 ISO/IEC JTC 1/SC 29/WG 5 N0041

**ISO/IEC JTC 1/SC 29/WG 5**

**MPEG Joint Video Coding Team(s) with ITU-T SG 16**

**Convenorship: DE**

**Document type:** General

**Title:** Exploration experiment on enhanced compression beyond VVC capability

**Status:** Approved

**Date of document:** 2021-01-15

**Source:** ISO/IEC JTC 1/SC 29/WG 5

**Expected action:** Info

**Action due date:** None

**No. of pages: 19** (without cover page)

**Email of Convenor:** ohm @ ient . rwth-aachen . de

**Committee URL:** https://isotc.iso.org/livelink/livelink/open/jtc1sc29wg5

|  |  |
| --- | --- |
| **Joint Video Experts Team (JVET)**  **of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29**  21st Meeting, by teleconference, 6–15 Jan. 2021 | Document: JVET-U2024-v2 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **Exploration Experiment on Enhanced Compression beyond VVC capability** | | |
| *Status:* | Output document to JVET | | |
| *Purpose:* | EE description | | |
| *Author(s) or Contact(s):* | Vadim Seregin  Jie Chen  Semih Esenlik  Fabrice Le Leannec  Ling Li  Martin Winken  Jacob Strom  Xiaoyu Xiu  Kai Zhang | Tel: Email: | [vseregin@qti.qualcomm.com](mailto:vseregin@qti.qualcomm.com)  [jiechen.cj@alibaba-inc.com](mailto:jiechen.cj@alibaba-inc.com)  [semih.esenlik@huawei.com](mailto:semih.esenlik@huawei.com)  [fabrice.leleannec@interdigital.com](mailto:fabrice.leleannec@interdigital.com)  [aurali@tencent.com](mailto:aurali@tencent.com)  [jacob.strom@ericsson.com](mailto:jacob.strom@ericsson.com)  [martin.winken@hhi.fraunhofer.de](mailto:martin.winken@hhi.fraunhofer.de)  [xiaoyuxiu@kwai.com](mailto:xiaoyuxiu@kwai.com)  [zhangkai.video@bytedance.com](mailto:zhangkai.video@bytedance.com) |
| *Source:* | EE coordinators | | |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Abstract

This document describes Exploration Experiments (EEs) planned to be performed between 21st and 22nd JVET meetings to evaluate enhanced compression tools beyond VVC capability.

# Introduction

EE focus is to evaluate individual coding technologies and their combinations. Contributions improving compression efficiency further is highly encouraged.

JVET-U0100 based on VTM-10.0 will be released to EE software branch and will be used as EE common software base for tool on and tool off tests. Technologies not included into JVET-U0100 can be implemented on top of VTM, but it is encouraged to have it integrated into the EE common software base.

Cross-check for EE common software base (all tools on and all tools off) is mandatory, while for other EE tests, cross-checks are encouraged, but not required.

EE related discussion shall happen on JVET and JVET-CE reflectors.

# Test conditions and evaluation criteria

Tests shall be performed according to VTM CTC described in JVET-T2010 with additional TGM 4:2:0 optional class. TGM tests are required for testing SCC tools.

AI and RA test configurations are required for intra tool testing, while RA and LB test configurations are required for inter tool testing. LP configuration is optional.

Tool on and tool off tests are performed in the EE. In tool on tests VTM-11.0 is used as an anchor and EE common software base is used as a reference in the tool off testing. Tool on tests are required.

# Timeline

**T1** = 2 weeks after JVET meeting (Jan. 29, 2021): JVET-U0100 software as EE common software base is released to EE software branch

**T2** = T1 + 2 weeks (Feb. 12, 2021): EE description is finalized and EE common software base is frozen at this time.

It is encouraged to provide partial test results for EE common software base.

**T3** = T2 + 2 weeks (Feb. 26, 2021): Initial software release for EE tests.

JVET-U0048, bilateral filter from JVET-P0073 with bilateral filter elements of JVET-P0078

**T4** = 22nd JVET meeting – 3 weeks (Mar. 30, 2021): Software in EE branches is frozen

It is encouraged to make partial test results available earlier.

# List of tools

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Tests** | **Tester** | **Cross-checker** |
| **1 Base** | |  |  |
| 1.1 | Non-normative changes introduced by macros and cfg file parameter changes |  |  |
| 1.2 | Normative changes introduced by macros |  |  |
| 1.3 | Non-normative and normative changes (1.1+1.2) |  |  |
| **2 Intra prediction** | |  |  |
| 2.1 | Intra template matching (JVET-U0048) |  |  |
| 2.2 | Multi-model LM (MMLM) |  |  |
| 2.3 | Gradient PDPC |  |  |
| 2.4 | Secondary MPM |  |  |
| 2.5 | Reference sample interpolation and smoothing for intra-prediction |  |  |
| 2.6 | Decoder side intra mode derivation (DIMD) |  |  |
| **3 Inter prediction** | |  |  |
| 3.1 | Local illumination compensation (LIC) |  |  |
| 3.2 | Non-adjacent spatial candidates |  |  |
| 3.3 | Template matching (TM) |  | S. Esenlik (Huawei) |
| 3.4 | Multi-pass decoder-side motion vector refinement |  | S. Esenlik (Huawei) |
| 3.5 | Multi-pass decoder-side motion vector refinement with sample-based BDOF |  | S. Esenlik (Huawei) |
| 3.6 | OBMC |  |  |
| 3.7 | 12-tap interpolation |  |  |
| 3.8.a | Multi-hypothesis prediction (MHP) |  |  |
| 3.8.b | MHP without the restriction on non-equal weight in BCW |  |  |
| 3.9 | Affine MMVD |  |  |
| 3.10 | CIIP with PDPC blending |  |  |
| **4 Transform and coefficient coding** | |  |  |
| 4.1 | 8-states DQ |  |  |
| 4.2 | Extended LFNST |  |  |
| 4.3 | Transform sign prediction |  |  |
| **5 In-loop filtering** | |  |  |
| 5.1 | ALF with multiple fixed filters |  |  |
| 5.2 | Bilateral filter |  |  |
| **6 Entropy coding** | |  |  |
| 6.1 | Extended precision |  |  |
| 6.2 | Extended precision with slice type based CABAC window |  |  |
| **7 Combinations** | |  |  |
| 7.1 | Intra prediction (2.2+…+2.6) |  |  |
| 7.2 | Template + bilateral matching (3.3 + 3.5) |  | S. Esenlik (Huawei) |
| 7.4 | JVET-U0100 (EE common software) |  |  |

