|  |
| --- |
| **INTERNATIONAL ORGANIZATION FOR STANDARDIZATION ORGANISATION INTERNATIONALE DE NORMALISATION ISO/IEC JTC 1/SC 29/WG 5 MPEG JOINT VIDEO CODING TEAM(S) WITH ITU-T SG 16** |
| **ISO/IEC JTC 1 / SC 29 / WG 5 N 90** |
| **Online, 6–15 October 2021** |
| |  |  | | --- | --- | | **Title:** | **Algorithm description of Enhanced Compression Model 3 (ECM 3)** | | **Source:** | **Convenor (Jens-Rainer Ohm)** | | **Type:** | **General** | | **Subtype:** | **N/A** | | **Status:** | **Approved** | | **Date:** | **2022-01-07** | | **Expected Action:** | **Info** | | **Action due date:** | **N/A** | | **No. of pages** | **28** (without this cover page) | | **Email of convenor:** | **ohm @ ient . rwth-aachen . de** | | **Committee URL:** | **https://sd.iso.org/documents/ui/#!/browse/iso/iso-iec-jtc-1/iso-iec-jtc-1-sc-29/iso-iec-jtc-1-sc-29-wg-5** | |

|  |  |
| --- | --- |
| **Joint Video Experts Team (JVET)**  **of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29**  23rd Meeting, by teleconference, 7–16 July 2021 | Document: JVET-X2025-v2 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **Algorithm description of Enhanced Compression Model 3 (ECM 3)** | | |
| *Status:* | Output document of JVET | | |
| *Purpose:* | Algorithm description of Enhanced Compression Model 2 | | |
| *Author(s) or Contact(s):* | Muhammed Coban Fabrice Le Léannec Mohammed Golam Sarwer Jacob Ström | Email: | [mcoban@qti.qualcomm.com](mailto:mcoban@qti.qualcomm.com) [fabricel@xiaomi.com](mailto:fabricel@xiaomi.com) m.sarwer@alibaba-inc.com [jacob.strom@ericsson.com](mailto:jacob.strom@ericsson.com) |
| *Source:* | Editors | | |

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

# Abstract

This document is Enhanced Compression Model 3 (ECM 3) software algorithm description. It includes the coding features and encoding methods implemented in ECM-3.0 software that are under coordinated exploration study by the Joint Video Exploration Team (JVET) of ITU-T VCEG and ISO/IEC MPEG as potential enhanced video coding technology beyond the capabilities of VVC.

ITU-T VCEG (Q6/16) and ISO/IEC MPEG (JTC 1/SC 29/WG 11) are studying the potential need for standardization of future video coding technology with a compression capability that significantly exceeds that of the current VVC standard. Such future standardization action could either take the form of additional extension(s) of VVC or an entirely new standard. The groups are working together on this exploration activity in a joint collaboration effort known as the Joint Video Exploration Team (JVET) to evaluate compression technology designs proposed by their experts in this area. The first Exploration Experiments (EE) were established in JVET meeting during 6–15 January 2021.

Contents

[Abstract 1](#_Toc91155992)

[1 Introduction 3](#_Toc91155993)

[2 Scope 3](#_Toc91155994)

[3 Algorithm description of the Enhanced Compression Model Software 3](#_Toc91155995)

[3.1 Intra prediction 3](#_Toc91155996)

[3.1.1 Multi-model LM (MMLM) 3](#_Toc91155997)

[3.1.2 Gradient PDPC 3](#_Toc91155998)

[3.1.3 Secondary MPM 3](#_Toc91155999)

[3.1.4 Reference sample interpolation and smoothing for intra-prediction 4](#_Toc91156000)

[3.1.5 Decoder side intra mode derivation (DIMD) 4](#_Toc91156001)

[3.1.6 Intra template matching 5](#_Toc91156002)

[3.1.7 Fusion for template-based intra mode derivation (TIMD) 6](#_Toc91156003)

[3.1.8 Combination of CIIP with TIMD and TM merge 6](#_Toc91156004)

[3.2 Inter prediction 7](#_Toc91156005)

