INTERNATIONAL STANDARD

ISO/IEC 23090-15

First edition
2022-11

Information technology — Coded representation of immersive media —

Part 15: Conformance testing for versatile video coding

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

Partie 15: Essai de conformité pour le codage vidéo polyvalent

Reference number
ISO/IEC 23090-15:2022(E)

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).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see https://patents.iec.ch).

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*, in collaboration with ITU-T Study Group 16 (as Rec. ITU-T H.266.1).

A list of all parts in the ISO/IEC 23090 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.

Information technology — Coded representation of immersive media —

Part 15: Conformance testing for versatile video coding

1 Scope

This document specifies a set of tests and procedures designed to indicate whether encoders or decoders meet the requirements specified in Rec. ITU-T H.266 | ISO/IEC 23090-3.

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.

Rec. ITU-T H.266:2020 | ISO/IEC 23090-3:2021, *Information technology – Coded representation of immersive media– Part 3: Versatile video coding*

Rec. ITU-T H.266.2 | ISO/IEC 23090-16, *Information technology – Coded representation of immersive media – Part 16: Reference software for versatile video coding*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in Rec. ITU-T H.266 | ISO/IEC 23090-3 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 that conforms to specified syntax requirements or sequence of bits to be tested for conformance to such syntax requirements

3.2
decoder
embodiment of a specified decoding process or process to be tested for conformance to such a decoding process specification

3.3
encoder
embodiment of a process that produces a *bitstream* (3.1)

3.4
reference software decoder
particular *decoder* (3.2) provided as a software package for use as an example available for study, as a potential starting basis for the development of other decoders, as a way of testing *bitstreams* (3.1) for conformance to a decoding process specification, or as a reference for comparison with the behaviour of other decoders

3.5
reference software encoder
particular *encoder* (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 encoders, or as a reference for comparison with the behaviour of other encoders

4 Abbreviated terms

AMVP Adaptive motion vector prediction

CCLM Cross-component linear model

CIIP Combined inter/intra prediction

CST Chroma separate tree

CTC Common test conditions

DCT Discrete cosine transform

DMVR Decoder-side motion vector refinement

DQ Dependent quantization

DST Discrete sine transform

FTP File transfer protocol

ISP Intra subblock partitioning

JCCR Joint coding of chroma residuals

MMVD Merge with MVD

MPM Most probable mode

MRL Multiple reference line

AMVP Adaptive motion vector prediction

CCLM Cross-component linear model

CIIP Combined inter/intra prediction

CST Chroma separate tree

CTC Common test conditions

DCT Discrete cosine transform

DMVR Decoder-side motion vector refinement

DQ Dependent quantization

DST	Discrete sine transform
FTP	File transfer protocol
ISP	Intra subblock partitioning
JCCR	Joint coding of chroma residuals
MMVD	Merge with MVD
MPM	Most probable mode
MRL	Multiple reference line
MVD	Motion vector difference
NUT	NAL unit type
PDPC	Position-dependent (intra) prediction combination
PERP	Padded equirectangular projection
RPR	Reference picture resampling
SAD	Sum of absolute differences
SBT	Subblock transform
SCC	Screen content coding
SbTMVP	Subblock based temporal motion vector prediction.
SMVD	Symmetric MVD
TMVP	Temporal motion vector prediction.

5 Conventions

The conventions in Clause 5 of Rec. ITU-T H.266:2020 | ISO/IEC 23090-3:2021 apply.

6 Conformance testing for ITU-T H.266 | ISO/IEC 23090-3

6.1 General

The conformance testing data for Rec. ITU-T H.266 | ISO/IEC 23090-3 is found in the electronic attachment that can be obtained at the following location:

https://standards.iso.org/iso-iec/23090/-15/ed-1/en/

The following subclauses specify normative tests for verifying conformance of video bitstreams as well as decoders. Those normative tests make use of test data (bitstream test suites) provided as an electronic attachment to this document and the reference software decoder specified in Rec. ITU-T H.266.2 | ISO/IEC 23090-16.

6.2 Bitstream conformance

Bitstream conformance for Rec. ITU-T H.266 | ISO/IEC 23090-3 is specified by clause C.4 of Rec. ITU-T H.266:2020 | ISO/IEC 23090-3:2021.

6.3 Decoder conformance

Decoder conformance for Rec. ITU-T H.266 | ISO/IEC 23090-3 is specified by clause C.5 of Rec. ITU-T H.266:2020 | ISO/IEC 23090-3:2021.

6.4 Procedure to test bitstreams

A bitstream that is claimed to conform to Rec. ITU-T H.266 | ISO/IEC 23090-3 shall pass the following normative test. 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 Rec. ITU-T H.266 | ISO/IEC 23090-3.

The bitstream under test shall be decoded by processing it with the reference software decoder specified in Rec. ITU-T H.266.2 | ISO/IEC 23090-16. 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. When the bitstream under test contains decoded picture hash SEI messages, the hash values signalled in the decoded picture hash SEI messages in the bitstream shall match those calculated by the reference software decoder.

Successfully passing this test provides only a strong presumption that the bitstream under test does indeed meet all the requirements specified in Rec. ITU-T H.266 | ISO/IEC 23090-3 that are tested by the reference software decoder.

Additional tests may be necessary to more thoroughly check that the bitstream properly meets all the requirements specified in Rec. ITU-T H.266 | ISO/IEC 23090-3, including hypothetical reference decoder (HRD) conformance (based on Annexes C and D). Such complementary tests may be performed using other video bitstream verifiers that perform more complete tests than those implemented by the reference software decoder.

Rec. ITU-T H.266 | ISO/IEC 23090-3 contains several informative recommendations that are not an integral part of that Recommendation | International Standard. When testing a bitstream for conformance, it may also be useful to test whether or not the bitstream follows those recommendations.

To check the 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 Rec. ITU-T H.266 | ISO/IEC 23090-3.

A bitstream verifier may not necessarily perform all stages of the decoding process specified in Rec. ITU-T H.266 | ISO/IEC 23090-3 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 that conforms to Rec. ITU-T H.266 | ISO/IEC 23090-3 has values of general_profile_idc, general_tier_flag, and general_level_idc corresponding to a set of specified constraints on a bitstream for which a decoder conforming to a corresponding specified profile, tier, and level is required in Annex A of Rec. ITU-T H.266:2020 | ISO/IEC 23090-3:2021 to properly perform the decoding process.

6.5.2 Contents of the bitstream file

The associated conformance testing bitstreams are included with this document as an electronic attachment. The following information is included in a single zipped file for each such bitstream.

— *.bit – bitstream (provided for all bitstreams)

— *.txt – description (provided for all bitstreams)

— *.yuv.md5 – MD5 checksum of the complete decoded yuv file (provided for all bitstreams)

— *.md5 – MD5 checksum of the bitstream file (provided for all bitstreams)

— *.opl – output picture log (provided for all bitstreams)

— *.cfg – config file used to generate bitstream with VTM encoder software (not provided for all bitstreams, not applicable if a VTM encoder release version was not used)

6.5.3 Requirements on output of the decoding process and timing

Two classes of decoder conformance are specified:

— output order conformance; and

— output timing conformance.

The output of the decoding process is specified in clause 8 and Annex C of Rec. ITU-T H.266:2020 | ISO/IEC 23090-3:2021.

For output order conformance, it is a requirement that all of the cropped decoded pictures specified for output in Annex C of Rec. ITU-T H.266:2020 | ISO/IEC 23090-3:2021 shall be output by a conforming decoder in the specified order and that the values of the decoded samples of the cropped decoded pictures that are output shall be (exactly equal to) the values specified in clause 8 of Rec. ITU-T H.266:2020 | ISO/IEC 23090-3:2021.

For output timing conformance, it is a requirement that a conforming decoder shall also output the cropped decoded pictures at the picture rates and times specified in Annex C of Rec. ITU-T H.266:2020 | ISO/IEC 23090-3:2021.

The display process, which ordinarily follows the output of the decoding process, is outside the scope of this document.

6.5.4 Static tests for output order conformance

Static tests of a video decoder require testing of the samples of the cropped decoded pictures that are output from the decoder, and can be accomplished when the decoded samples at the output of the decoding process are available. It may not be possible to perform this type of test with a production decoder (due to the lack of an appropriate accessible interface in the design at which to perform the test). In such a case this test should be performed by the manufacturer during the design and development phase. Static tests are used for testing the decoding process.

The pictures that are output by the decoder under test are checked to ensure that the following requirements are fulfilled:

— The cropped decoded pictures that are output by the decoder under test shall correspond to those that are output by the reference software decoder.

— The cropped decoded pictures that are output by the decoder under test shall be output in the same order as those that are output by the reference software decoder.

— The values of the samples of the cropped decoded pictures that are output by the decoder under test shall be identical to those that are output by the reference software decoder.

To assist with the checking of the decoding process and the cropped decoded pictures, hash values for the cropped decoded pictures that are output by conforming decoders are provided in a corresponding output picture log file for each test bitstream that is used in the specified conformance tests, and most of these test bitstreams also contain decoded picture hash SEI messages that may be used for checking the results of the decoding process of the decoder under test.

6.5.5 Dynamic tests for output timing conformance

Dynamic tests are applied to check that all the decoded samples of the cropped decoded pictures are output and that the timing of the output of the decoder's decoded samples conforms to the specifications of Clause 8 and Annex C of Rec. ITU-T H.266:2020 | ISO/IEC 23090-3:2021, and to verify that the decoder under test can operate according to bitstream flow characteristics prescribed by the specified HRD models (as specified by the CPB and DPB specification in Annex C of Rec. ITU-T H.266:2020 | ISO/IEC 23090-3:2021) when the bits of the bitstream are delivered at the proper rate.

The dynamic test is often easier to perform on a complete decoding system, which may include a systems decoder, a video decoder and a display process. It may be possible to record the output of the display process and to check that display order and timing of the cropped decoded pictures are correct at the output of the display process. However, since the display process is not within the normative scope of Rec. ITU-T H.266 | ISO/IEC 23090-3, there may be cases where the output of the display process differs in timing or value even though the video decoder is conforming. In this case, the output of the video decoder itself (before the display process) would need to be captured in order to perform the dynamic tests on the video decoder. In particular the output order and timing of the output of the cropped decoded pictures shall be correct.

If buffering period and picture timing SEI messages are included in the test bitstream, HRD conformance shall be verified using the values of nal_initial_cpb_removal_delay, nal_initial_cpb_removal_offset, au_cpb_removal_delay_minus1 and pic_dpb_output_delay that are included in the bitstream.

If buffering period and picture timing SEI messages are not included in the bitstream, the following inferences shall be made to generate the missing parameters:

— fixed_pic_rate_general_flag[i] shall be inferred to be equal to 1.

— low_delay_hrd_flag[i] shall be inferred to be equal to 0.

— cbr_flag[subLayerId][j] shall be inferred to be equal to 0.

— The frame rate of the bitstream shall be inferred to be equal to the frame rate value specified in the .txt file for the bitstream. If this is missing, then a frame rate of either 25 or 30000 ÷ 1001 can be inferred.

— The bit rate of the bitstream shall be inferred to be equal to the maximum value for the level specified in Table 136 in Rec. ITU-T H.266:2020 | ISO/IEC 23090-3:2021.

— CPB and DPB sizes shall be inferred to be equal to the maximum value for the level specified in Table 135 in Rec. ITU-T H.266:2020 | ISO/IEC 23090-3:2021.

With the above inferences, the HRD shall be operated as follows:

— The CPB is filled starting at time t = 0, until it is full, before removal of the first access unit. This means that the bp_nal_initial_cpb_removal_delay[i][j] shall be inferred to be equal to the total CPB buffer size divided by the bit rate divided by 90000 (rounded downwards) and bp_vcl_initial_cpb_removal_offset[i][j] shall be inferred to be equal to zero.

— The first access unit is removed at time t = bp_nal_initial_cpb_removal_delay[i][j] ÷ 90000 and subsequent access units are removed at intervals based on the picture distance.

— Using these inferences with the accompanying bitstreams, the CPB will not overflow or underflow and the DPB will not overflow.

6.5.6 Decoder conformance test for a particular profile, tier, and level

In order for a decoder for a particular profile, tier, and level to claim output order conformance to Rec. ITU-T H.266 | ISO/IEC 23090-3, the decoder shall successfully pass the static test specified in Clause 6.5.4 with all the bitstreams of the normative test suite specified for testing decoders of this particular profile, tier, and level combination.

In order for a decoder of a particular profile, tier, and level to claim output timing conformance to Rec. ITU-T H.266 | ISO/IEC 23090-3, the decoder shall successfully pass both the static test specified in clause 6.5.4 and the dynamic test specified in Clause 6.5.5 with all the bitstreams of the normative test suite specified for testing decoders of this particular profile, tier, and level.

Tables 1 through 10 specify the normative test suites. The profile, tier, and level combinations are described in the tables or in the .txt file associated with the bitstream.

6.6 Specification of the test bitstreams

6.6.1 General

Some characteristics of each bitstream listed in Table 1 are specified in this clause.

6.6.2 Test bitstreams – Coding tools for Main 10 profile with 4:2:0 chroma format and 10 bit depth

6.6.2.1 Chroma separate tree (CST)

6.6.2.1.1 Test bitstream CST_A

Specification: All pictures are coded in I slices with CST enabled. CST is tested with all possible luma and chroma block sizes, and luma-chroma block size combinations (e.g., luma block size is larger than, equal to, or smaller than the corresponding chroma block size).

Functional stage: Reconstruction process.

Purpose: Check that the decoder can properly decode slices with CST enabled.

6.6.2.2 Dependent quantization (DQ)

6.6.2.2.1 Test bitstream DQ_A

Specification: The bitstream consists of three CVSs, with the following properties:

— The first CVS uses dependent quantization for all pictures, all non-related features (inter tools, ALF, ...) are disabled, and MTS and LFNST are disabled.