# Tools description

## Base

### Non-normative base

The following non-normative changes are introduced in the EE base software.

* Lossy encoder speedups for affine, amvr and merge, macros: AFFINE\_ENC\_OPT, AMVR\_ENC\_OPT, MERGE\_ENC\_OPT
* Use original samples for SAO and ALF with macro ALF\_SAO\_TRUE\_ORG
* Old PCM code removal for speedup, macro: REMOVE\_PCM
* More SIMD, macros: MCIF\_SIMD\_NEW, DIST\_SSE\_ENABLE, TRANSFORM\_SIMD\_OPT

In config files, e.g. cfg\_encoder\_randomaccess\_vtm.cfg

* GOP32 as in VTM-11.0 (but without the reference frame fix in VTM-11.1)
* MCTF enabled
* MinQTNonISlice is changed from 8 to 16

### Normative base

The following normative changes are introduced under the BASE macro in the EE base software. They either extend VVC capability or remove restrictions:

// Partition

#define CTU\_256 1 // Add CTU 256

#if CTU\_256

#define TU\_256 1 // Add TU 256, removed MTS zero out

#endif

#define REMOVE\_VPDU 1 // Remove VPDU restriction on BT/TT splitting

#if TU\_256

#define LMCS\_CHROMA\_CALC\_CU 1 // Derive chroma LMCS parameter based on neighbor CUs. Needed by VPDU removal and 128x128 transform.

#endif

//-- intra

#define INTRA\_RM\_SMALL\_BLOCK\_SIZE\_CONSTRAINTS 1 // Enable 2xN and Nx2 block by removing SCIPU constraints

#define CCLM\_LATENCY\_RESTRICTION\_RMV 1 // remove the latency between luma and chroma restriction of CCLM

#define LMS\_LINEAR\_MODEL 1 // LMS for parameters derivation of CCLM and MMLM mode, Remove constraint in derivation of neighbouring samples

//-- inter

#define CIIP\_RM\_BLOCK\_SIZE\_CONSTRAINTS 1 // Remove the 64x64 restriction and enable 8x4/4x8 block for CIIP

#define BDOF\_RM\_CONSTRAINTS 1

#define INTER\_RM\_SIZE\_CONSTRAINTS 1 // Remove size constraints for inter block

#define AFFINE\_RM\_CONSTRAINTS\_AND\_OPT 1 // Remove affine constraints and optimization

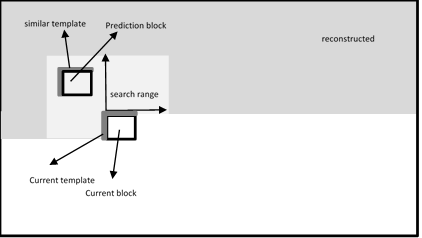
//-- loop filters

#define DB\_PARAM\_TID 1 // adjust DB parameters based on temporal ID

## Intra prediction

### Intra template matching

As proposed in JVET-U0048, intra predictor is derived by searching the closest match of the L shape template in the already reconstructed area of a picture within a predefined search range.



Various search range values will be tested in this EE test, as follows.

***List of tests to be performed***

|  |  |  |
| --- | --- | --- |
| # | Test | Tester |
| 1 | Intra TMP with search range equal to 64 | InterDigital |
| 2 | Intra TMP with search range equal to 128 | InterDigital |
| 3 | Intra TMP with search range equal to 256 | InterDigital |

Furthermore, various values of the newly introduced MaxTMPSize coding parameter may be tested.

### Multi-model LM (MMLM)

CCLM included in VVC is extended by adding three Multi-model LM (MMLM) modes as proposed in JVET-D0110/JVET-U0100. In each MMLM mode, the reconstructed neighboring samples are classified into two classes using a threshold which is the average of the luma reconstructed neighboring samples. The linear model of each class is derived using the Least-Mean-Square (LMS) method. For the CCLM mode, the LMS method is also used to derive the linear model.

The smallest chroma intra prediction unit (SCIPU) constraint is removed. In addition, the VPDU constraint for reducing CCLM prediction latency is also removed.