[3.2.1 Local illumination compensation (LIC) 7](#_Toc91156006)

[3.2.2 Non-adjacent spatial candidate 7](#_Toc91156007)

[3.2.3 Template matching (TM) 8](#_Toc91156008)

[3.2.4 Multi-pass decoder-side motion vector refinement 9](#_Toc91156009)

[3.2.5 Adaptive decoder-side motion vector refinement 11](#_Toc91156010)

[3.2.6 OBMC 11](#_Toc91156011)

[3.2.7 Sample-based BDOF 12](#_Toc91156012)

[3.2.8 Interpolation 12](#_Toc91156013)

[3.2.9 Multi-hypothesis prediction (MHP) 13](#_Toc91156014)

[3.2.10 Adaptive reordering of merge candidates with template matching (ARMC-TM) 14](#_Toc91156015)

[3.2.11 Geometric partitioning mode (GPM) with merge motion vector differences (MMVD) 15](#_Toc91156016)

[3.2.12 Geometric partitioning mode (GPM) with template matching (TM) 15](#_Toc91156017)

[3.2.13 Bilateral matching AMVP-merge mode 16](#_Toc91156018)

[3.3 Transform and coefficient coding 17](#_Toc91156019)

[3.3.1 Dependent quantization with 8-states 17](#_Toc91156020)

[3.3.2 Maximum Transform Size and Zeroing-out of Transform Coefficients 18](#_Toc91156021)

[3.3.3 Enhanced MTS for intra coding 18](#_Toc91156022)

[3.3.4 Secondary Transformation: LFNST extension with large kernel 18](#_Toc91156023)

[3.3.5 Sign prediction 19](#_Toc91156024)

[3.4 Adaptive loop filter 20](#_Toc91156025)

[3.4.1 ALF simplification removal 20](#_Toc91156026)

[3.4.2 ALF with fixed filters 20](#_Toc91156027)

[3.4.3 Filtering 20](#_Toc91156028)

[3.4.4 Classification 20](#_Toc91156029)

[3.4.5 Alternative 2x2 ALF classifier 21](#_Toc91156030)

[3.4.6 CCALF with long tap filter 22](#_Toc91156031)

[3.5 Bilateral filter 22](#_Toc91156032)

[3.6 Bilateral inloop filter on chroma 25](#_Toc91156033)

[3.7 Cross-Component Sample Adaptive Offset (CCSAO) 26](#_Toc91156034)

[3.8 Entropy coding 27](#_Toc91156035)

[3.8.1 Extended precision 27](#_Toc91156036)

[3.8.2 Slice-type-based window size 28](#_Toc91156037)

[4 References 28](#_Toc91156038)

# Introduction

This document provides algorithm description and encoding methods of the coding tools implemented in Enhanced Compression Model 3 (ECM 3) software. Tools consist of extensions of the tools in the existing VVC design and methods that were proposed but not included in the VVC standard [1], and also new coding tools beyond that.

# Scope

The ECM-3.0 reference software is provided to demonstrate a reference implementation of encoding techniques and the decoding process for JVET Enhanced compression beyond VVC capability exploration work. The reference software can be accessed via

https://vcgit.hhi.fraunhofer.de/ecm/ECM.git.

This document provides an algorithm description as well as an encoder-side description of the ECM-3.0, which serves as a tutorial for the algorithm and encoding model implemented in the ECM-3.0 software. The purpose of this document is to share a common understanding of the coding features and the reference encoding methods supported in the ECM-3.0 software, in order to facilitate the assessment of the technical impact of new technologies during the exploration work.

# Algorithm description of the Enhanced Compression Model Software

## Intra prediction

### Multi-model LM (MMLM)

CCLM included in VVC is extended by adding three Multi-model LM (MMLM) modes (JVET-D0110). 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. In these cases, a gradient based PDPC, extended from horizontal/vertical mode, is applied (JVET-Q0391). 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.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.

Shape

Description automatically generated with low confidence

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

The 4-tap cubic interpolation is replaced with a 6-tap cubic interpolation filter, as described in JVET-D0119, for the derivation of predicted samples from the reference samples.