— The second CVS uses dependent quantization for all pictures, all non-related features (inter tools, ALF, ...) are disabled, and MTS (for intra) and LFNST are enabled.

— The third CVS exercises a picture-level selection between dependent quantization, sign data hiding, and standard quantization, all non-related features (inter tools, ALF, ...) are disabled, and MTS (for intra) and LFNST are enabled.

Functional stage: Dependent quantization.

Purpose: Check that the decoder can properly decode slices with DQ enabled.

6.6.2.2.2 Test bitstream DQ_B

Specification: The bitstream consists of three CVSs of resolution 1920 x 1080, with the following properties:

— The first CVS uses dependent quantization for all pictures, all non-related features (inter tools, ALF, ...) are disabled, and MTS and LFNST are disabled.

— The second CVS uses dependent quantization for all pictures, all non-related features (inter tools, ALF, ...) are disabled, and MTS (for intra) and LFNST are enabled.

— The third CVS exercises a picture-level selection between dependent quantization, sign data hiding, and standard quantization, all non-related features (inter tools, ALF, ...) are disabled, and MTS (for intra) and LFNST are enabled.

Functional stage: Dependent quantization.

Purpose: Check that the decoder can properly decode slices with DQ enabled.

6.6.2.3 Cross-component linear model (CCLM)

6.6.2.3.1 Test bitstream CCLM_A

Specification: The bitstream exercises corner cases for coding structures using CCLM with the following properties:

— POC0: Chroma CU size is 64x64.

— POC1: First split of CU is horizontal, i.e. CU size is 64x32.

— POC2: First split of CU is quad, i.e. CU size is 32x32.

— POC3: First and second split of CU are horizontal and vertical, respectively.

— POC4: First split of CU is vertical or ternary, i.e. none of condition is satisfied for CCLM.

— POC5: CU size is 64x64 and ISP is enabled.

— POC6: First luma split is something else than quad.

Functional stage: Intra prediction.

Purpose: Check that the decoder can properly decode slices with CCLM enabled.

6.6.2.4 Multiple transform set (MTS)

6.6.2.4.1 Test bitstream MTS_A

Specification: The bitstream exercises the following transform features:

— 1st part

  — Explicit intra MTS on and explicit inter MTS off with low frequency non-separable transform (LFNST) disabled.

  — Include all test cases for ISP, MIP, luma tree, and CST.

  — Include all candidates of explicit MTS, i.e. DCT2-DCT2, DST7-DST7, DCT8-DST7, DST7-DCT8, and DCT8-DCT8.

  — Include all possible block sizes and partitions where all MTS combinations can happen.

— 2nd part

  — Implicit MTS on and explicit inter MTS off with LFNST disabled.

  — Include all test cases for ISP, MIP, luma tree, and CST.

  — Include all possible block sizes and partitions (especially for ISP) for all allowable MTS combinations.

Functional stage: Transform.

Purpose: Check that the decoder can properly decode slices with MTS enabled.

6.6.2.4.2 Test bitstream MTS_B

Specification: The bitstream exercises the following transform features:

— 1st part

  — Explicit intra MTS on and explicit inter MTS off with LFNST disabled.

  — Include all test cases for SBT, single tree and TU-tiling based on maximum transform size (64).

  — Include all candidates of explicit MTS, i.e., DCT2-DCT2, DST7-DST7, DCT8-DST7, DST7-DCT8, and DCT8-DCT8.

  — Include all possible block sizes and partitions (especially for SBT) where all MTS combinations can happen.

— 2nd part

  — Implicit intra MTS on and explicit inter MTS off with LFNST disabled.

  — Include all test cases for SBT, single tree, and TU-tiling based on maximum transform size (64).

  — Include all possible block sizes and partitions (especially for SBT) where all MTS combinations can happen.

— 3rd part

  — Implicit MTS on and explicit inter MTS off with LFNST disabled.

  — Include all test cases for SBT and single tree.

  — Include all possible block sizes and partitions (especially for SBT) where all MTS combinations can happen.

— 4th part

  — Explicit intra MTS on and explicit inter MTS on with LFNST disabled.

  — Include all test cases for SBT and single tree.

  — Include all possible block sizes and partitions (especially for SBT) where all MTS combinations can happen.

Functional stage: Transform.

Purpose: Check that the decoder can properly decode slices with MTS enabled.

6.6.2.5 Adaptive loop filter (ALF)

6.6.2.5.1 Test bitstream ALF_A

Specification: This bitstream uses both ALF and virtual boundary, as follows:

— Applies ALF virtual boundary (VB) at non-CTC CTU sizes (CTU size of 64 is used).

— Positions luma VB at 4 lines (Pos : 60) and chroma VB at 2 lines (Pos : 62) above the CTU height.

Functional stage: Adaptive loop filter.

Purpose: Check that the decoder can properly decode slices with ALF enabled.

6.6.2.5.2 Test bitstream ALF_B

Specification: This bitstream uses both ALF and virtual boundary, as follows:

— Applies ALF virtual boundary (VB) to sequences whose picture height is 1 CTU (CTU size of 128 is used as per CTC).

— Positions luma VB at 4 lines (Pos : 124) and chroma VB at 2 lines (Pos : 62) above the CTU height.

Functional stage: Adaptive loop filter.

Purpose: Check that the decoder can properly decode slices with ALF enabled.

6.6.2.5.3 Test bitstream ALF_C

Specification: Bitstream exercises clipping values of non-linear ALF.

Functional stage: Adaptive loop filter.

Purpose: Check that the decoder can properly decode slices with ALF enabled.

6.6.2.5.4 Test bitstream ALF_D

Specification: Bitstream uses multiple ALF APSs with LMCS enabled.

Functional stage: Adaptive loop filter.

Purpose: Check that the decoder can properly decode slices with ALF enabled.

6.6.2.6 Affine motion model (AFF)

6.6.2.6.1 Test bitstream AFF_A

Specification: The bitstream enables 6-parameter affine mode by SPS flag. All allowed blocks sizes of Affine merge mode are exercised multiple times. All allowed blocks sizes of Affine AMVP mode, including 4-parameter and 6-parameter Affine mode, are exercised multiple times. All allowed candidates for Affine merge mode, including two inherited candidates, four 6-parameter constructed candidates, two 4-parameter constructed candidates, and zero padding candidate are exercised multiple times. Inheritance of affine model from top CTU are exercised multiple times. Fallback mode for affine memory bandwidth restriction is triggered multiple times.

Functional stage: Affine mode inter prediction.

Purpose: Check that the decoder can properly decode slices with affine mode enabled.

6.6.2.6.2 Test bitstream AFF_B

Specification: The bitstream uses affine mode, with 6-parameter affine mode disabled by SPS flag. All allowed blocks sizes of Affine merge mode are exercised multiple times. All allowed blocks sizes of 4-parameter Affine AMVP mode are exercised multiple times. All allowed candidates for Affine merge mode, including two inherited candidates, two 4-parameter constructed candidates, and zero padding candidate are exercised multiple times. Inheritance of affine model from top CTU are exercised multiple times. Fallback mode for affine memory bandwidth restriction is triggered multiple times. All allowed blocks sizes of Affine merge mode are exercised multiple times. All allowed blocks sizes of 4-parameter Affine AMVP mode are exercised multiple times. All allowed candidates for Affine merge mode, including two inherited candidates, two 4-parameter constructed candidates, and zero padding candidate are exercised multiple times. Inheritance of affine model from top CTU are exercised multiple times. Fallback mode for affine memory bandwidth restriction is triggered multiple times.

Functional stage: Affine mode inter prediction.

Purpose: Check that the decoder can properly decode slices with affine mode enabled.

6.6.2.7 Subblock-based temporal merging candidates (SbTMVP)

6.6.2.7.1 Test bitstream SbTMVP_A

Specification: The bitstream uses SbTMVP when affine is disabled.

Functional stage: Inter prediction process.

Purpose: Check that the decoder can properly decode PUs with SbTMVP on and affine off.

6.6.2.7.2 Test bitstream SbTMVP_B

Specification: This bitstream disables SbTMVP.

Functional stage: Inter prediction process.

Purpose: Check that the decoder can properly decode PUs with SbTMVP off.

6.6.2.8 Adaptive motion vector resolution (AMVR)

6.6.2.8.1 Test bitstream AMVR_A

Specification: The bitstream exercises translational and affine AMVR with different settings. It represents a concatenation of 5 CVSs with the following properties:

— The first CVS exercises translational AMVR with amvr_precision_idx equal to 1 (i.e., 1 luma sample motion vector resolution).

— The second CVS exercises translational AMVR with amvr_precision_idx equal to 2 (i.e., 4 luma samples motion vector resolution).

— The third CVS exercises translational AMVR with amvr_precision_idx equal to 0 (i.e., 1/2 luma sample motion vector resolution). This implies application of the Switchable Interpolation Filter (SIF).

— The fourth CVS exercises affine AMVR with amvr_precision_idx equal to 0 (i.e., 1/16 luma sample motion vector resolution).

— The fifth CVS exercises affine AMVR with amvr_precision_idx equal to 1 (i.e., 1 luma sample motion vector resolution).

Functional stage: Inter prediction.

Purpose: Check that the decoder can properly decode bitstreams with AMVR enabled.

6.6.2.8.2 Test bitstream AMVR_B

Specification: The bitstream exercises AMVR. It cycles frame-by-frame between the following variants:

— Translational AMVR with amvr_precision_idx equal to 1 (i.e., 1 luma sample motion vector resolution).

— Translational AMVR with amvr_precision_idx equal to 2 (i.e., 4 luma samples motion vector resolution).

— Translational AMVR with amvr_precision_idx equal to 0 (i.e., 1/2 luma sample motion vector resolution), this implies application of the Switchable Interpolation Filter (SIF).

— Affine AMVR with amvr_precision_idx equal to 0 (i.e., 1/16 luma sample motion vector resolution).

— Affine AMVR with amvr_precision_idx equal to 1 (i.e., 1 luma sample motion vector resolution).

Functional stage: Inter prediction.

Purpose: Check that the decoder can properly decode bitstreams with AMVR enabled.

6.6.2.9 Bi-directional optical flow (BDOF)

6.6.2.9.1 Test bitstream BDOF_A

Specification: Bitstream exercises all possible implicit BDOF on/off conditions and subblock usages.

Functional stage: Inter prediction process.

Purpose: Check that the decoder can properly decode CUs with BDOF enabled.

6.6.2.10 Combined intra/inter prediction (CIIP)

6.6.2.10.1 Test bitstream CIIP_A

Specification: The bitstream exercises all possible inter direction, block sizes, and combining weights for CIIP.

Functional stage: Inter prediction process.

Purpose: Check that the decoder can properly decode CUs with CIIP enabled.

6.6.2.11 Merge with MVD (MMVD)

6.6.2.11.1 Test bitstream MMVD_A

Specification: The bitstream uses MMVD with different numbers of MMVD distance entries.

Functional stage: Inter prediction process.

Purpose: Check that the decoder can properly decode bitstreams with merge with MMVD enabled.

6.6.2.12 Bi-predictive with CU weights (BCW)

6.6.2.12.1 Test bitstream BCW_A

Specification: The bitstream exercises all possible combining weights for BCW.

Functional stage: Inter prediction process.

Purpose: Check that the decoder can properly decode CUs with BCW enabled.

6.6.2.13 Multi-reference line prediction (MRLP)

6.6.2.13.1 Test bitstream MRLP_A

Specification: The bitstream contains all possible combinations of extended intra reference lines for luma indicated by intra_luma_ref_idx = {1, 2} and most probable modes except the DC, indicated by intra_luma_mpm_idx ={0, 1, 2, 3, 4}. For the CUs at the top border of a CTU, extended references lines are not used in the MRL index is not present in the bitstream. All CTC tools are enabled.

Functional stage: Intra prediction and mode signalling processes.

Purpose: Test all combinations of reference line indices and associated MPM signalling with all tools on.

6.6.2.13.2 Test bitstream MRLP_B

Specification: The bitstream contains all possible combinations of extended intra reference lines for luma indicated by intra_luma_ref_idx = {1, 2} and most probable modes except the DC, indicated by intra_luma_mpm_idx = {0, 1, 2, 3, 4}. For the CUs at the top border of a CTU, extended references lines are not used in the MRL index is not present in the bitstream. The following prediction modes have been disabled: Intra: ISP and MIP, Inter: SBT, MMVD, Affine, SubPuMvp, IMV, BCW, BIO, CIIP, GPM, DisFracMMVD, AffineAmvr, DMVR, SMVD, and PROF.

Functional stage: Intra prediction and mode signalling processes.

Purpose: Test all combinations of reference line indices and associated MPM signalling with specific intra and inter tools turned off.

6.6.2.14 Intra block copy mode (IBC)

6.6.2.14.1 Test bitstream IBC_A

Specification: This bitstream exercises general IBC features, merge, skip and AMVP modes.

Functional stage: Intra prediction.

Purpose: Check that the decoder can properly decode bitstreams with IBC enabled.

6.6.2.14.2 Test bitstream IBC_B

Specification: This bitstream exercises general IBC features, merge, skip and AMVP modes, with BV predictor size equal to 1 (MaxNumIBCMergeCand=1).

Functional stage: Intra prediction.

Purpose: Check that the decoder can properly decode bitstreams with IBC enabled.

6.6.2.14.3 Test bitstream IBC_C

Specification: This bitstream exercises general IBC features, merge, skip and AMVP modes, with Dual Tree disabled (DualTree=0).

Functional stage: Intra prediction.

Purpose: Check that the decoder can properly decode bitstreams with IBC enabled.