### Gradient PDPC

In VVC, for a few scenarios, PDPC may not be applied due to the unavailability of the secondary reference samples. As proposed in JVET-Q0391/JVET-U0100, in these cases, a gradient based PDPC, extended from horizontal/vertical mode, is applied. The PDPC weights (wT / wL) and nScale parameter for determining the decay in PDPC weights with respect to the distance from left/top boundary are set equal to corresponding parameters in horizontal/vertical mode, respectively. When the secondary reference sample is at a fractional sample position, bilinear interpolation is applied.

### Secondary MPM

Secondary MPM lists is introduced as described in JVET-D0114/JVET-U0100.The existing primary MPM (PMPM) list consists of 6 entries and the secondary MPM (SMPM) list includes 16 entries. A general MPM list with 22 entries is constructed first, and then the first 6 entries in this general MPM list are included into the PMPM list, and the rest of entries form the SMPM list. The first entry in the general MPM list is the Planar mode. The remaining entries are composed of the intra modes of the left (L), above (A), below-left (BL), above-right (AR), and above-left (AL) neighbouring blocks as shown in Figure 1, the directional modes with added offset from the first two available directional modes of neighbouring blocks, and the default modes.

If a CU block is vertically oriented, the order of neighbouring blocks is A, L, BL, AR, AL; otherwise, it is L, A, BL, AR, AL.

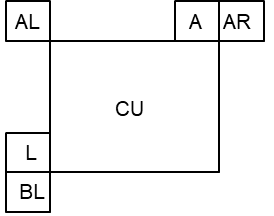


Figure 1. Neighbouring blocks (L, A, BL, AR, AL) used in the derivation of a general MPM list.

A PMPM flag is parsed first, if equal to 1 then a PMPM index is parsed to determine which entry of the PMPM list is selected, otherwise the SPMPM flag is parsed to determine whether to parse the SMPM index or the remaining modes.

### Reference sample interpolation and smoothing for intra-prediction

Three modifications are proposed. First, the 4-tap cubic interpolation is replaced with a 6-tap cubic interpolation filter, as described in JVET-D0119/JVET-U0100, for the derivation of predicted samples from the reference samples. Second, for the coding blocks where a gaussian interpolation filter is applied to filter the reference samples, a 6-tap gaussian filter is applied for larger blocks (W >= 32 and H >=32) and the existing VVC 4-tap gaussian interpolation filter is applied otherwise. Third, to derive the extended intra reference samples, the 4-tap interpolation filter is used instead of the nearest neighbor rounding.

### Decoder side intra mode derivation (DIMD)

When DIMD is applied, two intra modes are derived from the reconstructed neighbor samples, and those two predictors are combined with the planar mode predictor with the weights derived from the gradients as described in JVET-O0449/JVET-U0100.

Derived intra modes are included into the primary list of intra most probable modes (MPM), so the DIMD process is performed before the MPM list is constructed. The primary derived intra mode of a DIMD block is stored with a block and is used for MPM list construction of the neighboring blocks.

## Inter prediction

### Local illumination compensation (LIC)

LIC is an inter prediction technique to model local illumination variation between current block and its prediction block as a function of that between current block template and reference block template. The parameters of the function can be denoted by a scale *α* and an offset *β*, which forms a linear equation, that is, *α*\*p[x]+*β* to compensate illumination changes, where p[x] is a reference sample pointed to by MV at a location x on reference picture. Since *α* and *β* can be derived based on current block template and reference block template, no signaling overhead is required for them, except that an LIC flag is signaled for AMVP mode to indicate the use of LIC.

The local illumination compensation that was previously proposed in JVET-O0066 was included in JVET-U0100 for uni-prediction inter CUs with the following modifications.

* Intra neighbor samples can be used in LIC parameter derivation;
* LIC is disabled for blocks with less than 32 luma samples;
* For both non-subblock and affine modes, LIC parameter derivation is performed based on the template block samples corresponding to the current CU, instead of partial template block samples corresponding to first top-left 16x16 unit;
* Samples of the reference block template are generated by using MC with the block MV without rounding it to integer-pel precision.

### Non-adjacent spatial candidate

The non-adjacent spatial merge candidates as in JVET-L0399/JVET-U0100 are inserted after the TMVP in the regular merge candidate list. The pattern of spatial merge candidates is shown in Figure 3. The distances between non-adjacent spatial candidates and current coding block are based on the width and height of current coding block. The line buffer restriction is not applied.



Figure 3. Spatial neighboring blocks used to derive the spatial merge candidates

### Template matching (TM)

Template matching (TM) is a decoder-side MV derivation method to refine the motion information of the current CU by finding the closest match between a template (i.e., top and/or left neighbouring blocks of the current CU) in the current picture and a block (i.e., same size to the template) in a reference picture. As illustrated in Figure 4, a better MV is to be searched around the initial motion of the current CU within a [– 8, +8]-pel search range. The template matching that was previously proposed in JVET-J0021 is included in JVET-U0100 with two modifications: search step size is determined based on AMVR mode and TM can be cascaded with bilateral matching process in merge modes.

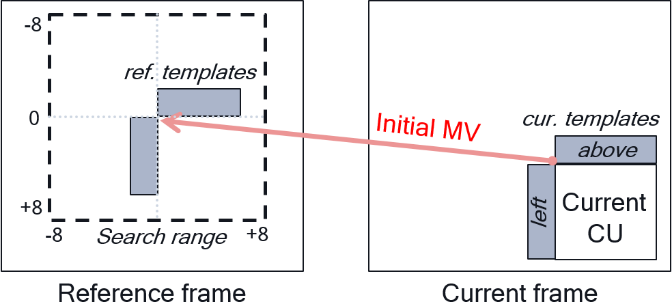


Figure 4. Template matching performs on a search area around initial MV.

