check that the decoder can properly decode bitstreams which has single detail level and Morton sorting disabled.

6.6.2.26.4 Test bitstream SLOD_D_LGE

Specification: The bitstream tests single detail level in LoD with Predicting Transform with Morton order-based sorting disabled.

— attr_coding_type is equal to 1 (LoD with Predicting Transform).

— lod_max_levels_minus1 is equal to 0.

— attr_canonical_order_enabled is equal to 0.

— pred_intra_lod_search_range is equal to -1.

— pred_intra_min_lod is equal to 0.

Functional stage: Attribute coding in Main profile and Dense profile.

Purpose: Check that the decoder can properly decode bitstreams which has single detail level and Morton sorting disabled.

6.6.2.27 Number of LoDs (NumLOD)

6.6.2.27.1 Test bitstream NumLOD_A_LGE

Specification: The bitstream exercises the range of number of the detail levels. The bitstream tests single detail level in LoD with Predicting Transform. Morton order-based sorting is enabled.

— attr_coding_type is equal to 1 (LoD with Predicting Transform).

— lod_max_levels_minus1 is equal to 0.

— attr_canonical_order_enabled is equal to 1.

— pred_intra_lod_search_range is equal to -1.

— pred_intra_min_lod is equal to 0.

Functional stage: Attribute coding in Simple profile.

Purpose: Check that the decoder can properly decode bitstreams which has single detail level.

6.6.2.27.2 Test bitstream NumLOD_B_LGE

Specification: The bitstream exercises the range of number of the detail levels. The bitstream tests single detail level in LoD with Predicting Transform. Morton order-based sorting is enabled.

— attr_coding_type is equal to 1 (LoD with Predicting Transform).

— lod_max_levels_minus1 is equal to 0.

— attr_canonical_order_enabled is equal to 1.

— pred_intra_lod_search_range is equal to -1.

— pred_intra_min_lod is equal to 0.

Functional stage: Attribute coding in Main profile and Dense profile.

Purpose: Check that the decoder can properly decode bitstreams which has single detail level.

6.6.2.27.3 Test bitstream NumLOD_C_LGE

Specification: The bitstream exercises the range of number of the detail levels. The bitstream tests six detail levels in LoD with Predicting Transform. Morton order-based sorting is disabled.

— attr_coding_type is equal to 1 (LoD with Predicting Transform).

— lod_max_levels_minus1 is equal to 5.

Functional stage: Attribute coding in Simple profile.

Purpose: Check that the decoder can properly decode bitstreams which has six detail levels.

6.6.2.27.4 Test bitstream NumLOD_D_LGE

Specification: The bitstream exercises the range of number of the detail levels. The bitstream tests six detail levels in LoD with Lifting Transform. Morton order-based sorting is disabled.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

— lod_max_levels_minus1 is equal to 5.

Functional stage: Attribute coding in Main profile and Dense profile.

Purpose: Check that the decoder can properly decode bitstreams which has six detail levels.

6.6.2.27.5 Test bitstream NumLOD_E_LGE

Specification: The bitstream exercises the range of number of the detail levels. The bitstream tests twelve detail levels in LoD with Predicting Transform. Morton order-based sorting is disabled.

— attr_coding_type is equal to 1 (LoD with Predicting Transform).

— lod_max_levels_minus1 is equal to 11.

Functional stage: Attribute coding in Simple profile.

Purpose: Check that the decoder can properly decode bitstreams which has twelve detail levels.

6.6.2.27.6 Test bitstream NumLOD_F_LGE

Specification: The bitstream exercises the range of number of the detail levels. The bitstream tests twelve detail levels in LoD with Lifting Transform. Morton order-based sorting is disabled.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

— lod_max_levels_minus1 is equal to 11.

Functional stage: Attribute coding in Main profile and Dense profile.

Purpose: Check that the decoder can properly decode bitstreams which has twelve detail levels.

6.6.2.28 LoD method: Decimation (DecLOD)

6.6.2.28.1 Test bitstream DecLOD_A_LGE

Specification: The bitstream uses Periodic subsampling to generate detail levels in LoD with Lifting Transform. The bitstream tests different sampling period for each detail level.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

— lod_decimation_mode is equal to 1 (Periodic subsampling).

— lod_sampling_period_minus2 is equal to [2,2,2,2,2,1,1,1,1,1,1,1].