6.6.2.14.4 Test bitstream IBC_D

Specification: This bitstream exercises general IBC features, merge, skip and AMVP modes, with AMVR disabled.

Functional stage: Intra prediction.

Purpose: Check that the decoder can properly decode bitstreams with IBC enabled.

6.6.2.14.5 Test bitstream IBC_E

Specification: This bitstream exercises general IBC blocks with all possible block sizes.

Functional stage: Intra prediction.

Purpose: Check that the decoder can properly decode bitstreams with IBC enabled.

6.6.2.15 Intra sub-partitioning (ISP)

6.6.2.15.1 Test bitstream ISP_A

Specification: The bitstream contains various combinations of block sizes, ISP split types, intra modes and LFNST indices. It uses an AI configuration and 34 frames with QP 28.

Functional stage: Intra prediction.

Purpose: Check that the decoder can properly decode bitstreams with ISP enabled.

6.6.2.15.2 Test bitstream ISP_B

Specification: The bitstream contains various combinations of block sizes, ISP split types, intra modes and LFNST indices. It uses a RA configuration and 161 frames with QP 34.

Functional stage: Intra prediction.

Purpose: Check that the decoder can properly decode bitstreams with ISP enabled.

6.6.2.16 Decoder motion vector refinement (DMVR)

6.6.2.16.1 Test bitstream DMVR_A

Specification: This bitstream exercises DMVR with the following features:

— All allowed blocks sizes of DMVR are exercised multiple times.

— Motion vector wraparound is enabled where DMVR uses wrapped around reference samples.

— BCW and explicit weighted bipredictionis turned on for luma and chroma components, to test disabling of DMVR.

— All integer delta motion vector combinations are exercised multiple times.

— All fractional delta MV are exercised multiple times.

Functional stage: Inter prediction.

Purpose: Check that the decoder can properly decode bitstreams with DMVR enabled.

6.6.2.16.2 Test bitstream DMVR_B

Specification: This bitstream exercises DMVR with corner cases of SAD variations.

Functional stage: Inter prediction process.

Purpose: Check that the decoder can properly decode bitstreams with DMVR enabled.

6.6.2.17 Sub-block transform (SBT)

6.6.2.17.1 Test bitstream SBT_A

Specification: The bitstream exercises SVT with all allowed blocks sizes exercised multiple times (at least once for each SBT mode for each allowed CU size).

Functional stage: Transform process.

Purpose: Check that the decoder can properly decode bitstreams with DMVR enabled.

6.6.2.18 Luma mapping with chroma scaling (LMCS)

6.6.2.18.1 Test bitstream LMCS_A

Specification: This bitstream tests control of LMCS at the slice level, with the picture split into 4 tiles and 4 rectangular slices.

Functional stage: In-loop filter process.

Purpose: Check that the decoder can properly decode bitstreams with LMCS enabled.

6.6.2.18.2 Test bitstream LMCS_B

Specification: This bitstream tests control of LMCS at the slice level, with the picture split into 8 rectangular slices, 12 tiles and 2 subpictures.

Functional stage: In-loop filter.

Purpose: Check that the decoder can properly decode bitstreams with LMCS enabled.

6.6.2.18.3 Test bitstream LMCS_C

Specification: This bitstream tests control of LMCS for the entire CVS according to NoLmcsConstraintFlag.

Functional stage: In-loop filter.

Purpose: Check that the decoder can properly decode bitstreams with LMCS enabled.

6.6.2.19 Sign data hiding (SDH)

6.6.2.19.1 Test bitstream SDH_A

Specification: This bitstream tests SDH on/off control at picture level.

Functional stage: In-loop filter.

Purpose: Check that the decoder can properly decode bitstreams with LMCS enabled.

6.6.2.20 Symmetric motion vector difference (SMVD)

6.6.2.20.1 Test bitstream SMVD_A

Specification: This bitstream exercises all allowed blocks sizes of SMVD mode multiple times.

Functional stage: Inter prediction.

Purpose: Check that the decoder can properly decode bitstreams with SMVD enabled.

6.6.2.21 Block-based delta pulse code modulation (BDPCM)

6.6.2.21.1 Test bitstream BDPCM_A

Specification: This bitstream exercises BDPCM-coded block of each possible block size, in both luma and chroma.

Functional stage: Intra coding.

Purpose: Check that the decoder can properly decode bitstreams with BDPCM enabled.

6.6.2.22 Matrix based intra prediction (MIP)

6.6.2.22.1 Test bitstream MIP_A

Specification: This bitstream exercises MIP in different combinations with other tools. The bitstream consists of three CVSs with the following properties:

— First CVS: For each M, N in {4, 8, 16, 32, 64}, the bitstream contains an MxN-luma-intra-block in which the intra-prediction signal is generated by a mip-mode and in which intra_mip_transposed_flag is false and it contains an MxN-luma-intra-block in which the intra-prediction signal is generated by a mip-mode and in which intra_mip_transposed_flag is true. For each mip-matrix occurring in the spec, the bitstream contains a luma-intra-block in which the intra-prediction signal is generated by a mip-mode that uses this mip-matrix.

— Second CVS: MIP is enabled, MTS and intra tools LFNST, ISP, MRL are disabled. Each slice is an intra slice.

— Third CVS: MIP is enabled, all other intra tools are enabled. Each slice is an intra slice.

Functional stage: Intra coding.

Purpose: Check that the decoder can properly decode bitstreams with MIP enabled.

6.6.2.22.2 Test bitstream MIP_B

Specification: This bitstream exercises MIP with all other CTC tools enabled.

Functional stage: Intra coding.

Purpose: Check that the decoder can properly decode bitstreams with MIP enabled.

6.6.2.23 Low frequency non-separable transform (LFNST)

6.6.2.23.1 Test bitstream LFNST_A

Specification: This bitstream exercises LFNST with MTS disabled, as follows:

— Include all test cases for ISP, MIP, luma tree, and CST with LFNST.

— Include all sets and candidates of LFNST, i.e. 4 sets and 2 candidates per set.

— Include all possible block sizes and partitions (especially for ISP) where LFNST can be applied.

Functional stage: Transform.

Purpose: Check that the decoder can properly decode bitstreams with LFNST enabled.

6.6.2.23.2 Test bitstream LFNST_B

Specification: This bitstream exercises LFNST with MTS disabled, as follows:

— Include all test cases for single tree with LFNST.

— Include all sets and candidates of LFNST, i.e. 4 sets and 2 candidates per set.

— Include all possible block sizes where LFNST can be applied.

Functional stage: Transform.

Purpose: Check that the decoder can properly decode bitstreams with LFNST enabled.

6.6.2.23.3 Test bitstream LFNST_C

Specification: This bitstream exercises LFNST and its signalling in different combination with other tools. The bitstream consists of four CVSs with the following properties:

— The first CVS enables LFNST and explicit MTS; ISP and MIP are enabled.

— The second CVS enables LFNST and explicit MTS; ISP and MIP are disabled.

— The third CVS enables LFNST and explicit MTS; ISP is enabled and MIP is disabled.

— The fourth CVS enables LFNST and explicit MTS; ISP is disabled and MIP is enabled.

Functional stage: Transform.

Purpose: Check that the decoder can properly decode bitstreams with LFNST enabled.

6.6.2.23.4 Test bitstream LFNST_D

Specification: This bitstream exercises LFNST and its signalling in different combination with other tools. The bitstream consists of four CVSs with the following properties:

— The first CVS enables LFNST and explicit MTS; ISP and MIP are enabled.

— The second CVS enables LFNST and explicit MTS; ISP and MIP are disabled.

— The third CVS enables LFNST and explicit MTS; ISP is enabled and MIP is disabled.

— The fourth CVS enables LFNST and explicit MTS; ISP is disabled and MIP is enabled.

Functional stage: Transform.

Purpose: Check that the decoder can properly decode bitstreams with LFNST enabled.

6.6.2.24 Transform tool set (MTS_LFNST)

6.6.2.24.1 Test bitstream MTS_LFNST_A

Specification: This bitstream exercises various types of enabling of MTS and LFNST. The bitstream consists of five parts with the following properties:

— 1st part

  — Explicit intra MTS on and explicit inter MTS off (CTC).

  — Include all test cases for ISP, MIP, luma tree, and CST.

  — Include all candidates of explicit MTS, i.e., DCT2-DCT2, DST7-DST7, DCT8-DST7, DST7-DCT8, and DCT8-DCT8.

  — Include all sets and candidates of LFNST, i.e., 4 sets and 2 candidates per set.

  — Include all possible block sizes and partitions (especially for ISP) where all combinations of MTS and LFNST can happen.

— 2nd part

  — Explicit intra MTS on and explicit inter MTS off with maximum transform size set to 32.

  — Include all test cases for ISP, MIP, luma tree, CST, and TU-tiling based on maximum transform size (32).

  - Include all candidates of explicit MTS, i.e., DCT2-DCT2, DST7-DST7, DCT8-DST7, DST7-DCT8, and DCT8-DCT8.
  - Include all sets and candidates of LFNST, i.e., 4 sets and 2 candidates per set.
  - Include all possible block sizes and partitions (especially for ISP) where all combinations of MTS and LFNST can happen.
- 3rd part
  - Implicit MTS on and explicit inter MTS off.
  - Include all test cases for ISP, MIP, luma tree, and CST.
  - Include all candidates of explicit MTS, i.e., DCT2-DCT2, DST7-DST7, DCT8-DST7, DST7-DCT8, and DCT8-DCT8.
  - Include all sets and candidates of LFNST, i.e., 4 sets and 2 candidates per set.
  - Include all possible block sizes and partitions (especially for ISP) where all combinations of MTS and LFNST can happen.
- 4th part
  - Implicit MTS on and explicit inter MTS on.
  - Include all test cases for ISP, MIP, luma tree, and CST.
  - Include all candidates of explicit MTS, i.e., DCT2-DCT2, DST7-DST7, DCT8-DST7, DST7-DCT8, and DCT8-DCT8.
  - Include all sets and candidates of LFNST, i.e., 4 sets and 2 candidates per set.
  - Include all possible block sizes and partitions (especially for ISP) where all combinations of MTS and LFNST can happen.
- 5th part
  - Explicit MTS on and explicit inter MTS on.
  - Include all test cases for ISP, MIP, luma tree, and CST.
  - Include all candidates of explicit MTS, i.e., DCT2-DCT2, DST7-DST7, DCT8-DST7, DST7-DCT8, and DCT8-DCT8.
  - Include all sets and candidates of LFNST, i.e., 4 sets and 2 candidates per set.
  - Include all possible block sizes and partitions (especially for ISP) where all combinations of MTS and LFNST can happen.

Functional stage: Transform.

Purpose: Check that the decoder can properly decode bitstreams with MTS and LFNST enabled.

6.6.2.24.2 Test bitstream MTS_LFNST_B

Specification: This bitstream exercises various types of enabling of MTS and LFNST. The bitstream consists of five parts with the following properties:

- 1st part
  - Explicit intra MTS on and explicit inter MTS off (CTC).
  - Include all test cases for SBT, single tree, and TU-tiling based on maximum transform size (64).

- Include all candidates of explicit MTS, i.e., DCT2-DCT2, DST7-DST7, DCT8-DST7, DST7-DCT8, and DCT8-DCT8.
- Include all sets and candidates of LFNST, i.e., 4 sets and 2 candidates per set.
- Include all possible block sizes and partitions (especially for SBT) where all combinations of MTS and LFNST can happen.

— 2nd part

- Explicit intra MTS on and explicit inter MTS off with maximum transform size set to 32.
- Include all test cases for SBT, single tree, and TU-tiling based on maximum transform size (32).
- Include all candidates of explicit MTS, i.e., DCT2-DCT2, DST7-DST7, DCT8-DST7, DST7-DCT8, and DCT8-DCT8.
- Include all sets and candidates of LFNST, i.e., 4 sets and 2 candidates per set.
- Include all possible block sizes and partitions (especially for SBT) where all combinations of MTS and LFNST can happen.

— 3rd part

- Implicit MTS on and explicit inter MTS off.
- Include all test cases for SBT, single tree and TU-tiling based on maximum transform size (64).
- Include all candidates of explicit MTS, i.e., DCT2-DCT2, DST7-DST7, DCT8-DST7, DST7-DCT8, and DCT8-DCT8.
- Include all sets and candidates of LFNST, i.e., 4 sets and 2 candidates per set.
- Include all possible block sizes and partitions (especially for SBT) where all combinations of MTS and LFNST can happen.

— 4th part

- Implicit MTS on and explicit inter MTS on.
- Include all test cases for SBT, single tree and TU-tiling based on maximum transform size (64).
- Include all candidates of explicit MTS, i.e., DCT2-DCT2, DST7-DST7, DCT8-DST7, DST7-DCT8, and DCT8-DCT8.
- Include all sets and candidates of LFNST, i.e., 4 sets and 2 candidates per set.
- Include all possible block sizes and partitions (especially for ISP) where all combinations of MTS and LFNST can happen.

— 5th part

- Explicit MTS on and explicit inter MTS on.
- Include all test cases for SBT, single tree and TU-tiling based on maximum transform size (64).
- Include all candidates of explicit MTS, i.e., DCT2-DCT2, DST7-DST7, DCT8-DST7, DST7-DCT8, and DCT8-DCT8.
- Include all sets and candidates of LFNST, i.e., 4 sets and 2 candidates per set.
- Include all possible block sizes and partitions (especially for SBT) where all combinations of MTS and LFNST can happen.

Functional stage: Transform.

Purpose: Check that the decoder can properly decode bitstreams with MTS and LFNST enabled.