In AMVP mode, an MVP candidate is determined based on template matching error to pick up the one which reaches the minimum difference between current block template and reference block template, and then TM performs only for this particular MVP candidate for MV refinement. TM refines this MVP candidate, starting from full-pel MVD precision (or 4-pel for 4-pel AMVR mode) within a [–8, +8]-pel search range by using iterative diamond search. The AMVP candidate may be further refined by using cross search with full-pel MVD precision (or 4-pel for 4-pel AMVR mode), followed sequentially by half-pel and quarter-pel ones depending on AMVR mode as specified in Table 1. This search process ensures that the MVP candidate still keeps the same MV precision as indicated by AMVR mode after TM process.

Table 1. Search patterns of AMVR and merge mode with AMVR.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Search pattern** | **AMVR mode** | | | | **Merge mode** | |
| **4-pel** | **Full-pel** | **Half-pel** | **Quarter-pel** | **AltIF=0** | **AltIF=1** |
| 4-pel diamond | v |  |  |  |  |  |
| 4-pel cross | v |  |  |  |  |  |
| Full-pel diamond |  | v | v | v | v | v |
| Full-pel cross |  | v | v | v | v | v |
| Half-pel cross |  |  | v | v | v | v |
| Quarter-pel cross |  |  |  | v | v |  |
| 1/8-pel cross |  |  |  |  | v |  |

In merge mode, similar search method is applied to the merge candidate indicated by the merge index. As Table 1 shows, TM may perform all the way down to 1/8-pel MVD precision or skipping those beyond half-pel MVD precision, depending on whether the alternative interpolation filter (that is used when AMVR is of half-pel mode) is used according to merged motion information. Besides, when TM mode is enabled, template matching may work as an independent process or an extra MV refinement process between block-based and subblock-based bilateral matching (BM) methods, depending on whether BM can be enabled or not according to its enabling condition check.

### Multi-pass decoder-side motion vector refinement

In JVET-U0100, a multi-pass decoder-side motion vector refinement is applied. In the first pass, bilateral matching (BM) is applied to the coding block. In the second pass, BM is applied to each 16x16 subblock within the coding block. In the third pass, MV in each 8x8 subblock is refined by applying bi-directional optical flow (BDOF). The refined MVs are stored for both spatial and temporal motion vector prediction.

##### First pass – Block based bilateral matching MV refinement

In the first pass, a refined MV is derived by applying BM to a coding block. Similar to decoder-side motion vector refinement (DMVR), in bi-prediction operation, a refined MV is searched around the two initial MVs (MV0 and MV1) in the reference picture lists L0 and L1. The refined MVs (MV0\_pass1 and MV1\_pass1) are derived around the initiate MVs based on the minimum bilateral matching cost between the two reference blocks in L0 and L1.

BM performs local search to derive integer sample precision intDeltaMV. The local search applies a 3×3 square search pattern to loop through the search range [–sHor, sHor] in horizontal direction and [–sVer, sVer] in vertical direction, wherein, the values of sHor and sVer are determined by the block dimension, and the maximum value of sHor and sVer is 8.

The bilateral matching cost is calculated as: bilCost = mvDistanceCost + sadCost. When the block size cbW \* cbH is greater than 64, MRSAD cost function is applied to remove the DC effect of distortion between reference blocks. When the bilCost at the center point of the 3×3 search pattern has the minimum cost, the intDeltaMV local search is terminated. Otherwise, the current minimum cost search point becomes the new center point of the 3×3 search pattern and continue to search for the minimum cost, until it reaches the end of the search range.

The existing fractional sample refinement is further applied to derive the final deltaMV. The refined MVs after the first pass is then derived as:

* MV0\_pass1 = MV0 + deltaMV
* MV1\_pass1 = MV1 – deltaMV

##### Second pass – Subblock based bilateral matching MV refinement

In the second pass, a refined MV is derived by applying BM to a 16×16 grid subblock. For each subblock, a refined MV is searched around the two MVs (MV0\_pass1 and MV1\_pass1), obtained on the first pass, in the reference picture list L0 and L1. The refined MVs (MV0\_pass2(sbIdx2) and MV1\_pass2(sbIdx2)) are derived based on the minimum bilateral matching cost between the two reference subblocks in L0 and L1.

For each subblock, BM performs full search to derive integer sample precision intDeltaMV. The full search has a search range [–sHor, sHor] in horizontal direction and [– sVer, sVer] in vertical direction, wherein, the values of sHor and sVer are determined by the block dimension, and the maximum value of sHor and sVer is 8.

The bilateral matching cost is calculated by applying a cost factor to the SATD cost between two reference subblocks, as: bilCost = satdCost \* costFactor. The search area (2\*sHor + 1) \* (2\*sVer + 1) is divided up to 5 diamond shape search regions shown on Figure 5. Diamond regions in the search area.. Each search region is assigned a costFactor, which is determined by the distance (intDeltaMV) between each search point and the starting MV, and each diamond region is processed in the order starting from the center of the search area. In each region, the search points are processed in the raster scan order starting from the top left going to the bottom right corner of the region. When the minimum bilCost within the current search region is less than a threshold equal to sbW \* sbH, the int-pel full search is terminated, otherwise, the int-pel full search continues to the next search region until all search points are examined.