Functional stage: Attribute coding in Simple profile.

Purpose: Check that the decoder can properly decode bitstreams which has different sampling period for each detail level with the periodic subsampling-based LoD generation.

6.6.2.28.2 Test bitstream DecLOD_B_LGE

Specification: The bitstream uses Periodic subsampling to generate detail levels in LoD with Lifting Transform. The bitstream tests different sampling period for each detail.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

— lod_decimation_mode is equal to 1 (Periodic subsampling).

— lod_sampling_period_minus2 is equal to [2,2,2,2,2,1,1,1,1,1,1,1].

Functional stage: Attribute coding in Main profile and Dense profile.

Purpose: Check that the decoder can properly decode bitstreams which has different sampling period for each detail level with the periodic subsampling-based LoD generation.

6.6.2.28.3 Test bitstream DecLOD_C_LGE

Specification: The bitstream uses Periodic subsampling to generate detail levels in LoD with Lifting Transform. The bitstream exercises on range of sampling period.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

— lod_decimation_mode is equal to 1 (Periodic subsampling).

— lod_sampling_period_minus2 is equal to 0.

Functional stage: Attribute coding in Simple profile.

Purpose: Check that the decoder can properly decode bitstreams which the periodic subsampling-based LoD generation is enabled.

6.6.2.28.4 Test bitstream DecLOD_D_LGE

Specification: The bitstream uses Periodic subsampling to generate detail levels levels in LoD with Lifting Transform. The bitstream exercises on range of sampling period.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

— lod_decimation_mode is equal to 1 (Periodic subsampling).

— lod_sampling_period_minus2 is equal to 0.

Functional stage: Attribute coding in Main profile and Dense profile.

Purpose: Check that the decoder can properly decode bitstreams which the periodic subsampling-based LoD generation is enabled.

6.6.2.28.5 Test bitstream DecLOD_E_LGE

Specification: The bitstream uses Periodic subsampling to generate detail levels in LoD with Lifting Transform. The bitstream exercises on range of sampling period.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

— lod_decimation_mode is equal to 1 (Periodic subsampling).

— lod_sampling_period_minus2 is equal to 2.

Functional stage: Attribute coding in Simple profile.

Purpose: Check that the decoder can properly decode bitstreams which the periodic subsampling-based LoD generation is enabled.

6.6.2.28.6 Test bitstream DecLOD_F_LGE

Specification: The bitstream uses Periodic subsampling to generate detail levels levels in LoD with Lifting Transform. The bitstream exercises on range of sampling period.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

— lod_decimation_mode is equal to 1 (Periodic subsampling).

— lod_sampling_period_minus2 is equal to 2.

Functional stage: Attribute coding in Main profile and Dense profile.

Purpose: Check that the decoder can properly decode bitstreams which the periodic subsampling-based LoD generation is enabled.

6.6.2.28.7 Test bitstream DecLOD_G_LGE

Specification: The bitstream uses Periodic subsampling to generate detail levels in LoD with Lifting Transform. The bitstream exercises on range of sampling period.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

— lod_decimation_mode is equal to 1 (Periodic subsampling).

— lod_sampling_period_minus2 is equal to 6.

Functional stage: Attribute coding in Simple profile.

Purpose: Check that the decoder can properly decode bitstreams which the periodic subsampling-based LoD generation is enabled.

6.6.2.28.8 Test bitstream DecLOD_H_LGE

Specification: The bitstream uses Periodic subsampling to generate detail levels levels in LoD with Lifting Transform. The bitstream exercises on range of sampling period.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

— lod_decimation_mode is equal to 1 (Periodic subsampling).

— lod_sampling_period_minus2 is equal to 6.

Functional stage: Attribute coding in Main profile and Dense profile.

Purpose: Check that the decoder can properly decode bitstreams which the periodic subsampling-based LoD generation is enabled.

6.6.2.29 LoD method: Distance (DisLOD)

6.6.2.29.1 Test bitstream DisLOD_A_XidianUniv

Specification: The bitstream tests distance-based decimation method used to generate detail levels. The bitstream exercises LOD initial distance and slice LOD distance offset in LoD with Lifting Transform. LOD initial distance and slice LOD distance offset presence equal to 0.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

— lod_decimation_mode is equal to 0 (No decimation).