6.6.2.25 Joint coding of chroma residuals (JCCR)

6.6.2.25.1 Test bitstream JCCR_A

Specification: Bitstream exercises all possible JCCR modes. In addition, different combinations for values of ph_joint_cbcr_sign_flag and sh_joint_cbcr_qp_offset syntax elements are included in the bitstream. Coded video contains three frames at resolution of 416x240.

Functional stage: TU reconstruction.

Purpose: Check that the decoder can properly decode TUs with different JCCR modes, different JCCR QP offsets and different JCCR signs.

6.6.2.25.2 Test bitstream JCCR_B

Specification: Bitstream exercises all possible JCCR modes. In addition, different combinations for values of ph_joint_cbcr_sign_flag and sh_joint_cbcr_qp_offset syntax elements are included in the bitstream. Coded video contains three frames at resolution of 1920x1080.

Functional stage: TU reconstruction.

Purpose: Check that the decoder can properly decode TUs with different JCCR modes, different JCCR QP offsets and different JCCR signs.

6.6.2.25.3 Test bitstream JCCR_C

Specification: This bitstream exercises joint chroma residual coding in combination with other tools. The bitstream consists of two CVSs with the following properties:

— The first CVS uses a random selection of the jointCbCr sign flag, forces the usage of all possible jointCbCr modes, non-related features (inter tools, ALF, ...) are disabled, MTS, LFNST, LMCS, and DQ are disabled.

— The first CVS uses a random selection of the jointCbCr sign flag, forces the usage of all possible jointCbCr modes, non-related features (inter tools, ALF, ...) are disabled, MTS (for intra), LFNST, LMCS, and DQ are enabled.

Functional stage: TU reconstruction.

Purpose: Check that the decoder can properly decode TUs with different JCCR modes.

6.6.2.25.4 Test bitstream JCCR_D

Specification: This bitstream exercises joint chroma residual coding in combination with other tools. The bitstream consists of two CVSs with the following properties:

— The first CVS uses a random selection of the jointCbCr sign flag, forces the usage of all possible jointCbCr modes, non-related features (inter tools, ALF, ...) are disabled, MTS, LFNST, LMCS, and DQ are disabled.

— The first CVS uses a random selection of the jointCbCr sign flag, forces the usage of all possible jointCbCr modes, non-related features (inter tools, ALF, ...) are disabled, MTS (for intra), LFNST, LMCS, and DQ are enabled.

Functional stage: TU reconstruction.

Purpose: Check that the decoder can properly decode TUs with different JCCR modes.

6.6.2.25.5 Test bitstream JCCR_E

Specification: Bitstream exercises all possible JCCR modes. In addition, different combinations for values of ph_joint_cbcr_sign_flag and sh_joint_cbcr_qp_offset syntax elements are included in the bitstream. Coded video contains three frames at resolution of 416x240.

Functional stage: TU reconstruction.

Purpose: Check that the decoder can properly decode TUs with different JCCR modes, different JCCR QP offsets and different JCCR signs.

6.6.2.25.6 Test bitstream JCCR_F

Specification: Bitstream exercises all possible JCCR modes. In addition, different combinations for values of ph_joint_cbcr_sign_flag and sh_joint_cbcr_qp_offset syntax elements are included in the bitstream. Coded video contains three frames at resolution of 1920x1080.

Functional stage: TU reconstruction.

Purpose: Check that the decoder can properly decode TUs with different JCCR modes, different JCCR QP offsets and different JCCR signs.

6.6.2.26 Temporal motion vector predictor (TMVP)

6.6.2.26.1 Test bitstream TMVP_A

Specification: Bitstream disables TMVP. Uses scaled 'ParkRunning3' test sequence with QP 32 and random access configuration.

Functional stage: Motion vector prediction.

Purpose: Check that the decoder can properly decode bitstreams with temporal motion vector prediction enabled and disabled.

6.6.2.26.2 Test bitstream TMVP_B

Specification: Bitstream enables TMVP. Uses scaled 'ParkRunning3' test sequence with QP 32 and random access configuration.

Functional stage: Motion vector prediction.

Purpose: Check that the decoder can properly decode bitstreams with temporal motion vector prediction enabled and disabled.

6.6.2.26.3 Test bitstream TMVP_C

Specification: Bitstream disables TMVP. Uses scaled 'ParkRunning3' test sequence with QP 32 and low delay configuration.

Functional stage: Motion vector prediction.

Purpose: Check that the decoder can properly decode bitstreams with temporal motion vector prediction enabled and disabled.

6.6.2.26.4 Test bitstream TMVP_D

Specification: Bitstream enables TMVP. Uses scaled 'ParkRunning3' test sequence with QP 32 and low delay configuration.

Functional stage: Motion vector prediction.

Purpose: Check that the decoder can properly decode bitstreams with temporal motion vector prediction enabled and disabled.

6.6.2.27 Motion vector compression (MVCOMP)

6.6.2.27.1 Test bitstream MVCOMP_A

Specification: This bitstream includes large motion vectors that are stored in the temporal motion vector buffer and that are later retrieved for motion vector prediction.

Functional stage: Inter prediction.

Purpose: Check that the decoder can properly decode bitstreams with large motion vectors.

6.6.2.28 Sampled adaptive offset (SAO)

6.6.2.28.1 Test bitstream SAO_A

Specification: This bitstream uses SAO with ALF and CCALF disabled.

Functional stage: In-loop filter.

Purpose: Check that the decoder can properly decode bitstreams with various in-loop filter combinations.

6.6.2.28.2 Test bitstream SAO_B

Specification: This bitstream uses SAO with LMCS disabled.

Functional stage: In-loop filter.

Purpose: Check that the decoder can properly decode bitstreams with various in-loop filter combinations.

6.6.2.28.3 Test bitstream SAO_C

Specification: This bitstream uses SAO with ALF, CCALF, and LMCS disabled.

Functional stage: In-loop filter.

Purpose: Check that the decoder can properly decode bitstreams with various in-loop filter combinations.

6.6.2.29 Prediction refinement using optical flow (PROF)

6.6.2.29.1 Test bitstream PROF_A

Specification: This bitstream contains pictures with high rotation motion.

Functional stage: Inter prediction.

Purpose: Check that the decoder can properly decode bitstreams with PROF enabled.

6.6.2.29.2 Test bitstream PROF_B

Specification: The bitstream contains pictures with high zoom and rotation motion.

Functional stage: Inter prediction.

Purpose: Check that the decoder can properly decode bitstreams with PROF enabled.

6.6.2.30 Deblocking (DEBLOCKING)

6.6.2.30.1 Test bitstream DEBLOCKING_A

Specification: This bitstream has luma deblocking filters of lengths (7,7), (7,5), (5,7), (7,3), (3,7), (5,5), (5,3) and (3,5).

Functional stage: In-loop filter.

Purpose: Check that the decoder can properly decode bitstreams with deblocking filter enabled.

6.6.2.30.2 Test bitstream DEBLOCKING_B

Specification: This bitstream has luma deblocking filters of lengths (3,3) and (1,3).

Functional stage: In-loop filter.

Purpose: Check that the decoder can properly decode bitstreams with deblocking filter enabled.

6.6.2.30.3 Test bitstream DEBLOCKING_C

Specification: This bitstream tests that luma deblocking is performed on a 4 x 4 deblocking grid and ensures that constrained weak filter (1 + 1) is applied when one of the blocks sharing the edge has size <= 4 samples in the direction of deblocking.

Functional stage: In-loop filter.

Purpose: Check that the decoder can properly decode bitstreams with deblocking filter enabled.

6.6.2.30.4 Test bitstream DEBLOCKING_E

Specification: Bitstream exercises all luma and chroma deblocking lengths for deblocking of transform and prediction block and sub-block boundaries.

Functional stage: In-loop filter.

Purpose: Check that the decoder can properly decode bitstreams with deblocking filter enabled.

6.6.2.30.5 Test bitstream DEBLOCKING_F

Specification: Bitstream exercises deblocking control features luma adaptive deblocking and control of beta and tc for both luma and chroma.

Functional stage: In-loop filter.

Purpose: Check that the decoder can properly decode bitstreams with deblocking filter enabled.

6.6.2.31 Weighted prediction (WP)

6.6.2.31.1 Test bitstream WP_A

Specification: The bitstream was encoded in random access configuration. The content has fading to black. WP have been disabled for pictures with Tid equal to 2.

Functional stage: Inter prediction.

Purpose: Check that the decoder can properly decode bitstreams with weighted prediction enabled.

6.6.2.31.2 Test bitstream WP_B

Specification: The bitstream was encoded in low-delay configuration. The content has flashing and fading.

Functional stage: Inter prediction.

Purpose: Check that the decoder can properly decode bitstreams with weighted prediction enabled.

6.6.2.32 Parallel merge (PMERGE)

6.6.2.32.1 Test bitstream PMERGE_A

Specification: This bitstream exercises the sps_log2_parallel_merge_level_minus2 equals to 1 with luma CTB size 128x128.

Functional stage: Inter prediction.

Purpose: Check that the decoder can properly decode bitstream with parallel merge control.

6.6.2.32.2 Test bitstream PMERGE_B

Specification: This bitstream exercises the sps_log2_parallel_merge_level_minus2 equals to 2 with luma CTB size 128x128.

Functional stage: Inter prediction.

Purpose: Check that the decoder can properly decode bitstream with parallel merge control.

6.6.2.32.3 Test bitstream PMERGE_C

Specification: This bitstream exercises the sps_log2_parallel_merge_level_minus2 equals to 3 with luma CTB size 128x128.

Functional stage: Inter prediction.

Purpose: Check that the decoder can properly decode bitstream with parallel merge control.

6.6.2.32.4 Test bitstream PMERGE_D

Specification: This bitstream exercises the sps_log2_parallel_merge_level_minus2 equals to 4 with luma CTB size 128x128.

Functional stage: Inter prediction.

Purpose: Check that the decoder can properly decode bitstream with parallel merge control.

6.6.2.32.5 Test bitstream PMERGE_E

Specification: This bitstream exercises the sps_log2_parallel_merge_level_minus2 equals to 5 with luma CTB size 128x128.

Functional stage: Inter prediction.

Purpose: Check that the decoder can properly decode bitstream with parallel merge control.

6.6.2.33 Intra prediction (IP)

6.6.2.33.1 Test bitstream IP_A

Specification: The bitstream exercises luminance intra prediction modes.

Functional stage: Intra sample prediction.

Purpose: Test intra sample reconstruction process.

6.6.2.33.2 Test bitstream IP_B

Specification: Bitstream exercises all the intra prediction modes.

Functional stage: Intra sample prediction.

Purpose: Test intra sample reconstruction process, especially wide angle modes in non-square blocks.

6.6.2.34 Luma intra prediction mode (MPM)

6.6.2.34.1 Test bitstream MPM_A

Specification: This bitstream contains MPM candidate for all the sizes of the blocks, i.e. all WxH-sized TUs, where both W and H are equal to one of the following values: {4, 8, 16 or 32}. All intra coding tools except MIP are enabled.

Functional stage: Intra prediction.

Purpose: Check that the decoder can properly decode all MPM modes.

6.6.2.35 CTU sizes (CTU, CU)

6.6.2.35.1 Test bitstream CTU_A

Specification: Bitstream exercises all possible CU sizes when maximum CTU size is set to 128x128.

Functional stage: Partitioning.

Purpose: Check that a decoder can parse and reconstruct correctly when maximum CTU size is set to 128x128.

6.6.2.35.2 Test bitstream CTU_B

Specification: Bitstream exercises all possible CU sizes when maximum CTU size is set to 64x64.

Functional stage: Partitioning.

Purpose: Check that a decoder can parse and reconstruct correctly when maximum CTU size is set to 64x64.

6.6.2.35.3 Test bitstream CTU_C

Specification: Bitstream exercises all possible CU sizes when maximum CTU size is set to 32x32.

Functional stage: Test the parsing and reconstruction of slices.

Purpose: Check that a decoder can parse and reconstruct correctly when maximum CTU size is set to 32x32.

6.6.2.36 Trees and partitioning (TREE, QTBTT)

6.6.2.36.1 Test bitstream TREE_A

Specification: This bitstream exercises a range of tree size and depths for CTUSize=32.

Functional stage: Partitioning.

Purpose: Check that the decoder can properly decode all partitioning modes.

6.6.2.36.2 Test bitstream TREE_B

Specification: This bitstream exercises a range of tree size and depths for CTUSize=64.

Functional stage: Partitioning.

Purpose: Check that the decoder can properly decode all partitioning modes.

6.6.2.36.3 Test bitstream TREE_C

Specification: This bitstream exercises a range of tree size and depths for CTUSize=128.

Functional stage: Partitioning.

Purpose: Check that the decoder can properly decode all partitioning modes.

6.6.2.36.4 Test bitstream QTBTT_A

Specification: Bitstream exercises all possible range of CU sizes and depths for QTBTT partitions.

Functional stage: Test the parsing and reconstruction of slices.

Purpose: Check that a decoder can parse and reconstruct correctly for all exercise range of CU sizes and depth for QTBTT partitions.

6.6.2.37 Boundary partition (BOUNDARY)

6.6.2.37.1 Test bitstream BOUNDARY_A

Specification: This bitstream tests boundary handling on specific resolution with WidthxHeight, where Width = 256+8*n, Height = 256 + 8*m, m and n belong to {0...15}. QT depths for boundary blocks are selected as {1, 2, 3, 4} with POC = {1, 2, 3, 4}, respectively.

Functional stage: Partitioning.

Purpose: Check that the decoder can properly decode all partitioning modes.

6.6.2.38 Transform (TRANS)

6.6.2.38.1 Test bitstream TRANS_A

Specification: The bitstream has size 64 transform off, LFNST off, and DepQuant off for All Intra.

Functional stage: Transform.

Purpose: Check that the decoder can properly decode all transform modes.