Figure 5. Diamond regions in the search area.

The existing VVC DMVR fractional sample refinement is further applied to derive the final deltaMV(sbIdx2) . The refined MVs at second pass is then derived as:

* MV0\_pass2(sbIdx2) = MV0\_pass1 + deltaMV(sbIdx2)
* MV1\_pass2(sbIdx2) = MV1\_pass1 – deltaMV(sbIdx2)

##### Third pass – Subblock based bi-directional optical flow MV refinement

In the third pass, a refined MV is derived by applying BDOF to an 8×8 grid subblock. For each 8×8 subblock, BDOF refinement is applied to derive scaled Vx and Vy without clipping starting from the refined MV of the parent subblock of the second pass. The derived bioMv(Vx, Vy) is rounded to 1/16 sample precision and clipped between -32 and 32.

The refined MVs (MV0\_pass3(sbIdx3) and MV1\_pass3(sbIdx3)) at third pass are derived as:

* MV0\_pass3(sbIdx3) = MV0\_pass2(sbIdx2) + bioMv
* MV1\_pass3(sbIdx3) = MV0\_pass2(sbIdx2) – bioMv

### Sample-based BDOF

In the sample-based BDOF described in JVET-U0100, instead of deriving motion refinement (Vx, Vy) on a block basis, it is performed per sample.

The coding block is divided into 8×8 subblocks. For each subblock, whether to apply BDOF or not is determined by checking the SAD between the two reference subblocks against a threshold. If decided to apply BDOF to a subblock, for every sample in the subblock, a sliding 5×5 window is used and the existing BDOF process is applied for every sliding window to derive Vx and Vy. The derived motion refinement (Vx, Vy) is applied to adjust the bi-predicted sample value for the center sample of the window.

### OBMC

When OBMC is applied, top and left boundary pixels of a CU are refined using neighboring block’s motion information with a weighted prediction as described in JVET-L0101/JVET-U0100.

Conditions of not applying OBMC are as follows:

* When OBMC is disabled at SPS level
* When current block has intra mode or IBC mode
* When current block applies LIC
* When current luma block area is smaller or equal to 32

A subblock-boundary OBMC is further proposed to apply the same blending to the top, left, bottom, and right subblock boundary pixels using neighboring subblock’s motion information. It is enabled for the subblock based coding tools:

* Affine AMVP modes;
* Affine merge modes and subblock-based temporal motion vector prediction (SbTMVP);
* Subblock-based bilateral matching.

### Interpolation

The 8-tap interpolation filter used in VVC is replaced with a 12-tap filter in JVET-U0100. The proposed interpolation filter is derived from the sinc function of which the frequency response is cut off at Nyquist frequency, and cropped by a cosine window function. Table 2 gives the filter coefficients of all 16 phases. Figure 9. compares the frequency responses of the proposed interpolation filter with the VVC interpolation filter, all at half-pel phase.

Table 2. Filter coefficients of the proposed 12-tap interpolation filter

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1/16 | -1 | 2 | -3 | 6 | -14 | 254 | 16 | -7 | 4 | -2 | 1 | 0 |
| 2/16 | -1 | 3 | -7 | 12 | -26 | 249 | 35 | -15 | 8 | -4 | 2 | 0 |
| 3/16 | -2 | 5 | -9 | 17 | -36 | 241 | 54 | -22 | 12 | -6 | 3 | -1 |
| 4/16 | -2 | 5 | -11 | 21 | -43 | 230 | 75 | -29 | 15 | -8 | 4 | -1 |
| 5/16 | -2 | 6 | -13 | 24 | -48 | 216 | 97 | -36 | 19 | -10 | 4 | -1 |
| 6/16 | -2 | 7 | -14 | 25 | -51 | 200 | 119 | -42 | 22 | -12 | 5 | -1 |
| 7/16 | -2 | 7 | -14 | 26 | -51 | 181 | 140 | -46 | 24 | -13 | 6 | -2 |
| 8/16 | -2 | 6 | -13 | 25 | -50 | 162 | 162 | -50 | 25 | -13 | 6 | -2 |
| 9/16 | -2 | 6 | -13 | 24 | -46 | 140 | 181 | -51 | 26 | -14 | 7 | -2 |
| 10/16 | -1 | 5 | -12 | 22 | -42 | 119 | 200 | -51 | 25 | -14 | 7 | -2 |
| 11/16 | -1 | 4 | -10 | 19 | -36 | 97 | 216 | -48 | 24 | -13 | 6 | -2 |
| 12/16 | -1 | 4 | -8 | 15 | -29 | 75 | 230 | -43 | 21 | -11 | 5 | -2 |
| 13/16 | -1 | 3 | -6 | 12 | -22 | 54 | 241 | -36 | 17 | -9 | 5 | -2 |
| 14/16 | 0 | 2 | -4 | 8 | -15 | 35 | 249 | -26 | 12 | -7 | 3 | -1 |
| 15/16 | 0 | 1 | -2 | 4 | -7 | 16 | 254 | -14 | 6 | -3 | 2 | -1 |



Figure 9. Frequency responses of the proposed interpolation filter and the VVC interpolation filter at half-pel phase

### Multi-hypothesis prediction (MHP)

The multi-hypothesis prediction previously proposed in JVET-M0425 is included in JVET-U0100. Up to two additional predictors are signalled on top of inter AMVP mode. The resulting overall prediction signal is accumulated iteratively with each additional prediction signal.