For reference sample filtering, a 6-tap gaussian filter is applied for larger blocks (W >= 32 and H >=32), existing VVC 4-tap gaussian interpolation filter is applied otherwise. The extended intra reference samples are derived using the the 4-tap interpolation filter 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. The division operations in weight derivation is performed utilizing the same lookup table (LUT) based integerization scheme used by the CCLM. For example the division operation in the orientation calculation

is computed by the following LUT-based scheme:

x = Floor( Log2( Gx ) )

normDiff = ( ( Gx<< 4 ) >> x ) & 15

x +=( 3 + ( normDiff  !=  0 ) ? 1 : 0 )

Orient = (Gy\* ( DivSigTable[ normDiff ] | 8 ) + ( 1<<( x-1 ) )) >> x

where

DivSigTable[16] = { 0, 7, 6, 5 ,5, 4, 4, 3, 3, 2, 2, 1, 1, 1, 1, 0 }.

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.

### Intra template matching

Intra template matching prediction (Intra TMP) is a special intra prediction mode that copies the best prediction block from the reconstructed part of the current frame, whose L-shaped template matches the current template. For a predefined search range, the encoder searches for the most similar template to the current template in a reconstructed part of the current frame and uses the corresponding block as a prediction block. The encoder then signals the usage of this mode, and the same prediction operation is performed at the decoder side.

The prediction signal is generated by matching the L-shaped causal neighbor of the current block with another block in a predefined search area in Figure 2 consisting of:

R1: current CTU

R2: top-left CTU

R3: above CTU

R4: left CTU

SAD is used as a cost function.

Within each region, the decoder searches for the template that has least SAD with respect to the current one and uses its corresponding block as a prediction block.

The dimensions of all regions (SearchRange\_w, SearchRange\_h) are set proportional to the block dimension (BlkW, BlkH) to have a fixed number of SAD comparisons per pixel. That is:

SearchRange\_w = a \* BlkW

SearchRange\_h = a \* BlkH

Where ‘’ is a constant that controls the gain/complexity trade-off. In practice, ‘’is equal to 5.

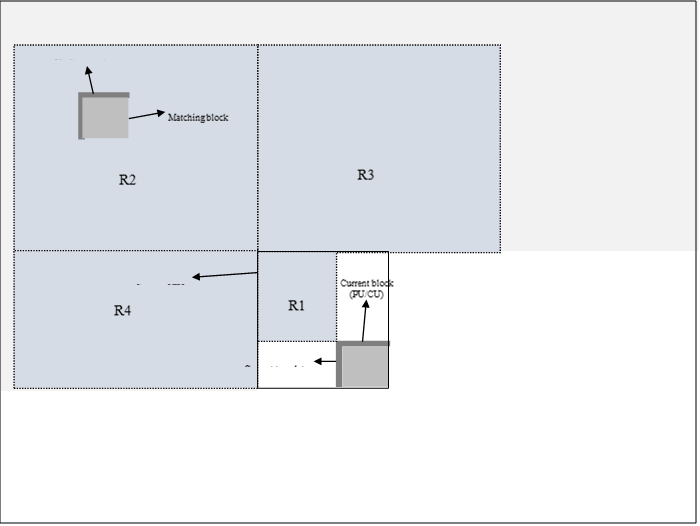


Figure 2. Intra template matching search area used.

The Intra template matching tool is enabled for CUs with size less than or equal to 64 in width and height. This maximum CU size for Intra template matching is configurable.

The Intra template matching prediction mode is signaled at CU level through a dedicated flag when DIMD is not used for current CU.

### Fusion for template-based intra mode derivation (TIMD)

For each intra prediction mode in MPMs, The SATD between the prediction and reconstruction samples of the template is calculated. First two intra prediction modes with the minimum SATD are selected as the TIMD modes. These two TIMD modes are fused with the weights after applying PDPC process, and such weighted intra prediction is used to code the current CU. Position dependent intra prediction combination (PDPC) is included in the derivation of the TIMD modes.