6.6.2.38.2 Test bitstream TRANS_B

Specification: The bitstream has size 64 transform on, LFNST off, and DepQuant off for All Intra.

Functional stage: Transform.

Purpose: Check that the decoder can properly decode all transform modes.

6.6.2.38.3 Test bitstream TRANS_C

Specification: The bitstream has size 64 transform off, LFNST off, and DepQuant off for random access.

Functional stage: Transform.

Purpose: Check that the decoder can properly decode all transform modes.

6.6.2.38.4 Test bitstream TRANS_D

Specification: The bitstream has size 64 transform on, LFNST off, and DepQuant off for random access.

Functional stage: Transform.

Purpose: Check that the decoder can properly decode all transform modes.

6.6.2.39 Quantization (QUANT)

6.6.2.39.1 Test bitstream QUANT_A

Specification: This bitstream tests CU level QP adaptation.

Functional stage: Quantization.

Purpose: Check that the decoder can properly decode all quantization modes.

6.6.2.39.2 Test bitstream QUANT_B

Specification: This bitstream tests CU level QP adaptation.

Functional stage: Quantization.

Purpose: Check that the decoder can properly decode all quantization modes.

6.6.2.39.3 Test bitstream QUANT_C

Specification: This bitstream uses low QP with transform skip, so that deblocking filtering not used.

Functional stage: Quantization, In-loop filtering.

Purpose: Check that the decoder can properly decode all quantization modes.

6.6.2.39.4 Test bitstream QUANT_D

Specification: Deblocking filtering is forced to be used by setting LoopFilterTcOffset_div2 and LoopFilterTcOffset_div2 to 12.

Functional stage: Quantization, In-loop filtering.

Purpose: Check that the decoder can properly decode all quantization modes.

6.6.2.39.5 Test bitstream QUANT_E

Specification: The bitstream use of local chroma QP offsets. It signals a PPS chroma QP offset list of maximal size, making use of maximal QP offset range.

Functional stage: Quantization.

Purpose: Check that the decoder can parse and use local chroma QP offsets.

6.6.2.40 Scaling list (SCALING)

6.6.2.40.1 Test bitstream SCALING_A

Specification: Bitstream uses scaling list in monochrome mode. Composed of 2 concatenated streams. The first one sets scaling lists every 8 frames, the second one sets scaling lists based on temporal ID.

Functional stage: Inverse quantization.

Purpose: Check that decoder can parse and use scaling list quantization matrices.

6.6.2.40.2 Test bitstream SCALING_B

Specification: The bitstream uses quantization matrices in colour mode for LFNST blocks. Composed of 2 concatenated streams. The first one sets scaling lists every 8 frames, the second one sets scaling lists based on temporal ID with sps_scaling_matrix_for_lfnst_disabled_flag OFF.

Functional stage: Inverse quantization.

Purpose: Check that decoder can parse and use scaling list quantization matrices.

6.6.2.40.3 Test bitstream SCALING_C

Specification: The bitstream sses scaling lists, reuses apsId, sets scaling lists ON and OFF. Uses slices, varying the scaling list enable flag in slice header.

Functional stage: Inverse quantization.

Purpose: Check that decoder can parse and use scaling list quantization matrices.

6.6.2.41 Entropy coding (ENTROPY)

6.6.2.41.1 Test bitstream ENTROPY_A

Specification: This bitstream uses high bitrate for Low Delay P mode, with MIP enabled.

Functional stage: Entropy coding.

Purpose: Check that the decoder properly decodes all entropy coding modes.

6.6.2.41.2 Test bitstream ENTROPY_B

Specification: The bitstream includes all combinations of cabac_init_flag in slice header (0, 1, absent).

Functional stage: Entropy coding.

Purpose: Check that the decoder properly decodes all entropy coding modes.

6.6.2.41.3 Test bitstream ENTROPY_C

Specification: Bitstream tests CABAC initialization, sweeping QP from 0 to 63.

Functional stage: Entropy coding.

Purpose: Check that the decoder properly decodes all entropy coding modes.

6.6.2.42 Entropy coding (ENTMAINTIER)

6.6.2.42.1 Test bitstream ENTMAINTIER_A

Specification: The bitstream contains 3 independent CVSs containing one picture, formed from one slice. Each CVS contains the maximum number of bits, given the profile, level and tier, and assuming equal distribution of bits between pictures coded at the maximum luma sample rate for the level, at Level 4. The 3 concatenated CVSs are as follows:

— The first picture does not require any CABAC zero words.

— The second picture requires one CABAC zero word.

— The third picture requires a substantial quantity of CABAC zero words (75 % of the bitstream is padding).

All 3 pictures have a very low subjective quality level due to the artificial nature of these bitstreams. All VCL NAL units contain almost (within 3 bytes worth) of the maximum number of allowed bins, for their size.

Functional stage: Entropy coding.

Purpose: Check that the decoder properly decodes all entropy coding modes.

6.6.2.42.2 Test bitstream ENTMAINTIER_B

Specification: The bitstream contains 3 independent CVSs containing one picture, formed from one slice. Each CVS contains the maximum number of bits, given the profile, level and tier, and assuming equal distribution of bits between pictures coded at the maximum luma sample rate for the level, at Level 4.1. The 3 concatenated CVSs are as follows:

— The first picture does not require any CABAC zero words.

— The second picture requires one CABAC zero word.

— The third picture requires a substantial quantity of CABAC zero words (75 % of the bitstream is padding).

All 3 pictures have a very low subjective quality level due to the artificial nature of these bitstreams. All VCL NAL units contain almost (within 3 bytes worth) of the maximum number of allowed bins, for their size.

Functional stage: Entropy coding.

Purpose: Check that the decoder properly decodes all entropy coding modes.

6.6.2.42.3 Test bitstream ENTMAINTIER_C

Specification: The bitstream contains 3 independent CVSs containing one picture, formed from one slice. Each CVS contains the maximum number of bits, given the profile, level and tier, and assuming equal distribution of bits between pictures coded at the maximum luma sample rate for the level, at Level 5. The 3 concatenated CVSs are as follows:

— The first picture does not require any CABAC zero words.

— The second picture requires one CABAC zero word.

— The third picture requires a substantial quantity of CABAC zero words (75 % of the bitstream is padding).

All 3 pictures have a very low subjective quality level due to the artificial nature of these bitstreams. All VCL NAL units contain almost (within 3 bytes worth) of the maximum number of allowed bins, for their size.

Functional stage: Entropy coding.

Purpose: Check that the decoder properly decodes all entropy coding modes.

6.6.2.42.4 Test bitstream ENTMAINTIER_D

Specification: The bitstream contains 3 independent CVSs containing one picture, formed from one slice. Each CVS contains the maximum number of bits, given the profile, level and tier, and assuming equal distribution of bits between pictures coded at the maximum luma sample rate for the level, at Level 5.1. The 3 concatenated CVSs are as follows:

— The first picture does not require any CABAC zero words.

— The second picture requires one CABAC zero word.

— The third picture requires a substantial quantity of CABAC zero words (75 % of the bitstream is padding).

All 3 pictures have a very low subjective quality level due to the artificial nature of these bitstreams. All VCL NAL units contain almost (within 3 bytes worth) of the maximum number of allowed bins, for their size.

Functional stage: Entropy coding.

Purpose: Check that the decoder properly decodes all entropy coding modes.

6.6.2.43 Entropy coding (ENTHIGHTIER)

6.6.2.43.1 Test bitstream ENTHIGHTIER_A

Specification: The bitstream contains 3 independent CVSs containing one picture, formed from one slice. Each CVS contains the maximum number of bits, given the profile, level and tier, and assuming equal distribution of bits between pictures coded at the maximum luma sample rate for the level, at Level 4 for High tier. The 3 concatenated CVSs are as follows:

— The first picture does not require any CABAC zero words.

— The second picture requires one CABAC zero word.

— The third picture requires a substantial quantity of CABAC zero words (75 % of the bitstream is padding).

All 3 pictures have a very low subjective quality level due to the artificial nature of these bitstreams. All VCL NAL units contain almost (within 3 bytes worth) of the maximum number of allowed bins, for their size.

Functional stage: Entropy coding.

Purpose: Check that the decoder properly decodes all entropy coding modes.

6.6.2.43.2 Test bitstream ENTHIGHTIER_B

Specification: The bitstream contains 3 independent CVSs containing one picture, formed from one slice. Each CVS contains the maximum number of bits, given the profile, level and tier, and assuming

equal distribution of bits between pictures coded at the maximum luma sample rate for the level, at Level 4.1 for High tier. The 3 concatenated CVSs are as follows:

— The first picture does not require any CABAC zero words.

— The second picture requires one CABAC zero word.

— The third picture requires a substantial quantity of CABAC zero words (75 % of the bitstream is padding).

All 3 pictures have a very low subjective quality level due to the artificial nature of these bitstreams. All VCL NAL units contain almost (within 3 bytes worth) of the maximum number of allowed bins, for their size.

Functional stage: Entropy coding.

Purpose: Check that the decoder properly decodes all entropy coding modes.

6.6.2.43.3 Test bitstream ENTHIGHTIER_C

Specification: The bitstream contains 3 independent CVSs containing one picture, formed from one slice. Each CVS contains the maximum number of bits, given the profile, level and tier, and assuming equal distribution of bits between pictures coded at the maximum luma sample rate for the level, at Level 5 for High tier. The 3 concatenated CVSs are as follows:

— The first picture does not require any CABAC zero words.

— The second picture requires one CABAC zero word.

— The third picture requires a substantial quantity of CABAC zero words (75 % of the bitstream is padding).

All 3 pictures have a very low subjective quality level due to the artificial nature of these bitstreams. All VCL NAL units contain almost (within 3 bytes worth) of the maximum number of allowed bins, for their size.

Functional stage: Entropy coding.

Purpose: Check that the decoder properly decodes all entropy coding modes.

6.6.2.43.4 Test bitstream ENTHIGHTIER_D

Specification: The bitstream contains 3 independent CVSs containing one picture, formed from one slice. Each CVS contains the maximum number of bits, given the profile, level and tier, and assuming equal distribution of bits between pictures coded at the maximum luma sample rate for the level, at Level 5.1 for High tier. The 3 concatenated CVSs are as follows:

— The first picture does not require any CABAC zero words.

— The second picture requires one CABAC zero word.

— The third picture requires a substantial quantity of CABAC zero words (75 % of the bitstream is padding).

All 3 pictures have a very low subjective quality level due to the artificial nature of these bitstreams. All VCL NAL units contain almost (within 3 bytes worth) of the maximum number of allowed bins, for their size.

Functional stage: Entropy coding.

Purpose: Check that the decoder properly decodes all entropy coding modes.

6.6.2.44 All merge modes (MERGE)

6.6.2.44.1 Test bitstream MERGE_A

Specification: This bitstream exercises the maximum number of merge candidates (MaxNumMergeCand) = 1 and Maximum number of GPM merge candidates (MaxNumGpmMergeCand) = 0.

Functional stage: Inter prediction.

Purpose: Check that the decoder properly decodes all merge modes.

6.6.2.44.2 Test bitstream MERGE_B

Specification: This bitstream exercises the maximum number of merge candidates (MaxNumMergeCand) = 2 and Maximum number of GPM merge candidates (MaxNumGpmMergeCand) = 2.

Functional stage: Inter prediction.

Purpose: Check that the decoder properly decodes all merge modes.

6.6.2.44.3 Test bitstream MERGE_C

Specification: This bitstream exercises the maximum number of merge candidates (MaxNumMergeCand) = 3 and Maximum number of GPM merge candidates (MaxNumGpmMergeCand) = 3.

Functional stage: Inter prediction.

Purpose: Check that the decoder properly decodes all merge modes.

6.6.2.44.4 Test bitstream MERGE_D

Specification: This bitstream exercises the maximum number of merge candidates (MaxNumMergeCand) = 4 and Maximum number of GPM merge candidates (MaxNumGpmMergeCand) = 4.

Functional stage: Inter prediction.

Purpose: Check that the decoder properly decodes all merge modes.

6.6.2.44.5 Test bitstream MERGE_E

Specification: This bitstream exercises the maximum number of merge candidates (MaxNumMergeCand) = 5 and Maximum number of GPM merge candidates (MaxNumGpmMergeCand) = 5.

Functional stage: Inter prediction.

Purpose: Check that the decoder properly decodes all merge modes.

6.6.2.44.6 Test bitstream MERGE_F

Specification: This bitstream exercises the maximum number of merge candidates (MaxNumMergeCand) = 1 and Maximum number of GPM merge candidates (MaxNumGpmMergeCand) = 0.

Functional stage: Inter prediction.

Purpose: Check that the decoder properly decodes all merge modes.

6.6.2.44.7 Test bitstream MERGE_G

Specification: This bitstream exercises the maximum number of merge candidates (MaxNumMergeCand) = 2 and Maximum number of GPM merge candidates (MaxNumGpmMergeCand) = 2.

Functional stage: Inter prediction.

Purpose: Check that the decoder properly decodes all merge modes.

6.6.2.44.8 Test bitstream MERGE_H

Specification: This bitstream exercises the maximum number of merge candidates (MaxNumMergeCand) = 3 and Maximum number of GPM merge candidates (MaxNumGpmMergeCand) = 3.

Functional stage: Inter prediction.

Purpose: Check that the decoder properly decodes all merge modes.

6.6.2.44.9 Test bitstream MERGE_I

Specification: This bitstream exercises the maximum number of merge candidates (MaxNumMergeCand) = 4 and Maximum number of GPM merge candidates (MaxNumGpmMergeCand) = 4.

Functional stage: Inter prediction.

Purpose: Check that the decoder properly decodes all merge modes.

6.6.2.44.10 Test bitstream MERGE_J

Specification: This bitstream exercises the maximum number of merge candidates (MaxNumMergeCand) = 5 and Maximum number of GPM merge candidates (MaxNumGpmMergeCand) = 5.