The weighting factor *α* is specified according to the following table:

|  |  |
| --- | --- |
| **add\_hyp\_weight\_idx** | ***α*** |
| 0 | 1/4 |
| 1 | -1/8 |

For inter AMVP mode, MHP is only applied if non-equal weight in BCW is selected in bi-prediction mode.

### Afiine MMVD

The affine MMVD mode was proposed in JVET-N0378. The notion of MMVD that applies a distance offset to a base candidate motion is extended to the CPMVs of affine merge mode. In Affine MMVD mode, there is only one base vector candidate which is the first affine candidate in the subblock merge candidate list, and thus there is no base vector candidate flag to be signaled. There are three sets of distance offsets to be selected based on the sequence resolution, as follows:

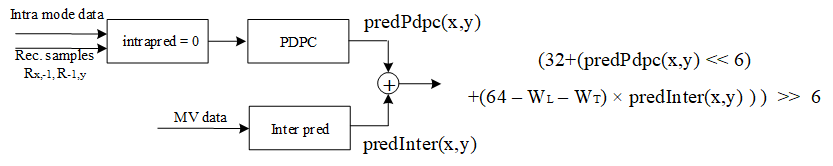
* {1/16, 1/8, 1/4, 1/2, 1} for sequences smaller than 1280x720
* {1/4, 1/2, 1, 2, 4} for sequences smaller than 2560x1600
* {1/2, 1, 2, 4, 8} for sequences larger than or equal to 2560x1600

Same as MMVD, affine MMVD also supports 4 directions (i.e., (+,0), (-,0), (0,+), (0,-)) for each of its distance offsets. When the base candidate is of uni-prediction, the offset vector (i.e., distance offset times direction) is added up equally to each CPMV. For the bi-prediction candidate, the offset vector is added up equally to L0 CPMVs, while the offset vector is firstly mirrored based on POC distance and is then added up to L1 CPMVs.

### CIIP with PDPC blending

The CIIP mode is extended as described in JVET-O0537. In this extended mode (CIIP\_PDPC), the prediction of the regular merge mode is refined using the above (Rx, -1) and left (R-1, y) reconstructed samples. This refinement inherits the position dependent prediction combination (PDPC) scheme. The flowchart of the prediction of the CIIP\_PDPC mode can be depicted in the next figure, where WT and WL are the weighted values which depend on the sample position in the block as defined in PDPC.

The CIIP\_PDPC mode is signaled together with CIIP mode. When CIIP flag is true, another flag, namely CIIP\_PDPC flag, is further signaled to indicate whether to use CIIP\_PDPC.

****

**Flowchart of the extended CIIP mode using PDPC**

## Transform and coefficient coding

### 8-state Dependent Quantization

The 8-state dependent quantization proposed in JVET-Q0243.

### Secondary Transformation: Extended LFNST

As proposed in JVET-U0100, the LFNST design in VVC is extended as follows:

* The number of LFNST sets (*S*) and candidates (*C*) are extended to *S*=35 and *C*=3, and the LFNST set (lfnstTrSetIdx) for a given intra mode (predModeIntra) is derived according to the following formula:
  + For predModeIntra < 2, lfnstTrSetIdx is equal to 2
  + lfnstTrSetIdx = predModeIntra, for predModeIntra in [0,34]
  + lfnstTrSetIdx = 68 – predModeIntra, for predModeIntra in [35,66]
* The 8x8 LFNST matrix size (i.e., the transform support) in VVC is extended to full 64x64, such that it is not reduced as no zeroing-out process is applied to secondary transform coefficients.

### Sign prediction

Following JVET-D0031, JVET-J0021, JVET-U0100, basic idea of the coefficient sign prediction method is for applicable transform coefficients to calculate reconstructed residual for both negative and positive sign combinations and select the hypothesis that minimizes a cost function.

To derive the best sign, the cost function is defined as discontinuity measure across block boundary shown on Figure 11. It is measured for all hypotheses, and the one with the smallest cost is selected as a predictor for coefficient signs.

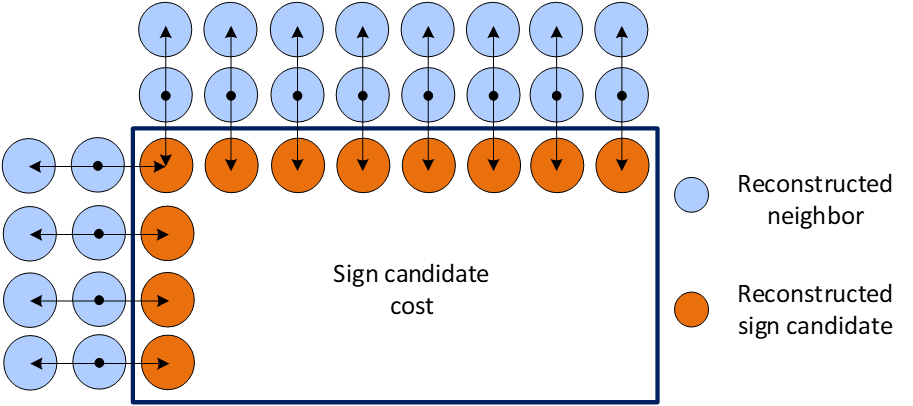


Figure 11. Discontinuity measure.

The cost function is defined as a sum of absolute second derivatives in the residual domain for the above row and left column as follows

where *R* is reconstructed neighbors, *P* is prediction of the current block, and *r* is the residual hypothesis. The term can be calculated only once per block and only residual hypothesis is subtracted.