— lod_max_levels_minus1 is equal to 11.

— lod_initial_dist_log2 is equal to 0.

— lod_dist_log2_offset_present is equal to 0.

— aps_scalable_enable_flag is equal to 0.

Functional stage: Attribute coding in Dense profile.

Purpose: Check that the decoder can properly decode bitstreams in which distance-based LoD generation method is enabled.

6.6.2.29.2 Test bitstream DisLOD_B_XidianUniv

Specification: The bitstream tests distance-based decimation method used to generate detail levels. The bitstream exercises on range of LOD initial distance and slice LOD distance offset in LoD with Lifting Transform. LOD initial distance sets to 6 and slice LOD distance offset presence equal to off.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

— lod_decimation_mode is equal to 0 (No decimation).

— lod_max_levels_minus1 is equal to 11.

— lod_initial_dist_log2 is equal to 6.

— lod_dist_log2_offset_present is equal to 0.

— aps_scalable_enable_flag is equal to 0.

Functional stage: Attribute coding in Dense profile.

Purpose: Check that the decoder can properly decode bitstreams in which distance-based LoD generation method is enabled.

6.6.2.29.3 Test bitstream DisLOD_C_XidianUniv

Specification: The bitstream tests distance-based decimation method used to generate detail levels. The bitstream exercises on range of LOD initial distance and slice LOD distance offset in LoD with Lifting Transform. LOD initial distance sets to 12 and slice LOD distance offset presence equal to off.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

— lod_decimation_mode is equal to 0 (No decimation).

— lod_max_levels_minus1 is equal to 11.

— lod_initial_dist_log2 is equal to 12.

— lod_dist_log2_offset_present is equal to 0.

— aps_scalable_enable_flag is equal to 0.

Functional stage: Attribute coding in Dense profile.

Purpose: Check that the decoder can properly decode bitstreams in which distance-based LoD generation method is enabled.

6.6.2.29.4 Test bitstream DisLOD_D_XidianUniv

Specification: The bitstream tests distance-based decimation method used to generate detail levels. The bitstream tests single level of details in LoD with Lifting Transform. The bitstream exercises on range of LOD initial distance and slice LOD distance offset. LOD initial distance sets and slice LOD distance offset presence to 0.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

— lod_decimation_mode is equal to 0 (No decimation).

— lod_max_levels_minus1 is equal to 0.

— lod_initial_dist_log2 is equal to 0.

— lod_dist_log2_offset_present is equal to 0.

— aps_scalable_enable_flag is equal to 0.

Functional stage: Attribute coding in Dense profile.

Purpose: Check that the decoder can properly decode bitstreams in which distance-based LoD generation method is enabled.

6.6.2.29.5 Test bitstream DisLOD_E_XidianUniv

Specification: The bitstream tests distance-based decimation method used to generate detail levels. The bitstream tests single level of details in LoD with Lifting Transform. The bitstream exercises on range of LoD initial distance and slice LoD distance offset. LoD initial distance sets to 6 and slice LoD distance offset presence equal to off.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

— lod_decimation_mode is equal to 0 (No decimation).

— lod_max_levels_minus1 is equal to 0.

— lod_initial_dist_log2 is equal to 6.

— lod_dist_log2_offset_present is equal to 0.

— aps_scalable_enable_flag is equal to 0.

Functional stage: Attribute coding in Dense profile.

Purpose: Check that the decoder can properly decode bitstreams in which distance-based LoD generation method is enabled.

6.6.2.29.6 Test bitstream DisLOD_F_XidianUniv

Specification: The bitstream tests distance-based decimation method used to generate detail levels. The bitstream tests single level of details in LoD with Lifting Transform. The bitstream exercises on range of LoD initial distance and slice LoD distance offset. LoD initial distance sets to 12 and slice LoD distance offset presence equal to off.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

— lod_decimation_mode is equal to 0 (No decimation).

— lod_max_levels_minus1 is equal to 0.

— lod_initial_dist_log2 is equal to 12.

— lod_dist_log2_offset_present is equal to 0.

— aps_scalable_enable_flag is equal to 0.

Functional stage: Attribute coding in Dense profile.

Purpose: Check that the decoder can properly decode bitstreams in which distance-based LoD generation method is enabled.