Functional stage: Inter prediction.

Purpose: Check that the decoder properly decodes all merge modes.

6.6.2.45 Position dependent prediction combination (PDPC)

6.6.2.45.1 Test bitstream PDPC_A

Specification: This bitstream tests the clipping function in PDPC for horizontal or vertical intra prediction modes.

Functional stage: Intra coding.

Purpose: Check that the decoder properly decodes the bitstream when PDPC is enabled.

6.6.2.45.2 Test bitstream PDPC_B

Specification: This bitstream uses DPC with various block sizes.

Functional stage: Intra coding.

Purpose: Check that the decoder properly decodes the bitstream when PDPC is enabled.

6.6.2.45.3 Test bitstream PDPC_C

Specification: This bitstream tests the clipping function in PDPC and the PDPC conditional check on the intra prediction mode. In this test each picture is a single I-slice where all the luma blocks are encoded using an identical intra prediction mode.

Functional stage: Intra coding.

Purpose: Check that the decoder properly decodes the bitstream when PDPC is enabled.

6.6.2.46 Wavefronts (WPP)

6.6.2.46.1 Test bitstream WPP_A

Specification: The bitstream is encoded with sps_entropy_coding_sync_enabled_flag equal to 1.

Functional stage: Inter prediction.

Purpose: Check that the decoder can properly decode bitstreams with wavefront enabled.

6.6.2.46.2 Test bitstream WPP_B

Specification: The bitstream is encoded with rectangular tile and sps_entropy_coding_sync_enabled_flag equal to 1 and pictures contain 4 tiles.

Functional stage: Inter prediction.

Purpose: Check that the decoder can properly decode bitstreams with wavefront enabled and tiles.

6.6.2.47 Lossless and near-lossless, include transform skip (LOSSLESS)

6.6.2.47.1 Test bitstream LOSSLESS_A

Specification: The coded slices are either I or B, and all blocks employ the transform skip mode and the regular residual coding stage for entropy coding.

Functional stage: Test the parsing and reconstruction of slices.

Purpose: Check that a decoder can parse and reconstruct correctly when the bitstream consists of transform skip mode and the regular residual coding stage operating at the lossless operation point.

6.6.2.47.2 Test bitstream LOSSLESS_B

Specification: The coded slices are either I or B, and most of the blocks employ the transform skip mode and the corresponding residual coding stage for entropy coding.

Functional stage: Test the parsing and reconstruction of slices.

Purpose: Check that a decoder can parse and reconstruct correctly when the bitstream consists of a high amount of transform skip mode but using the transform skip residual coding for entropy coding.

6.6.2.48 Reference picture resizing (RPR)

6.6.2.48.1 Test bitstream RPR_A

Specification: This bitstream has 4 pictures. The bitstream contains CUs encoded with inter-prediction mode using reference pictures with a higher resolution than the current picture. The luma resolution is 832x480 for pictures 0 and 1 and 1664x960 for pictures 2 and 3.

Functional stage: Inter prediction.

Purpose: Check that the decoder can properly decode a bitstream with RPR enabled.

6.6.2.48.2 Test bitstream RPR_B

Specification: The bitstream contains CUs encoded with inter-prediction mode using reference pictures with a higher resolution than the current picture. The luma resolution is 832x480 for pictures 0 and 1 and of 416x240 for pictures 2 and 3.

Functional stage: Inter prediction.

Purpose: Check that the decoder can properly decode a bitstream with RPR enabled.

6.6.2.48.3 Test bitstream RPR_C

Specification: The bitstream contains CUs encoded with inter-prediction mode using reference pictures with a higher resolution than the current picture. The luma resolution is 832x480 for pictures 0 and 1 and of 560x320 for pictures 2 and 3.

Functional stage: Inter prediction.

Purpose: Check that the decoder can properly decode a bitstream with RPR enabled.

6.6.2.48.4 Test bitstream RPR_D

Specification: The bitstream uses reference picture resampling with 2x ratio with sps_chroma_horizontal_collocated_flag = 03.

Functional stage: Inter prediction.

Purpose: Check that the decoder can properly decode a bitstream with RPR enabled.

6.6.2.49 Cross-component ALF (CCALF)

6.6.2.49.1 Test bitstream CCALF_A

Specification: This bitstream enables CCALF with filters that would exceed the output dynamic range and require a clip.

Functional stage: In-loop filtering.

Purpose: Check that the decoder can properly decode a bitstream with CCALF enabled.

6.6.2.49.2 Test bitstream CCALF_B

Specification: This bitstream enables CCALF for all CTUs in the bitstream.

Functional stage: In-loop filtering.

Purpose: Check that the decoder can properly decode a bitstream with CCALF enabled.

6.6.2.49.3 Test bitstream CCALF_C

Specification: This bitstream changes CCALF filters on a picture-by-picture basis.

Functional stage: In-loop filtering.

Purpose: Check that the decoder can properly decode a bitstream with CCALF enabled.

6.6.2.49.4 Test bitstream CCALF_D

Specification: This bitstream enables CCALF for random CTUs in the bitstream (both channels).

Functional stage: In-loop filtering.

Purpose: Check that the decoder can properly decode a bitstream with CCALF enabled.

6.6.2.50 Geometric partitioning mode (GPM)

6.6.2.50.1 Test bitstream GPM_A

Specification: This bitstream contains CUs with all the combinations of geometric partition modes, i.e. all the WxH sized CUs with 0 – 63 geometric partition modes. The value of WxH is equal to one of the following values: {8x8, 8x16, 8x32, 16x8, 16x16, 16x32, 16x64, 32x8, 32x16, 32x32, 32x64, 64x16, 64x32 and 64x64}.

Functional stage: Inter prediction.

Purpose: Check that the decoder can properly decode a bitstream with GPM enabled.

6.6.2.50.2 Test bitstream GPM_B

Specification: This bitstream uses different numbers of GPM candidates, including 2, 3, 4, 5 and 6. Correspondingly, the value of sps_max_num_merge_cand_minus_max_num_gpm_cand is set to 4, 3, 2, 1 and 0, respectively.

Functional stage: Inter prediction.

Purpose: Check that the decoder can properly decode a bitstream with GPM enabled.

6.6.2.51 Coding tool sets (CodingToolSets)

6.6.2.51.1 Test bitstream CodingToolsSets_A

Specification: This bitstream enables and disables coding tools corresponding to set A in Table 2.

Functional stage: General decoding.

Purpose: Check that a decoder can decode bitstreams using various combinations of coding tools.

6.6.2.51.2 Test bitstream CodingToolsSets_B

Specification: This bitstream enables and disables coding tools corresponding to set B in Table 2.

Functional stage: General decoding.

Purpose: Check that a decoder can decode bitstreams using various combinations of coding tools.

6.6.2.51.3 Test bitstream CodingToolsSets_C

Specification: This bitstream enables and disables coding tools corresponding to set C in Table 2.

Functional stage: General decoding.

Purpose: Check that a decoder can decode bitstreams using various combinations of coding tools.

6.6.2.51.4 Test bitstream CodingToolsSets_D

Specification: This bitstream enables and disables coding tools corresponding to set D in Table 2.

Functional stage: General decoding.

Purpose: Check that a decoder can decode bitstreams using various combinations of coding tools.

6.6.2.51.5 Test bitstream CodingToolsSets_E

Specification: This bitstream enables and disables coding tools corresponding to set E in Table 2.

Functional stage: General decoding.

Purpose: Check that a decoder can decode bitstreams using various combinations of coding tools.

6.6.3 Test bitstreams – High-level syntax features for Main 10 profile with 4:2:0 chroma format and 10 bit depth

6.6.3.1 Access unit delimiter (AUD)

6.6.3.1.1 Test bitstream: AUD_A

Specification: Pictures may or may not include associated Access Unit Delimiter (AUD) NAL units. The first 10 pictures of this bitstream do not include AUD, the next 10 pictures included AUD, and finally the last 10 pictures do not include AUD.

Functional stage: High-level syntax processing / picture boundary processing.

Purpose: Check that the decoder can handle and transition between pictures with and without associated Access Unit Delimiter NAL units.

6.6.3.2 Filler (FILLER)

6.6.3.2.1 Test bitstream FILLER_A

Specification: Each picture includes associated Filler data NAL units.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode when filler data NAL units are present in the bitstream.

6.6.3.3 Decoding Capability Indication (DCI)

6.6.3.3.1 Test bitstream DCI_A

Specification: The bitstream includes a DCI NAL unit.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode when DCI NAL unit is present in the bitstream.

6.6.3.3.2 Test bitstream DCI_B

Specification: The bitstream does not include a DCI NAL unit.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode when DCI NAL unit is not present in the bitstream.

6.6.3.4 Sequence parameter set (SPS)

6.6.3.4.1 Test bitstream SPS_A

Specification: Multiple SPSs are signalled in the same CVS. SPS with SPS ID equal to 0 is used and the other SPSs are never referenced.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode when multiple SPSs, including unreferenced ones, are contained in the same CVS.

6.6.3.4.2 Test bitstream SPS_B

Specification: Multiple SPSs are signalled in the bitstream. Different SPS IDs are used in the different CVSs.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode when different CVSs use different SPSs with different SPS IDs.

6.6.3.4.3 Test bitstream SPS_C

Specification: Multiple SPSs are signalled in the bitstream and SPS with SPS ID equal to 0 is used in the bitstream. ALF and BCW are enabled in SPS for the first two CVSs and are disabled in SPS for the third CVS.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode when the content of an SPS is replaced while using the same SPS ID in different CVSs.

6.6.3.5 Video usability information (PQ, HLG)

6.6.3.5.1 Test bitstream PQ_A

Specification: This bitstream uses VUI transfer characteristics for PQ content.

Functional stage: High-level syntax.

Purpose: Check that the decoder can properly parse the VUI.

6.6.3.5.2 Test bitstream HLG_A

Specification: This bitstream uses video usability information transfer characteristics for HLG content.

Functional stage: High-level syntax.

Purpose: Check that the decoder can properly parse the VUI.

6.6.3.5.3 Test bitstream HLG_B

Specification: This bitstream uses VUI to indicate "backward-compatible HLG" which is encoded using transfer characteristics with encoded as 1 and also include the alternative transfer characteristics SEI message with preferred transfer characteristics set to 18.

Functional stage: High-level syntax.

Purpose: Check that the decoder can properly parse the VUI.

6.6.3.6 Picture parameter set (PPS)

6.6.3.6.1 Test bitstream PPS_A

Specification: Multiple PPSs are signalled in the bitstream. PPS with PPS ID equal to 0 is used for each picture and the other PPSs are never referenced.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode when multiple PPSs, including unreferenced ones, are contained in the same CVS.

6.6.3.6.2 Test bitstream PPS_B

Specification: Each picture uses an individual PPS with a different PPS ID.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode when PPS IDs are switched for individual pictures in the bitstream.

6.6.3.6.3 Test bitstream PPS_C

Specification: Multiple PPSs are signalled in the bitstream. PPS with PPS ID equal to 0 is used for the first two CVSs. The content of the PPS is updated and used for all pictures in the second CVS.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode when the content of a PPS is replaced while using the same PPS ID in different pictures.

6.6.3.7 Mixed NUT (MNUT)

6.6.3.7.1 Test bitstream MNUT_A

Specification: The bitstream contains mixed NAL unit types. Encoded subpicture bitstreams were merged into one bitstream using the subpicMergeApp tool that is included in the VTM package. The bitstream contains 4 subpictures arranged in 2x2 formation. The first subpicture bitstream has CRA subpicture every 32th picture while the other 3 subpicture bitstreams don't have IRAP pictures after the first picture. Hence, in every 32th picture there is a mix of CRA subpicture and trailing subpictures. The encoder was slightly modified to ensure that reference picture list syntax is the same for CRA bitstream and non-CRA bitstreams.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode bitstreams with mixed NAL unit types in the same picture.

6.6.3.7.2 Test bitstream MNUT_B

Specification: The bitstream contains mixed NAL unit bitstream that contains mixed NAL unit types in some of the pictures and uses the sps_idr_rpl_present_flag syntax element.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode bitstreams with mixed NAL unit types in the same picture.

6.6.3.8 Extension of parameter set (PSEXT)

6.6.3.8.1 Test bitstream: PSEXT_A

Specification: The extension flag of the following parameter sets is set to one. DCI, VPS, SPS, PPS, APS.

Functional stage: Test the handling when the extension_flag is set to one and the related extension_data is absent for the following parameter sets. DCI, VPS, SPS, PPS, APS.

Purpose: Check that the decoder can parse the extension_flag set to one and handle when the related extension_data is absent for the following parameter sets. DCI, VPS, SPS, PPS, APS.

6.6.3.8.2 Test bitstream: PSEXT_B

Specification: The extension flag of the following parameter sets is set to one. DCI, VPS, SPS, PPS, APS.

Functional stage: Test the handling when the extension_flag is set to one and the related extension_data has one or more bits for the following parameter sets. DCI, VPS, SPS, PPS, APS.

Purpose: Check that the decoder can parse the extension_flag set to one and handle when the related extension_data has one or more bits for the following parameter sets. DCI, VPS, SPS, PPS, APS.

6.6.3.9 Hypothetical reference decoder (HRD)

6.6.3.9.1 Test bitstream HRD_A

Specification: This bitstream tests AU-based HRD operation, using the Buffering Period SEI and Picture Timing SEI messages.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly process HRD data.

6.6.3.9.2 Test bitstream HRD_B

Specification: This bitstream tests AU-based HRD operation, using the Buffering Period SEI and Picture Timing SEI messages, with 2 DUs in each AU.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly process HRD data.