The costs of the two selected modes are compared with a threshold, in the test the cost factor of 2 is applied as follows:

costMode2 < 2\*costMode1.

If this condition is true, the fusion is applied, otherwise the only mode1 is used.

Weights of the modes are computed from their SATD costs as follows:

weight1 = costMode2/(costMode1+ costMode2)

weight2 = 1 - weight1

The division operations are conducted using the same lookup table (LUT) based integerization scheme used by the CCLM.

### Combination of CIIP with TIMD and TM merge

In CIIP mode, the prediction samples are generated by weighting an inter prediction signal predicted using CIIP-TM merge candidate and an intra prediction signal predicted using TIMD derived intra prediction mode. The method is only applied to coding blocks with an area less than or equal to 1024.

The TIMD derivation method is used to derive the intra prediction mode in CIIP. Specifically, the intra prediction mode with the smallest SATD values in the TIMD mode list is selected and mapped to one of the 67 regular intra prediction modes.

In addition, it is also proposed to modify the weights (wIntra, wInter) for the two tests if the derived intra prediction mode is an angular mode. For near-horizontal modes (2 <= angular mode index < 34), the current block is vertically divided as shown in   
Figure *3* (a); for near-vertical modes (34 <= angular mode index <= 66), the current block is horizontally divided as shown in  
Figure *3* (b).

The (wIntra, wInter) for different sub-blocks are shown in Table 1.

  
Figure 3 The division method for angular modes.

*Table 1. The modified weights used for angular modes.*

|  |  |
| --- | --- |
| **The sub-block index** | **(wIntra, wInter)** |
| 0 | (6, 2) |
| 1 | (5, 3) |
| 2 | (3, 5) |
| 3 | (2, 6) |

With CIIP-TM, a CIIP-TM merge candidate list is built for the CIIP-TM mode. The merge candidates are refined by template matching. The CIIP-TM merge candidates are also reordered by the ARMC method as regular merge candidates. The maximum number of CIIP-TM merge candidates is equal to two.

## 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 proposed in JVET-O0066 is used 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 are inserted after the TMVP in the regular merge candidate list. The pattern of spatial merge candidates is shown in Figure 443. 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 4. 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 554, a better MV is searched around the initial motion of the current CU within a [– 8, +8]-pel search range. The template matching method in JVET-J0021 is used with the following modifications: search step size is determined based on AMVR mode and TM can be cascaded with bilateral matching process in merge modes.

Diagram

Description automatically generated with medium confidence

Figure 5. Template matching performs on a search area around initial MV.

In AMVP mode, an MVP candidate is determined based on template matching error to select the one which reaches the minimum difference between the current block template and the reference block template, and then TM is performed 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 21. This search process ensures that the MVP candidate still keeps the same MV precision as indicated by the AMVR mode after TM process. In the search process, if the difference between the previous minimum cost and the current minimum cost in the interation is less than a threshold that is equal to the area of the block, the search process terminates.

Table 2. 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 21 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

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 665. 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. Additionally, if the difference between the previous minimum cost and the current minimum cost in the iteration is less than a threshold that is equal to the area of the block, the search process terminates.



Figure 6. 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

### Adaptive decoder-side motion vector refinement

Adaptive decoder side motion vector refinement method is an extension of multi-pass DMVR which consists of the two new merge modes to refine MV only in one direction, either L0 or L1, of the bi prediction for the merge candidates that meet the DMVR conditions. The multi-pass DMVR process is applied for the selected merge candidate to refine the motion vectors, however either MVD0 or MVD1 is set to zero in the 1st pass (i.e. PU level) DMVR.

The merge candidates for the new merge mode are derived from spatial neighboring coded blocks, TMVPs, non-adjacent blocks, HMVPs, pair-wise candidate, similar as in the regular merge mode. The difference is that only those meet DMVR conditions are added into the candidate list. The same merge candidate list is used by the two new merge modes. Merge index is coded as in regular merge mode.

### 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.

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 performed by applying the same blending to the top, left, bottom, and right subblock boundary pixels using neighboring subblocks’ 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.