6.6.2.29.7 Test bitstream DisLOD_G_XidianUniv

Specification: The bitstream tests distance-based decimation method used to generate detail levels. The bitstream tests single level of details in LoD with Lifting Transform. The bitstream exercises on range of LoD initial distance and slice LoD distance offset with scalable attribute coding enabled. LoD initial distance and slice LoD distance offset presence equal to off.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

— lod_decimation_mode is equal to 0 (No decimation).

— lod_max_levels_minus1 is equal to 11.

— lod_initial_dist_log2 is equal to 0.

— lod_dist_log2_offset_present is equal to 0.

— aps_scalable_enable_flag is equal to 1.

— geom_qp_mul_log2 is equal to 3.

Functional stage: Attribute coding in Dense profile.

Purpose: Check that the decoder can properly decode bitstreams in which distance-based LoD generation method is enabled.

6.6.2.29.8 Test bitstream DisLOD_H_XidianUniv

Specification: The bitstream tests distance-based decimation method used to generate detail levels. The bitstream tests single level of details in LoD with Lifting Transform. The bitstream exercises on range of LoD initial distance and slice LoD distance offset with scalable attribute coding enabled. LoD initial distance sets to 6 and slice LoD distance offset presence equal to off.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

— lod_decimation_mode is equal to 0 (No decimation).

— lod_max_levels_minus1 is equal to 11.

— lod_initial_dist_log2 is equal to 6.

— lod_dist_log2_offset_present is equal to 0.

— aps_scalable_enable_flag is equal to 1.

— geom_qp_mul_log2 is equal to 3.

Functional stage: Attribute coding in Dense profile.

Purpose: Check that the decoder can properly decode bitstreams in which distance-based LoD generation method is enabled.

6.6.2.29.9 Test bitstream DisLOD_I_XidianUniv

Specification: The bitstream tests distance-based decimation method used to generate detail levels. The bitstream tests single level of details in LoD with Lifting Transform. The bitstream exercises on range of LoD initial distance and slice LoD distance offset with scalable attribute coding enabled. LoD initial distance sets to 12 and slice LoD distance offset presence equal to off.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

— lod_decimation_mode is equal to 0 (No decimation).

— lod_max_levels_minus1 is equal to 11.

— lod_initial_dist_log2 is equal to 12.

— lod_dist_log2_offset_present is equal to 0.

— aps_scalable_enable_flag is equal to 1.

— geom_qp_mul_log2 is equal to 3.

Functional stage: Attribute coding in Dense profile.

Purpose: Check that the decoder can properly decode bitstreams in which distance-based LoD generation method is enabled.

6.6.2.30 LoD method: Block-based (CentLOD)

6.6.2.30.1 Test bitstream CentLOD_A_LGE

Specification: The bitstream uses block-based LOD generation with scalable attribute coding enabled. When scalable attribute coding is enabled, block-based LOD generation shall be performed. The bitstream tests with scalable attribute coding enabled.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

— lod_scalability_enabled equal to 1.

Functional stage: Attribute coding in Main profile and Dense profile.

Purpose: Check that the decoder can properly decode bitstreams in which scalable attribute coding is enabled.

6.6.2.30.2 Test bitstream CentLOD _B_LGE

Specification: The bitstream tests block-based LOD generation with scalable attribute coding disabled.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

— lod_scalability_enabled equal to 0.

— lod_decimation_mode equal to 2.

— lod_initial_dist_log2 equal to 0.

— lod_sampling_period_minus2 equal to 2.

Functional stage: Attribute coding in Simple profile.

Purpose: Check that the decoder can properly decode bitstreams in which block-based LOD generation is enabled.

6.6.2.30.3 Test bitstream CentLOD _C_LGE

Specification: The bitstream tests block-based LOD generation with scalable attribute coding disabled.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

— lod_scalability_enabled equal to 0.

— lod_decimation_mode equal to 2.

— lod_initial_dist_log2 equal to 0.

— lod_sampling_period_minus2 equal to 2.

Functional stage: Attribute coding in Main profile and Dense profile.

Purpose: Check that the decoder can properly decode bitstreams in which block-based LOD generation is enabled.

6.6.2.30.4 Test bitstream CentLOD _D_LGE

Specification: The bitstream tests block-based LOD generation with initial spatial subsampling factor. The bitstream exercises on range of LoD initial distance with scalable attribute coding disabled. LoD initial distance sets to 6 and sampling period is equal to 4.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

— lod_scalability_enabled equal to 0.