6.6.3.10 Adaptation parameter set (APSALF, APSLMCS, APSMULT, SUFAPS)

6.6.3.10.1 Test bitstream APSALF_A

Specification: This bitstream uses multiple ALF APS, with only ALF APS is present in the bistream (LMCS and scaling list are disabled).

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode bitstreams using APSs.

6.6.3.10.2 Test bitstream APSLMCS_A

Specification: The bitstream uses multiple (3) LMCS APS (APS id = 0, 1 and 2), only LMCS APS is present in the bitstream (ALF is disabled).

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode bitstreams using APSs.

6.6.3.10.3 Test bitstream APSLMCS_B

Specification: The bitstream uses multiple (3) LMCS APS (APS id = 0, 1 and 2), both LMCS APS and ALF APS are present in the bitstream.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode bitstreams using APSs.

6.6.3.10.4 Test bitstream APSLMCS_C

Specification: The bitstream tests the use of LMCS APS with a large variation of lmcsCW values ([15 ~ 320]) in each of the 16 bins.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode bitstreams using APSs.

6.6.3.10.5 Test bitstream APSLMCS_D

Specification: The bitstream tests the use of LMCS APS with different min/max bin index.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode bitstreams using APSs.

6.6.3.10.6 Test bitstream APSLMCS_E

Specification: The bitstream tests the use of LMCS APS with negative CRSOffset value.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode bitstreams using APSs.

6.6.3.10.7 Test bitstream APSMULT_A

Specification: Multiple scaling list APSs (with scaling list APS ID equal to 0 and 1) are signalled in the bitstream. For each picture, the referenced scaling list APS ID is decided according to the picture's POC number.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode when multiple scaling list APSs are contained in the same CVS.

6.6.3.10.8 Test bitstream APSMULT_B

Specification: Multiple scaling list APSs are signalled in the bitstream with the same scaling list APS ID. When a scaling list ASP is signalled, it will overwrite the existing scaling list ASP.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode when multiple scaling list APSs are contained in the same CVS, and they can be overwritten by each other.

6.6.3.10.9 Test bitstream SUFAPS_A

Specification: The bitstream contains suffix APS NAL units for ALF.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode bitstreams containing suffix APSs.

6.6.3.11 Random access points (RAP)

6.6.3.11.1 Test bitstream RAP_A

Specification: The bitstream starts with a CRA picture. The CRA picture is followed only by RASL pictures, which are expected to be discarded.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode bitstreams containing various random access picture types.

6.6.3.11.2 Test bitstream RAP_B

Specification: The bitstream starts with a CRA picture. The CRA picture is followed by RASL and other pictures.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode bitstreams containing various random access picture types.

6.6.3.11.3 Test bitstream RAP_C

Specification: The bitstream contains IDR pictures with RADL pictures.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode bitstreams containing various random access picture types.

6.6.3.11.4 Test bitstream RAP_D

Specification: The bitstream contains IDR pictures without leading pictures.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode bitstreams containing various random access picture types.

6.6.3.12 Picture output (POUT)

6.6.3.12.1 Test bitstream POUT_A

Specification: This bitstream exercises picture output related syntax, with both values of ph_pic_output_flag.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly output pictures.

6.6.3.13 Gradual decoder refresh (GDR)

6.6.3.13.1 Test bitstream GDR_A

Specification: The bitstream starts with GDR pictures with recovery POC = 0.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can decode and handle GDR signalling.

6.6.3.13.2 Test bitstream GDR_B

Specification: The bitstream starts with GDR picture (POC 10) with ph_recovery_poc_cnt = 51. A second GDR picture starts at POC 70 with ph_recovery_poc_cnt = 51.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can decode and handle GDR signalling.

6.6.3.13.3 Test bitstream GDR_C

Specification: The bitstreams starts with a GDR picture (POC 60) and ph_recovery_poc_cnt = 29

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can decode and handle GDR signalling.

6.6.3.13.4 Test bitstream GDR_D

Specification: The bitstream exercises overlapping GDR periods with GDR pictures at POC 5, 6 and 30 with recovery points at POC 26, 33 and 51 respectively. The bitstream starts at POC 5 but conformance starts at POC 26. GDR picture at POC 5 references missing pictures POC 1-4, at POC 26 the refresh period has been completed and a decoder may start output.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can decode and handle GDR signalling.

6.6.3.14 Picture order count (POC)

6.6.3.14.1 Test bitstream POC_A

Specification: This bitstream exercises POC derivation, including POC reset, using the sps_poc_msb_cycle_flag, ph_poc_msb_cycle_present, ph_poc_msb_cycle_val and sps_poc_msb_cycle_len_minus1 syntax elements.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly derive POC values.

6.6.3.15 Tiles (TILE)

6.6.3.15.1 Test bitstream TILE_A

Specification: Each picture contains a single tile and single slice.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode single tile and single slice case.

6.6.3.15.2 Test bitstream TILE_B

Specification: Each picture contains uniform tile partitioning along both horizontal and vertical directions with each tile containing single slice.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode uniformly partitioned tiles.

6.6.3.15.3 Test bitstream TILE_C

Specification: Each picture contains tile partitioning with one row and N columns with each tile containing single slice.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode when tile partitioning has one row and N columns.

6.6.3.15.4 Test bitstream TILE_D

Specification: Each picture contains tile partitioning with N rows and one column with each tile containing single slice.

Functional stage: High-level syntax processing

Purpose: Check that the decoder can properly decode when tile partitioning has N rows and one column.

6.6.3.15.5 Test bitstream TILE_E

Specification: Each picture contains tile partitioning with M rows and N columns with each tile containing single slice.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode when tile partitioning has M rows and N columns.

6.6.3.15.6 Test bitstream TILE_F

Specification: Each picture contains tile partitioning with M rows and N columns with each tile containing multiple slices.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode when tile partitioning has M rows and N columns with each tile containing multiple slices.

6.6.3.15.7 Test bitstream TILE_G

Specification: Each picture contains tile partitioning with M rows and N columns and the whole picture containing a single slice.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode when tile partitioning has M rows and N columns and the whole picture containing a single slice.

6.6.3.16 Slices (SLICES)

6.6.3.16.1 Test bitstream SLICES_A

Specification: This bitstream exercises several different slice/tile layouts.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode when a variety of slice and tile layouts are used.

6.6.3.17 Subpictures (SUBPIC)

6.6.3.17.1 Test bitstream SUBPIC_A

Specification: This bitstream exercises several different subpicture layouts.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode when a variety of subpicture layouts are used.

6.6.3.17.2 Test bitstream SUBPIC_B

Specification: This bitstream exercises several different subpicture and slice layouts.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode when a variety of subpicture layouts with slices are used.

6.6.3.17.3 Test bitstream SUBPIC_C

Specification: This bitstream exercises signalling of equal size subpictures where each subpicture includes only one tile, slice and CTU. Parameters for RPL, deblocking, SAO, ALF, weighted prediction and delta QP are signalled in the PH.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode bitstreams with equal size subpictures and parameter information in the PH.

6.6.3.17.4 Test bitstream SUBPIC_D

Specification: This bitstream exercises signalling of subpictures with varying subpicture IDs using multiple PPSs in a way that demonstrates a panning of the field-of-view.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode subpictures when subpicture IDs vary in the CVS.

6.6.3.17.5 Test bitstream SUBPIC_E

Specification: Bitstream exercises deblocking control features across subpicture boundaries.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode bitstreams when a variety of subpicture layouts and deblocking control features are used.

6.6.3.18 Picture header and slice header (PHSH)

6.6.3.18.1 Test bitstream PHSH_B

Specification: This bitstreams includes two CVS. In the first CVS, the picture header is included in the slice header. In the second CVS, the picture header is in its own NAL unit.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode a picture header whether included in a slice header or not.

6.6.3.19 Temporal scalability (TEMPSCAL)

6.6.3.19.1 Test bitstream TEMPSCAL_A

Specification: This bitstream has 6 temporal sublayers with a GOP size of 32.

Functional stage: High-level syntax processing / GOP processing.

Purpose: Check that the decoder can handle a GOP of 32.

6.6.3.19.2 Test bitstream TEMPSCAL_B

Specification: This bitstream has 5 temporal sublayers with a GOP size of 16 and HRD signalling (including buffering period and picture timing SEI message with timing) for all temporal sublayers.

Functional stage: High-level syntax processing / Hypothetical Reference Decoder.

Purpose: Check the hypothetical reference decoder for temporal sublayers.

6.6.3.19.3 Test bitstream TEMPSCAL_C

Specification: This bitstream has 3 temporal sublayers with a GOP size of 6.

Functional stage: High-level syntax processing / GOP processing.

Purpose: Check that the decoder can properly decode bitstreams with various hierarchy structures.

6.6.3.20 Inter-layer reference picture lists (ILRPL)

6.6.3.20.1 Test bitstream ILRPL_A

Specification: This bitstream contains two layers, with layer 1 referencing layer 0.

Functional stage: High-level syntax processing/scalability.

Purpose: Check that the decoder can properly decode bitstreams using inter layer reference prediction.

6.6.3.21 Reference picture lists (RPL)

6.6.3.21.1 Test bitstream RPL_A

Specification: RPL in SPS, PH and SH. Active and inactive entries. Maximum RPL length.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode and handle reference picture lists with short term references.

6.6.3.22 Long term ref picture (LTRP)

6.6.3.22.1 Test bitstream LTRP_A

Specification: LTRP handling and picture marking.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode and handle reference picture lists with short and long term references.

6.6.3.23 Number of active ref pics (ACTPIC)

6.6.3.23.1 Test bitstream ACTPIC_A

Specification: This bitstream contains 1 active picture in list 0, and 1 active picture in list1.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode bitstreams containing various numbers of active reference pictures.

6.6.3.23.2 Test bitstream ACTPIC_B

Specification: This bitstream contains 1 active picture in list 0, and up to 2 active frames in list 1.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode bitstreams containing various numbers of active reference pictures.

6.6.3.23.3 Test bitstream ACTPIC_C

Specification: This bitstream contains 1 active picture in list 0, and up to 2 active frames in list 1.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode bitstreams containing various numbers of active reference pictures.

6.6.3.24 Virtual boundaries (VIRTUAL)

6.6.3.24.1 Test bitstream VIRTUAL_A

Specification: Virtual boundaries are enabled and signalled in SPS with 3 vertical and 2 horizontal virtual boundaries.

Functional stage: High-level syntax processing and in-loop filter process.

Purpose: Check that the decoder can properly decode and handle the virtual boundaries signalled in the SPS.

6.6.3.24.2 Test bitstream VIRTUAL_B

Specification: Virtual boundaries are enabled and signalled in PH only for pictures with an odd POC value. When virtual boundaries are enabled in a picture, 2 vertical and 1 horizontal virtual boundaries are applied.

Functional stage: High-level syntax processing and in-loop filter process.

Purpose: Check that the decoder can properly decode and handle the virtual boundaries signalled in the PH.

6.6.3.25 Reference wraparound (WRAP)

6.6.3.25.1 Test bitstream WRAP_A

Specification: This bitstream uses reference wraparound mode with content using the PERP format for 360° video in random access mode.

Functional stage: Inter prediction.

Purpose: Check that the decoder can properly decode bitstreams using reference wraparound.

6.6.3.25.2 Test bitstream WRAP_B

Specification: This bitstream uses reference wraparound mode with content using the PERP format for 360° video in random access mode.

Functional stage: Inter prediction.

Purpose: Check that the decoder can properly decode bitstreams using reference wraparound.

6.6.3.25.3 Test bitstream WRAP_C

Specification: This bitstream uses reference wraparound mode with content using the PERP format for 360° video in low delay B mode.

Functional stage: Inter prediction.

Purpose: Check that the decoder can properly decode bitstreams using reference wraparound.

6.6.3.25.4 Test bitstream WRAP_D

Specification: This bitstream uses reference wraparound mode with content using the PERP format for 360° video in low delay B mode.

Functional stage: Inter prediction.

Purpose: Check that the decoder can properly decode bitstreams using reference wraparound.

6.6.3.26 360-degree video (CUBEMAP, ERP)

6.6.3.26.1 Test bitstream CUBEMAP_A

Specification: A generalized cubemap projection SEI message is signalled in the bitstream with packing type equal to 2, mapping function equal to 2, and guard band flag equal to 0 to indicate that the coded pictures are 3x2-packed non-uniform cubemap projected pictures without guard bands.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can parse the generalized cubemap projection SEI message.

6.6.3.26.2 Test bitstream CUBEMAP_B

Specification: A generalized cubemap projection SEI message is signalled in the bitstream with packing type equal to 3, mapping function equal to 1, and guard band flag equal to 1 to indicate that the coded pictures are 6x1-packed equal-angular cubemap projected pictures with guard bands.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can parse the generalized cubemap projection SEI message.

6.6.3.26.3 Test bitstream CUBEMAP_C

Specification: A generalized cubemap projection SEI message is signalled in the bitstream with packing type equal to 4, mapping function equal to 0, and guard band flag equal to 1 to indicate that the coded pictures are 5x1-packed hemicubemap projected pictures with guard bands.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can parse the generalized cubemap projection SEI message.

6.6.3.26.4 Test bitstream ERP_A

Specification: An equirectangular projection SEI message is signalled in the bitstream with guard band flag equal to 1 to indicate that the coded pictures are equirectangular projected pictures with guard bands on the left and right sides.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can parse the equirectangular projection SEI message.

6.6.3.27 Conformance cropping window (CROP)

6.6.3.27.1 Test bitstream CROP_A

Specification: This bitstream uses large offsets for the conformance cropping window that are not aligned with CTU boundaries.

Functional stage: High-level syntax processing / conformance cropping.