When OBMC mode is used in CIIP mode with LMCS, inter blending is performed prior to LMCS mapping of inter samples. LMCS is applied to blended inter samples which are combined with LMCS applied intra samples in CIIP mode,

, where represents the samples predicted by the motion of current block in the original domain, represents the samples predicted in the mapped domain, represents the samples predicted by the motion of neighboring blocks in the original domain, and and are the weights.

### Sample-based BDOF

In the sample-based BDOF, 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.

### Interpolation

The 8-tap interpolation filter used in VVC is replaced with a 12-tap filter. The 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 32 gives the filter coefficients of all 16 phases. Figure 776 compares the frequency responses of the interpolation filters with the VVC interpolation filter, all at half-pel phase.

Table 3. Filter coefficients of the 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 7. Frequency responses of the interpolation filter and the VVC interpolation filter at half-pel phase

### Multi-hypothesis prediction (MHP)

In the multi-hypothesis inter prediction mode (JVET-M0425), one or more additional motion-compensated prediction signals are signaled, in addition to the conventional bi prediction signal. The resulting overall prediction signal is obtained by sample-wise weighted superposition. With the bi prediction signal and the first additional inter prediction signal/hypothesis , the resulting prediction signal is obtained as follows:

The weighting factor is specified by the new syntax element **add\_hyp\_weight\_idx**, according to the following mapping:

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

Analogously to above, more than one additional prediction signal can be used. The resulting overall prediction signal is accumulated iteratively with each additional prediction signal.

The resulting overall prediction signal is obtained as the last (i.e., the having the largest index ). Within this EE, up to two additional prediction signals can be used (i.e., is limited to 2).

The motion parameters of each additional prediction hypothesis can be signaled either explicitly by specifying the reference index, the motion vector predictor index, and the motion vector difference, or implicitly by specifying a merge index. A separate multi-hypothesis merge flag distinguishes between these two signalling modes.

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

Combination of MHP and BDOF is possible, however the BDOF is only applied to the bi-prediction signal part of the prediction signal (i.e., the ordinary first two hypotheses).

### Adaptive reordering of merge candidates with template matching (ARMC-TM)

The merge candidates are adaptively reordered with template matching (TM). The reordering method is applied to regular merge mode, template matching (TM) merge mode, and affine merge mode (excluding the SbTMVP candidate). For the TM merge mode, merge candidates are reordered before the refinement process.

After a merge candidate list is constructed, merge candidates are divided into several subgroups. The subgroup size is set to 5 for regular merge mode and TM merge mode. The subgroup size is set to 3 for affine merge mode. Merge candidates in each subgroup are reordered ascendingly according to cost values based on template matching. For simplification, merge candidates in the last but not the first subgroup are not reordered.

The template matching cost of a merge candidate is measured by the sum of absolute differences (SAD) between samples of a template of the current block and their corresponding reference samples. The template comprises a set of reconstructed samples neighboring to the current block. Reference samples of the template are located by the motion information of the merge candidate.

When a merge candidate utilizes bi-directional prediction, the reference samples of the template of the merge candidate are also generated by bi-prediction as shown in Figure 887.