— lod_decimation_mode equal to 2.

— lod_initial_dist_log2 equal to 6.

— lod_sampling_period_minus2 equal to 2.

Functional stage: Attribute coding in Simple profile.

Purpose: Check that the decoder can properly decode bitstreams in which block-based LOD generation is enabled.

6.6.2.30.5 Test bitstream CentLOD _E_LGE

Specification: The bitstream tests block-based LOD generation with initial spatial subsampling factor. The bitstream exercises on range of LoD initial distance with scalable attribute coding disabled. LoD initial distance sets to 6 and sampling period is equal to 4.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

— lod_scalability_enabled equal to 0.

— lod_decimation_mode equal to 2.

— lod_initial_dist_log2 equal to 6.

— lod_sampling_period_minus2 equal to 2.

Functional stage: Attribute coding in Main profile and Dense profile.

Purpose: Check that the decoder can properly decode bitstreams in which block-based LOD generation is enabled.

6.6.2.30.6 Test bitstream CentLOD_F_LGE

Specification: The bitstream tests block-based LoD generation with initial spatial subsampling factor. The bitstream exercises on range of LoD initial distance with scalable attribute coding disabled. LoD initial distance sets to 12 and sampling period is equal to 4.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

— lod_scalability_enabled is equal to 0.

— lod_decimation_mode is equal to 2.

— lod_initial_dist_log2 is equal to 12.

— lod_sampling_period_minus2 is equal to 2.

Functional stage: Attribute coding in Simple profile.

Purpose: Check that the decoder can properly decode bitstreams in which block-based LoD generation is enabled.

6.6.2.30.7 Test bitstream CentLOD_G_LGE

Specification: The bitstream tests block-based LoD generation with initial spatial subsampling factor. The bitstream exercises on range of LoD initial distance with scalable attribute coding disabled. LoD initial distance sets to 12 and sampling period is equal to 4.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

— lod_scalability_enabled is equal to 0.

— lod_decimation_mode is equal to 2.

— lod_initial_dist_log2 is equal to 12.

— lod_sampling_period_minus2 is equal to 2.

Functional stage: Attribute coding in Main profile and Dense profile.

Purpose: Check that the decoder can properly decode bitstreams in which block-based LoD generation is enabled.

6.6.2.30.8 Test bitstream CentLOD_H_LGE

Specification: The bitstream tests block-based LOD generation with sampling period. The bitstream exercises on range of sampling period with scalable attribute coding disabled. Sampling period is equal to 2.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

— lod_scalability_enabled equal to 0.

— lod_decimation_mode equal to 2.

— lod_initial_dist_log2 equal to 0.

— lod_sampling_period_minus2 equal to 0.

Functional stage: Attribute coding in Simple profile.

Purpose: Check that the decoder can properly decode bitstreams in which block-based LOD generation is enabled.

6.6.2.30.9 Test bitstream CentLOD_I_LGE

Specification: The bitstream tests block-based LOD generation with sampling period. The bitstream exercises on range of sampling period with scalable attribute coding disabled. Sampling period is equal to 2.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

— lod_scalability_enabled equal to 0.

— lod_decimation_mode equal to 2.

— lod_initial_dist_log2 equal to 0.

— lod_sampling_period_minus2 equal to 0.

Functional stage: Attribute coding in Main profile and Dense profile.

Purpose: Check that the decoder can properly decode bitstreams in which block-based LOD generation is enabled.

6.6.2.30.10 Test bitstream CentLOD_J_LGE

Specification: The bitstream tests block-based LOD generation with sampling period. The bitstream exercises on range of sampling period with scalable attribute coding disabled. Sampling period is equal to 8.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

— lod_scalability_enabled equal to 0.

— lod_decimation_mode equal to 2.

— lod_initial_dist_log2 equal to 0.

— lod_sampling_period_minus2 equal to 6.

Functional stage: Attribute coding in Simple profile.

Purpose: Check that the decoder can properly decode bitstreams in which block-based LOD generation is enabled.

6.6.2.30.11 Test bitstream CentLOD_K_LGE

Specification: The bitstream tests block-based LOD generation with sampling period. The bitstream exercises on range of sampling period with scalable attribute coding disabled. Sampling period is equal to 8.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

— lod_scalability_enabled equal to 0.