Purpose: Check that the decoder outputs the correct conformance cropped region.

6.6.3.27.2 Test bitstream CROP_B

Specification: The bitstream uses odd offset values for the conformance cropping window.

Functional stage: High-level syntax processing / conformance cropping.

Purpose: Check that the decoder outputs the correct conformance cropped region.

6.6.3.28 Bumping (BUMP)

6.6.3.28.1 Test bitstream BUMP_A

Specification: This bitstream exercises bumping process in regard to the setting of DPB size and reordering. The bitstream is generated with optimized values for dpb_max_dec_pic_buffering_minus1[i] and dpb_max_num_reorder_pics[i].

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode bitstreams exercising the DPB bumping process.

6.6.3.28.2 Test bitstream BUMP_B

Specification: This bitstream exercises bumping process in regard to the setting of DPB size and reordering. The bitstream is generated with optimized value for dpb_max_num_reorder_pics[i] but with DPB size that is more than the optimal one.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode bitstreams exercising the DPB bumping process.

6.6.3.28.3 Test bitstream BUMP_C

Specification: This bitstream exercises bumping process in regard to the setting of DPB size and reordering. The bitstream is generated with non-optimal values for both dpb_max_dec_pic_buffering_minus1[i] and dpb_max_num_reorder_pics[i].

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode bitstreams exercising the DPB bumping process.

6.6.3.29 Decoded picture buffer (DPB)

6.6.3.29.1 Test bitstream DPB_A

Specification: This bitstream signals sublayer decoded picture buffer (DPB) sizes for multiple sublayers.

Functional stage: High-level syntax processing.

Purpose: Check that decoder can properly decode bitstreams exercising various DPB parameters.

6.6.3.29.2 Test bitstream DPB_B

Specification: This bitstream signals sublayer decoded picture buffer (DPB) size for a single sublayer.

Functional stage: High-level syntax processing.

Purpose: Check that the decoder can properly decode bitstreams exercising various DPB parameters.

6.6.3.30 Field pictures (FIELD)

6.6.3.30.1 Test bitstream FIELD_A

Specification: This bitstream, contains frame-field information SEI messages for each picture indicating either it is top or bottom field coded.

Functional stage: Test field coding when sps_field_seq_flag is equal to 1.

Purpose: Check that the decoder can properly decode pictures coded in field coding.

6.6.3.30.2 Test bitstream FIELD_B

Specification: This bitstream has field_seq_flag in SPS is equal to 1 contains frame-field information SEI messages. Different values of vui_chroma_sample_loc_type_top_field and vui_chroma_sample_loc_type_bottom_field are tested.

Functional stage: Test field coding when sps_field_seq_flag is equal to 1.

Purpose: Check that the decoder can properly decode pictures coded in field coding.

6.6.4 Test bitstreams – Additional chroma formats and bit depths for Main 10 profile

6.6.4.1 8 bit 4:0:0 (8b400)

6.6.4.1.1 Test bitstream 8b400_A

Specification: 8-bit 4:0:0 bitstream at a low resolution.

Functional stage: Additional chroma format and bit depth setting.

Purpose: Check that the decoder can properly handle the chroma format of 4:0:0 and InternalBitDepth of 8.

6.6.4.1.2 Test bitstream 8b400_B

Specification: 8-bit 4:0:0 bitstream at a higher resolution.

Functional stage: Additional chroma format and bit depth setting.

Purpose: Check that the decoder can properly handle the chroma format of 4:0:0 and InternalBitDepth of 8.

6.6.4.2 8 bit 4:2:0 (8b420)

6.6.4.2.1 Test bitstream 8b420_A

Specification: 8-bit 4:2:0 bitstream at a low resolution.

Functional stage: Additional bit depth setting.

Purpose: Check that the decoder can properly handle the InternalBitDepth of 8.

6.6.4.2.2 Test bitstream 8b420_B

Specification: 8-bit 4:2:0 bitstream at a higher resolution.

Functional stage: Additional bit depth setting.

Purpose: Check that the decoder can properly handle the InternalBitDepth of 8.

6.6.5 Test bitstreams – Coding tools for Main 10 4:4:4 profile for 4:4:4 chroma format and 10 bit depth

6.6.5.1 10-bit 4:4:4 (10b444)

6.6.5.1.1 Test bitstream 10b444_A

Specification: The bitstream is 10-bit 4:4:4, in all intra mode, and does not enable any 4:4:4-specific tools.

Functional stage: Decoder.

Purpose: Check that the decoder can properly decode 10-bit 4:4:4 content.

6.6.5.1.2 Test bitstream 10b444_B

Specification: The bitstream is 10-bit 4:4:4, in random access mode, and does not enable any 4:4:4-specific tools.

Functional stage: Decoder.

Purpose: Check that the decoder can properly decode 10-bit 4:4:4 content.

6.6.5.2 Adaptive colour transform (ACT)

6.6.5.2.1 Test bitstream ACT_A

Specification: This bitstream tests ACT with CU level adaptation of the colour spaces between RGB and YCoCg, in random access mode. The bitstream is Main 10 4:4:4 profile, Main tier, Level 6.

Functional stage: Adaptive colour transform.

Purpose: Check that the decoder can properly decode a bitstream with ACT enabled.

6.6.5.2.2 Test bitstream ACT_B

Specification: This bitstream tests ACT with CU level adaptation of the colour spaces between RGB and YCoCg, in all intra mode. The bitstream is Main 10 4:4:4 profile, Main tier, Level 6.

Functional stage: Adaptive colour transform.

Purpose: Check that the decoder can properly decode a bitstream with ACT enabled.

6.6.5.3 Palette mode (PALETTE)

6.6.5.3.1 Test bitstream PALETTE_A

Specification: This bitstream forces the majority of the CUs to be coded using palette mode in a Random Access configuration. The bitstream conforms to the Main 10 4:4:4 profile, Main tier.

Functional stage: Palette.

Purpose: Check that the decoder can properly decode a bitstream with palette mode enabled.

6.6.5.3.2 Test bitstream PALETTE_B

Specification: This bitstream forces the majority of the CUs to be coded using palette mode in an All Intra configuration. The bitstream conforms to the Main 10 4:4:4 profile, Main tier.

Functional stage: Palette.

Purpose: Check that the decoder can properly decode a bitstream with palette mode enabled.

6.6.5.3.3 Test bitstream PALETTE_C

Specification: This bitstream forces the majority of the CUs to be coded using palette mode in a Low Delay B configuration. The bitstream conforms to the Main 10 4:4:4 profile, Main tier.

Functional stage: Palette.

Purpose: Check that the decoder can properly decode a bitstream with palette mode enabled.

6.6.5.3.4 Test bitstream PALETTE_D

Specification: This bitstream forces the majority of the CUs to be coded using palette mode in an All Intra configuration. The bitstream conforms to the Main 10 4:4:4 profile, Main tier.

Functional stage: Palette.

Purpose: Check that the decoder can properly decode a bitstream with palette mode enabled.

6.6.5.3.5 Test bitstream PALETTE_E

Specification: This bitstream forces the majority of the CUs to be coded using palette mode in an All Intra configuration. The bitstream use 4:2:0 chroma format but conforms to the Main 10 4:4:4 profile, Main tier, Level 3, which supports use of palette mode for 4:2:0.

Functional stage: Palette.

Purpose: Check that the decoder can properly decode a bitstream with palette mode enabled.

6.6.5.4 Entropy coding (ENT444MAINTIER)

6.6.5.4.1 Test bitstream ENT444MAINTIER_A

Specification: The bitstream contains 3 independent CVSs containing one picture, formed from one slice. Each CVS contains the maximum number of bits, given the profile, level and tier, and assuming equal distribution of bits between pictures coded at the maximum luma sample rate for the level, at Level 4. The 3 concatenated CVSs are as follows:

— The first picture does not require any CABAC zero words.

— The second picture requires one CABAC zero word.

— The third picture requires a substantial quantity of CABAC zero words (75 % of the bitstream is padding).

All 3 pictures have a very low subjective quality level due to the artificial nature of these bitstreams. All VCL NAL units contain almost (within 3 bytes worth) of the maximum number of allowed bins, for their size.

Functional stage: Entropy coding.

Purpose: Check that the decoder properly decodes all entropy coding modes.

6.6.5.4.2 Test bitstream ENT444MAINTIER_B

Specification: The bitstream contains 3 independent CVSs containing one picture, formed from one slice. Each CVS contains the maximum number of bits, given the profile, level and tier, and assuming equal distribution of bits between pictures coded at the maximum luma sample rate for the level, at Level 4.1. The 3 concatenated CVSs are as follows:

— The first picture does not require any CABAC zero words.

— The second picture requires one CABAC zero word.

— The third picture requires a substantial quantity of CABAC zero words (75 % of the bitstream is padding).

All 3 pictures have a very low subjective quality level due to the artificial nature of these bitstreams. All VCL NAL units contain almost (within 3 bytes worth) of the maximum number of allowed bins, for their size.

Functional stage: Entropy coding.

Purpose: Check that the decoder properly decodes all entropy coding modes.

6.6.5.4.3 Test bitstream ENT444MAINTIER_C

Specification: The bitstream contains 3 independent CVSs containing one picture, formed from one slice. Each CVS contains the maximum number of bits, given the profile, level and tier, and assuming

equal distribution of bits between pictures coded at the maximum luma sample rate for the level, at Level 5. The 3 concatenated CVSs are as follows:

— The first picture does not require any CABAC zero words.

— The second picture requires one CABAC zero word.

— The third picture requires a substantial quantity of CABAC zero words (75 % of the bitstream is padding).

All 3 pictures have a very low subjective quality level due to the artificial nature of these bitstreams. All VCL NAL units contain almost (within 3 bytes worth) of the maximum number of allowed bins, for their size.

Functional stage: Entropy coding.

Purpose: Check that the decoder properly decodes all entropy coding modes.

6.6.5.4.4 Test bitstream ENT444MAINTIER_D

Specification: The bitstream contains 3 independent CVSs containing one picture, formed from one slice. Each CVS contains the maximum number of bits, given the profile, level and tier, and assuming equal distribution of bits between pictures coded at the maximum luma sample rate for the level, at Level 5.1. The 3 concatenated CVSs are as follows:

— The first picture does not require any CABAC zero words.

— The second picture requires one CABAC zero word.

— The third picture requires a substantial quantity of CABAC zero words (75 % of the bitstream is padding).

All 3 pictures have a very low subjective quality level due to the artificial nature of these bitstreams. All VCL NAL units contain almost (within 3 bytes worth) of the maximum number of allowed bins, for their size.

Functional stage: Entropy coding.

Purpose: Check that the decoder properly decodes all entropy coding modes.

6.6.5.5 Entropy coding (ENT444HIGHTIER)

6.6.5.5.1 Test bitstream ENT444HIGHTIER_A

Specification: The bitstream contains 3 independent CVSs containing one picture, formed from one slice. Each CVS contains the maximum number of bits, given the profile, level and tier, and assuming equal distribution of bits between pictures coded at the maximum luma sample rate for the level, at Level 4 for High tier. The 3 concatenated CVSs are as follows:

— The first picture does not require any CABAC zero words.

— The second picture requires one CABAC zero word.

— The third picture requires a substantial quantity of CABAC zero words (75 % of the bitstream is padding).

All 3 pictures have a very low subjective quality level due to the artificial nature of these bitstreams. All VCL NAL units contain almost (within 3 bytes worth) of the maximum number of allowed bins, for their size.

Functional stage: Entropy coding.

Purpose: Check that the decoder properly decodes all entropy coding modes.

6.6.5.5.2 Test bitstream ENT444HIGHTIER_B

Specification: The bitstream contains 3 independent CVSs containing one picture, formed from one slice. Each CVS contains the maximum number of bits, given the profile, level and tier, and assuming equal distribution of bits between pictures coded at the maximum luma sample rate for the level, at Level 4.1 for High tier. The 3 concatenated CVSs are as follows:

— The first picture does not require any CABAC zero words.

— The second picture requires one CABAC zero word.

— The third picture requires a substantial quantity of CABAC zero words (75 % of the bitstream is padding).

All 3 pictures have a very low subjective quality level due to the artificial nature of these bitstreams. All VCL NAL units contain almost (within 3 bytes worth) of the maximum number of allowed bins, for their size.

Functional stage: Entropy coding.

Purpose: Check that the decoder properly decodes all entropy coding modes.

6.6.5.5.3 Test bitstream ENT444HIGHTIER_C

Specification: The bitstream contains 3 independent CVSs containing one picture, formed from one slice. Each CVS contains the maximum number of bits, given the profile, level and tier, and assuming equal distribution of bits between pictures coded at the maximum luma sample rate for the level, at Level 5 for High tier. The 3 concatenated CVSs are as follows:

— The first picture does not require any CABAC zero words.

— The second picture requires one CABAC zero word.

— The third picture requires a substantial quantity of CABAC zero words (75 % of the bitstream is padding).

All 3 pictures have a very low subjective quality level due to the artificial nature of these bitstreams. All VCL NAL units contain almost (within 3 bytes worth) of the maximum number of allowed bins, for their size.

Functional stage: Entropy coding.

Purpose: Check that the decoder properly decodes all entropy coding modes.

6.6.5.5.4 Test bitstream ENT444HIGHTIER_D

Specification: The bitstream contains 3 independent CVSs containing one picture, formed from one slice. Each CVS contains the maximum number of bits, given the profile, level and tier, and assuming equal distribution of bits between pictures coded at the maximum luma sample rate for the level, at Level 5.1 for High tier. The 3 concatenated CVSs are as follows:

— The first picture does not require any CABAC zero words.

— The second picture requires one CABAC zero word.

— The third picture requires a substantial quantity of CABAC zero words (75 % of the bitstream is padding).