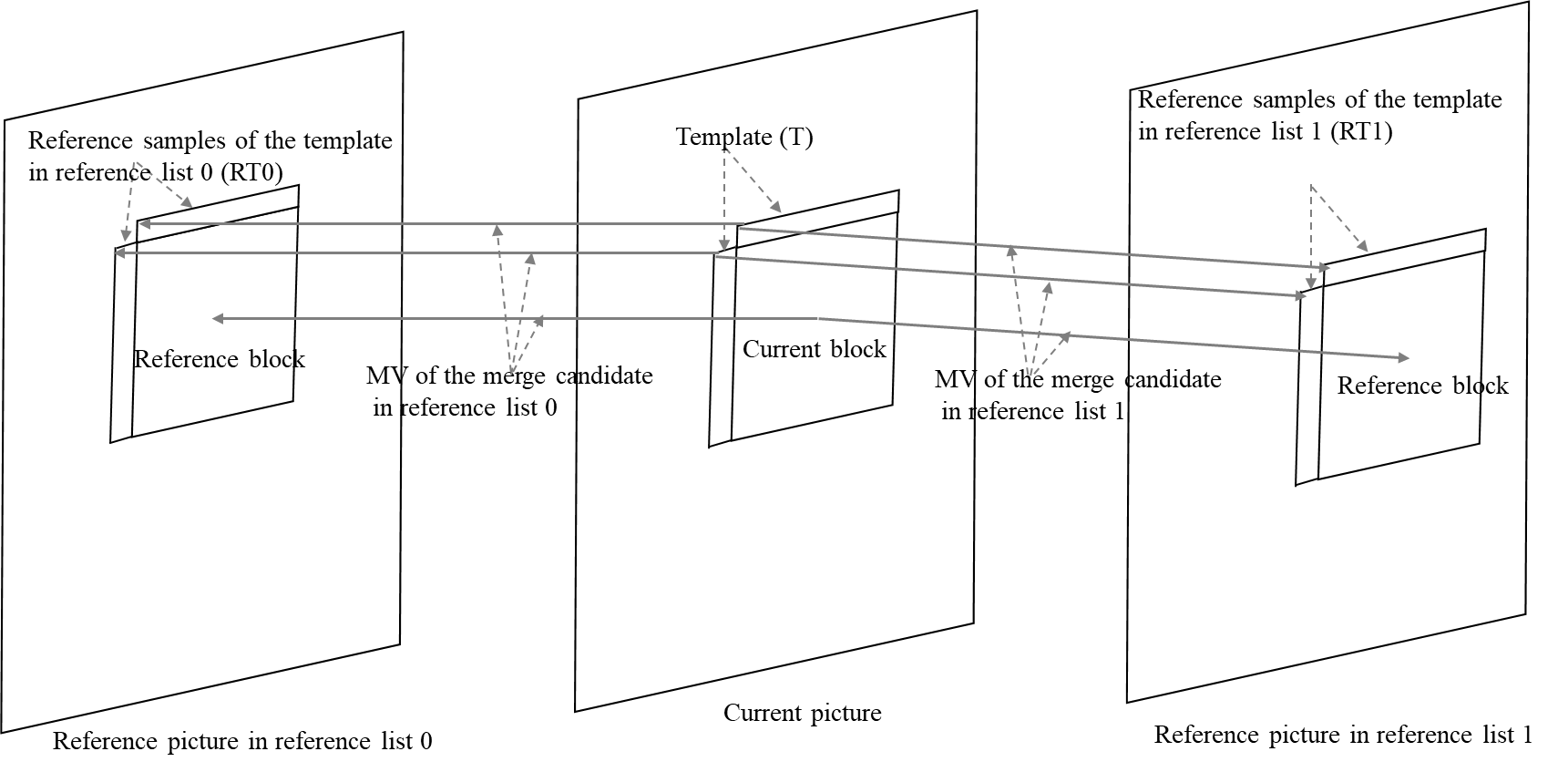
**

Figure 8. Template and reference samples of the template in reference pictures

For subblock-based merge candidates with subblock size equal to Wsub × Hsub, the above template comprises several sub-templates with the size of Wsub × 1, and the left template comprises several sub-templates with the size of 1 × Hsub. As shown in Figure 998, the motion information of the subblocks in the first row and the first column of current block is used to derive the reference samples of each sub-template.

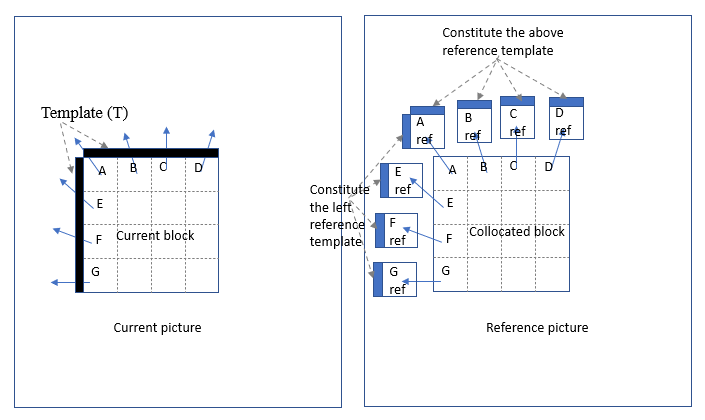


Figure 9. Template and reference samples of the template for block with sub-block motion using the motion information of the subblocks of the current block