— lod_decimation_mode equal to 2.

— lod_initial_dist_log2 equal to 0.

— lod_sampling_period_minus2 equal to 6.

Functional stage: Attribute coding in Main profile and Dense profile.

Purpose: Check that the decoder can properly decode bitstreams in which block-based LOD generation is enabled.

6.6.2.31 Scalable Lifting (SL)

6.6.2.31.1 Test bitstream SL_A_Sony

Specification: The bitstream exercises on scalable lifting coding enabled.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

— lod_scalability_enabled equal to 1.

Functional stage: Attribute coding in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which scalable attribute coding is enabled.

6.6.2.31.2 Test bitstream SL_A_LGE

Specification: The bitstream tests with scalable attribute coding enabled.

— attr_coding_type is equal to 2 (LoD with Lifting Transform).

— lod_scalability_enabled equal to 1.

Functional stage: Attribute coding in Main profile and Dense profile.

Purpose: Check that the decoder can properly decode bitstreams in which scalable attribute coding is enabled.

6.6.2.32 Predictor Search-Inter LOD (PSInterLOD)

6.6.2.32.1 Test bitstream PSInterLOD_A_LGE

Specification: The bitstream uses inter LoD predictor search. The bitstream tests with full search range to determine nearest neighbours for prediction between detail levels.

— pred_inter_lod_search_range equal to -1, which means full range search enabled

— pred_intra_lod_search_range equal to 0.

Functional stage: Attribute coding in Simple profile.

Purpose: Check that the decoder can properly decode bitstreams in which inter LoD predictor search is enabled.

6.6.2.32.2 Test bitstream PSInterLOD_B_LGE

Specification: The bitstream uses inter LoD predictor search. The bitstream tests with full search range to determine nearest neighbours for prediction between detail levels.

— pred_inter_lod_search_range equal to -1, which means full range search enabled

— pred_intra_lod_search_range equal to 0.

Functional stage: Attribute coding in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which inter LoD predictor search is enabled.

6.6.2.32.3 Test bitstream PSInterLOD_C_LGE

Specification: The bitstream exercises inter LoD predictor search range.

— pred_inter_lod_search_range equal to 60000.

— pred_intra_lod_search_range equal to 0.

Functional stage: Attribute coding in Simple profile.

Purpose: Check that the decoder can properly decode bitstreams in which inter LoD predictor search is enabled.

6.6.2.32.4 Test bitstream PSInterLOD_D_LGE

Specification: The bitstream exercises inter LoD predictor search range.

— pred_inter_lod_search_range equal to 60000.

— pred_intra_lod_search_range equal to 0.

Functional stage: Attribute coding in Main profile.

Purpose: Check that the decoder can properly decode bitstreams in which inter LoD predictor search is enabled.

6.6.2.32.5 Test bitstream PSInterLOD_E_LGE

Specification: The bitstream uses inter LoD predictor search with scalable attribute coding enabled. The bitstream tests with scalable attribute coding enabled. When scalable attribute coding is enabled, maximum nearest neighbour range used to limit the distance of the point registered as neighbour is specified. Maximum nearest neighbour range is equal to 5.

— lod_scalability_enabled equal to 1.

— pred_max_range_minus1 equal to 4.

— pred_inter_lod_search_range equal to -1, which means full range search enabled.

— pred_intra_lod_search_range equal to 0.

Functional stage: Attribute coding in Dense profile.

Purpose: Check that the decoder can properly decode bitstreams in which inter LoD predictor search and scalable attribute coding is enabled.

6.6.2.32.6 Test bitstream PSInterLOD_F_LGE

Specification: The bitstream uses inter LoD predictor search with scalable attribute coding enabled. The bitstream exercises maximum nearest neighbour range when scalable attribute coding is enabled. Maximum nearest neighbour range is equal to 1.

— lod_scalability_enabled equal to 1.

— pred_max_range_minus1 equal to 0.

— pred_inter_lod_search_range equal to -1, which means full range search enabled.

— pred_intra_lod_search_range equal to 0.

Functional stage: Attribute coding in Dense profile.

Purpose: Check that the decoder can properly decode bitstreams in which inter LoD predictor search and scalable attribute coding is enabled.