## In-loop filtering

### Adaptive loop filter

#### ALF simplification removal

In JVET-U0100, ALF gradient subsampling and ALF virtual boundary processing are removed. Block size for classification is reduced from 4x4 to 2x2. Filter size for both luma and chroma, for which ALF coefficients are signalled, is increased to 9x9.

#### ALF with fixed filters

To filter a luma sample, it is proposed to use three different classifiers (C0, C1 and C2) and three different sets of filters (F0, F1 and F2). Sets F0 and F1 contain fixed filters, with coefficients trained for classifiers C0 and C1. Coefficients of filters in F2 are signalled. Which filter from a set Fi is used for a given sample is decided by a class assigned to this sample using classifier Ci

##### Filtering

At first, two 13x13 diamond shape fixed filters F0 and F1 are applied to derive two intermediate samples and . After that, F2 is applied to , , and neighboring samples to derive a filtered sample as

where is the clipped difference between a neighboring sample and current sample and is the clipped difference between and current sample. The filter coefficients are signalled.

##### Classification

Based on directionality and activity , a class is assigned to each 2x2 block:

where represents the total number of directionalities .

As in VVC, values of the horizontal, vertical, and two diagonal gradients are calculated for each sample using 1-D Laplacian. The sum of the sample gradients within a 4×4 window that covers the target 2×2 block is used for classifier C0 and the sum of sample gradients within a 12×12 window is used for classifiers C1 and C2. The sums of horizontal, vertical and two diagonal gradients are denoted, respectively, as , , and . The directionality is determined by comparing

with a set of thresholds. The directionality is derived as in VVC using thresholds 2 and 4.5. For and , horizontal/vertical edge strength and diagonal edge strength are calculated first. Thresholds are used. Edge strength is 0 if ; otherwise, is the maximum integer such that Edge strength is 0 if ; otherwise, is the maximum integer such that . When , i.e., horizontal/vertical edges are dominant, the is derived by using Table 4 (a); otherwise, diagonal edges are dominant, the is derived by using Table 4 (b).

Table 4. Mapping of and to

(a) (b)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 |  |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  | 0 | 28 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 2 | 0 | 0 | 0 | 0 | 0 |  | 1 | 29 | 30 | 0 | 0 | 0 | 0 | 0 |
| 2 | 3 | 4 | 5 | 0 | 0 | 0 | 0 |  | 2 | 31 | 32 | 33 | 0 | 0 | 0 | 0 |
| 3 | 6 | 7 | 8 | 9 | 0 | 0 | 0 |  | 3 | 34 | 35 | 36 | 37 | 0 | 0 | 0 |
| 4 | 10 | 11 | 12 | 13 | 14 | 0 | 0 |  | 4 | 38 | 39 | 40 | 41 | 42 | 0 | 0 |
| 5 | 15 | 16 | 17 | 18 | 19 | 20 | 0 |  | 5 | 43 | 44 | 45 | 46 | 47 | 48 | 0 |
| 6 | 21 | 22 | 23 | 24 | 25 | 26 | 27 |  | 6 | 49 | 50 | 51 | 52 | 53 | 54 | 55 |

To obtain , the sum of vertical and horizontal gradients is mapped to the range of 0 to , where is equal to 4 for and 15 for and .

In an ALF\_APS, up to 4 luma filter sets are signalled, each set may have up to 25 filters.

### Bilateral filter

The proposed filter is carried out in the sample adaptive offset (SAO) loop-filter stage, as shown in Figure 1. Both the proposed bilateral filter (BIF) and SAO are using samples from deblocking as input. Each filter creates an offset, and these are added to the input sample and then clipped, before proceeding to ALF.

A picture containing object

Description automatically generated

Figure 12. Both the proposed filter (BIF) and SAO uses samples from the deblocking stage as input. Both create an offset, and these are added to the input sample and clipped.

In detail, the output sample is obtained as

where is the input sample from deblocking, is the offset from the bilateral filter and is the offset from SAO.

The proposed implementation provides the possibility for the encoder to enable or disable filtering at the CTU and slice level. The encoder takes a decision by evaluating the RDO cost.

The following syntax elements are introduced:

7.3.7 General slice header syntax

|  |  |
| --- | --- |
| slice\_header( ) { | Descriptor |
| … |  |
| if( pps\_bilateral\_filter\_enabled\_flag ) { |  |
| **slice\_bilateral\_filter\_all\_ctb\_enabled\_flag** | u(1) |
| if( !slice\_bilateral\_filter\_all\_ctb\_enabled\_flag ) |  |
| **slice\_bilateral\_filter\_enabled\_flag** | u(1) |
| } |  |

7.3.11.2 Coding tree unit syntax

|  |  |
| --- | --- |
| coding\_tree\_unit( ) { | Descriptor |
| … |  |
| if( !slice\_bilateral\_filter\_all\_ctb\_enabled\_flag && slice\_bilateral\_filter\_enabled\_flag ) |  |
| **bilateral\_filter\_ctb\_flag**[ xCtb >> CtbLog2SizeY ][ yCtb >> CtbLog2SizeY ] | u(1) |

The semantic is as follows:

**slice\_bilateral\_filter\_all\_ctb\_enabled\_flag** equal to 1 specifies that the bilateral filter is enabled and is applied to all CTBs in the current slice. When slice\_bilateral\_filter\_all\_ctb\_enabled\_flag is not present, it is inferred to be equal to 0.