6.6.2.32.7 Test bitstream PSInterLOD_G_LGE

Specification: The bitstream uses inter LoD predictor search with scalable attribute coding enabled. The bitstream exercises maximum nearest neighbour range when scalable attribute coding is enabled. Maximum nearest neighbour range is equal to 8.

— lod_scalability_enabled equal to 1.

— pred_max_range_minus1 equal to 7.

— pred_inter_lod_search_range equal to -1, which means full range search enabled.

— pred_intra_lod_search_range equal to 0.

Functional stage: Attribute coding in Dense profile.

Purpose: Check that the decoder can properly decode bitstreams in which inter LoD predictor search and scalable attribute coding is enabled.

6.6.2.33 Predictor Search-Intra LOD (PSIntraLOD)

6.6.2.33.1 Test bitstream PSIntraLOD_A_Panasonic

Specification: The bitstream test (pred_intra_min_lod=0, which means intra prediction on for all layers) and exercise range of pred_intra_lod_search_range with lod_scalability_enabled off.

— pred_intra_min_lod equal to 0.

— pred_intra_lod_search_range equal to 2.

— lod_scalability_enabled equal to 0.

Functional stage: Attribute coding in Dense profile.

Purpose: Check that the decoder can properly decode bitstreams which is used predictor search for intra LOD.

6.6.2.33.2 Test bitstream PSIntraLOD_B_Panasonic

Specification: The bitstream test (pred_intra_min_lod=0, which means intra prediction on for all layers) and exercise range of pred_intra_lod_search_range with lod_scalability_enabled on.

— pred_intra_min_lod equal to 0.

— pred_intra_lod_search_range equal to 154716 (for selected test stream).

— lod_scalability_enabled equal to 1.

Functional stage: Attribute coding in Dense profile.

Purpose: Check that the decoder can properly decode bitstreams which is used predictor search for intra LOD.

6.6.2.33.3 Test bitstream PSIntraLOD_C_Panasonic

Specification: The bitstream test (pred_intra_min_lod >0) and exercise range of pred_intra_lod_search_range with lod_scalability_enabled off.

— pred_intra_min_lod equal to 1.

— pred_intra_lod_search_range equal to 2.

— lod_scalability_enabled equal to 0.

Functional stage: Attribute coding in Dense profile.

Purpose: Check that the decoder can properly decode bitstreams which is used predictor search for intra LOD.

6.6.2.33.4 Test bitstream PSIntraLOD_D_Panasonic

Specification: The bitstream test (pred_intra_min_lod >0) and exercise range of pred_intra_lod_search_range with lod_scalability_enabled on.

— pred_intra_min_lod equal to 2.

— pred_intra_lod_search_range equal to 128.

— lod_scalability_enabled equal to 1.

Functional stage: Attribute coding in Dense profile.

Purpose: Check that the decoder can properly decode bitstreams which is used predictor search for intra LOD.

6.6.2.33.5 Test bitstream PSIntraLOD_E_Panasonic

Specification: The bitstream test (pred_intra_min_lod >0) and exercise range of pred_intra_lod_search_range with lod_scalability_enabled off.

— pred_intra_min_lod equal to 4.

— pred_intra_lod_search_range equal to 2.

— lod_scalability_enabled equal to 0.

Functional stage: Attribute coding in Dense profile.

Purpose: Check that the decoder can properly decode bitstreams which is used predictor search for intra LOD.

6.6.2.33.6 Test bitstream PSIntraLOD_F_Panasonic

Specification: The bitstream test (pred_intra_min_lod >0) and exercise range of pred_intra_lod_search_range with lod_scalability_enabled on.

— pred_intra_min_lod equal to 8.

— pred_intra_lod_search_range equal to 154716 (for selected test stream).

— lod_scalability_enabled equal to 1.

Functional stage: Attribute coding in Dense profile.

Purpose: Check that the decoder can properly decode bitstreams which is used predictor search for intra LOD.

6.6.2.34 Number of Predictor (NumPRED)

6.6.2.34.1 Test bitstream NumPRED_A_Panasonic

Specification: The bitstream tests number of predictors. The bitstream exercises on range of pred_set_size_minus1.

— pred_set_size_minus1 equal to 0.

Functional stage: Attribute coding in Main profile and Dense profile.

Purpose: Check that the decoder can properly decode bitstreams in which number of predictors is exercised.