**slice\_bilateral\_filter\_enabled\_flag** equal to 1 specifies that the bilateral filter is enabled and may be applied to CTBs of the current slice. When slice\_bilateral\_filter\_enabled\_flag is not present, it is inferred to be equal to slice\_bilateral\_filter\_all\_ctb\_enabled\_flag.

**bilateral\_filter\_ctb\_flag**[ xCtb >> CtbLog2SizeY ][ yCtb >> CtbLog2SizeY ] equal to 1 specifies that the bilateral filter is applied to the luma coding tree block of the coding tree unit at luma location ( xCtb, yCtb ). bilateral\_filter\_ctb\_flag [ cIdx ][ xCtb >> CtbLog2SizeY ][ yCtb >> CtbLog2SizeY ] equal to 0 specifies that the bilateral filter is not applied to the luma coding tree block of the coding tree unit at luma location ( xCtb, yCtb ). When bilateral\_filter\_ctb\_flag is not present, it is inferred to be equal (slice\_bilateral\_filter\_all\_ctb\_enabled\_flag & slice\_bilateral\_filter\_enabled\_flag).

For CTUs that are filtered, the filtering process proceeds as follows.

At the picture border, where samples are unavailable, the bilateral filter uses extension (sample repetition) to fill in unavailable samples. For virtual boundaries, the behavior is the same as for SAO, i.e., no filtering occurs.

The samples surrounding the center sample are denoted according to Figure 13, where A, B, L and R stands for above, below, left and right and where NW, NE, SW, SE stands for north-west etc.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

*Figure 13: Naming convention for samples surrounding the center sample, .*

Each surrounding sample , etc will contribute with a corresponding modifier value , , etc. These are calculated the following way: Starting with the contribution from the sample to the right, , we calculate the difference

where denotes absolute value. For data that is not 10-bit, we instead use , where n = 8 for 8-bit data etc. The resulting value is now clipped so that it is smaller than 16:

The modifier value is now calculated as

where is an array of 16 values determined by the value of qpb = QP + bilateral\_filter\_qp\_offset:

If qpb ≤ 22, then LUT(x) = (0, 4, 4, 4, 3, 2, 1 ,2, 1, 1, 1, 1, 0, 1, 1, -1), otherwise

if qpb ≤ 27, then LUT(x) = (0, 8, 11, 11, 7, 5, 5, 4 , 5, 4, 4, 2, 2 , 2, 2, -2), otherwise

if qpb ≤ 32, then LUT(x) = (0, 9, 16, 19, 22, 22, 20, 15, 12, 12, 11, 9, 9, 7, 8, -3), otherwise

if qpb ≤ 37, then LUT(x) = (0, 12, 21, 28, 33, 36, 40, 40, 40, 36, 29, 22, 19,17, 15, -3), otherwise

LUT(x) = (0, 17, 23, 33, 37, 41, 44, 44, 45, 44, 42, 27, 22, 17, 15, -3).

The modifier values for , and are calculated from , and in the same way. For diagonal samples, the calculation also follows Equations 2 and 3, but uses a value shifted by 1. Using the diagonal sample as an example, we get

and the other diagonal samples are calculated likewise. The modifier values are summed together

Note that equals for the previous sample. Likewise, equals for the sample above, and similar symmetries can be found also for the diagonal modifier values. This means that in a hardware implementation, it is sufficient to calculate four values , , and , and the remaining four values can be obtained from previously calculated values.

The value is now multiplied either by or , which can be done using a single adder and logical AND gates in the following way:

where denotes logical and and is the most significant bit of the multiplier and is the least significant bit. The value to multiply with is obtained using the minimum block dimension as shown in Table 1:

|  |  |  |  |
| --- | --- | --- | --- |
| **Block type** |  |  |  |
| Intra | 3 | 2 | 1 |
| Inter | 2 | 2 | 1 |

*Table 1: Obtaining the c parameter from the minimum size D=min(width,height) of the block.*

Finally, the bilateral filter offset is calculated. For full strength filtering, we use

whereas for half-strength filtering, we instead use

A general formula for n-bit data is to use

where bilateral\_filter\_strength can be 0 or 1 and is signalled in the pps.

In accordance with the mandates of the BoG report to not be too restricted by complexity considerations, some minor changes may be tried such as increasing the number of tables or increasing the filter support.

## Entropy coding

### Extended precision

In JVET-U0100, the intermediate precision used in the arithmetic coding engine is increased, including three elements. First, the precisions for two probability states are both increased to 15 bits, in comparison to 10 bits and 14 bits in VVC. Second, the LPS range update process is modified as below,

if q >= 16384

q = 215 – 1 – q

RLPS = ((range \* (q>>6)) >>9) + 1,

where range is a 9-bit variable representing the width of the current interval, q is a 15-bit variable representing the probability state of the current context model, and RLPS is the updated range for LPS. This operation can also be realized by looking up a 512×256-entry in 9-bit look-up table. Third, at the encoder side, the 256-entry look-up table used for bits estimation in VTM is extended to 512 entries.

### Slice-type-based window size

As described JVET-U0100, since statistics are different with different slice types, it is beneficial to have a context’s probability state updated at a rate that is optimal under the given slice type. Therefore, for each context model, three window sizes are pre-defined for I-, B-, and P-slices, respectively, like the initialization parameters.