### Geometric partitioning mode (GPM) with merge motion vector differences (MMVD)

GPM in VVC is extended by applying motion vector refinement on top of the existing GPM uni-directional MVs. A flag is first signalled for a GPM CU, to specify whether this mode is used. If the mode is used, each geometric partition of a GPM CU can further decide whether to signal MVD or not. If MVD is signalled for a geometric partition, after a GPM merge candidate is selected, the motion of the partition is further refined by the signalled MVDs information. All other procedures are kept the same as in GPM.

The MVD is signaled as a pair of distance and direction, similar as in MMVD. There are nine candidate distances (¼-pel, ½-pel, 1-pel, 2-pel, 3-pel, 4-pel, 6-pel, 8-pel, 16-pel), and eight candidate directions (four horizontal/vertical directions and four diagonal directions) involved in GPM with MMVD (GPM-MMVD). In addition, when pic\_fpel\_mmvd\_enabled\_flag is equal to 1, the MVD is left shifted by 2 as in MMVD.

### Geometric partitioning mode (GPM) with template matching (TM)

Template matching is applied to GPM. When GPM mode is enabled for a CU, a CU-level flag is signaled to indicate whether TM is applied to both geometric partitions. Motion information for each geometric partition is refined using TM. When TM is chosen, a template is constructed using left, above or left and above neighboring samples according to partition angle, as shown in Table 43. The motion is then refined by minimizing the difference between the current template and the template in the reference picture using the same search pattern of merge mode with half-pel interpolation filter disabled.

Table 4. Template for the 1st and 2nd geometric partitions, where A represents using above samples, L represents using left samples, and L+A represents using both left and above samples.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Partition angle | 0 | 2 | 3 | 4 | 5 | 8 | 11 | 12 | 13 | 14 |
| 1st partition | A | A | A | A | L+A | L+A | L+A | L+A | A | A |
| 2nd partition | L+A | L+A | L+A | L | L | L | L | L+A | L+A | L+A |
| Partition angle | 16 | 18 | 19 | 20 | 21 | 24 | 27 | 28 | 29 | 30 |
| 1st partition | A | A | A | A | L+A | L+A | L+A | L+A | A | A |
| 2nd partition | L+A | L+A | L+A | L | L | L | L | L+A | L+A | L+A |

A GPM candidate list is constructed as follows:

1. Interleaved List-0 MV candidates and List-1 MV candidates are derived directly from the regular merge candidate list, where List-0 MV candidates are higher priority than List-1 MV candidates. A pruning method with an adaptive threshold based on the current CU size is applied to remove redundant MV candidates.
2. Interleaved List-1 MV candidates and List-0 MV candidates are further derived directly from the regular merge candidate list, where List-1 MV candidates are higher priority than List-0 MV candidates. The same pruning method with the adaptive threshold is also applied to remove redundant MV candidates.
3. Zero MV candidates are padded until the GPM candidate list is full.

The GPM-MMVD and GPM-TM are exclusively enabled to one GPM CU. This is done by firstly signaling the GPM-MMVD syntax. When both two GPM-MMVD control flags are equal to false (i.e., the GPM-MMVD are disabled for two GPM partitions), the GPM-TM flag is signaled to indicate whether the template matching is applied to the two GPM partitions. Otherwise (at least one GPM-MMVD flag is equal to true), the value of the GPM-TM flag is inferred to be false.

### Bilateral matching AMVP-merge mode

The bi-directional predictor is composed of an AMVP predictor in one direction and a merge predictor in the other direction. The mode can be enabled to a coding block when the selected merge predictor and the AMVP predictor satisfy DMVR condition, where there is at least one reference picture from the past and one reference picture from the future relatively to the current picture and the distances from two reference pictures to the current picture are the same, the bilateral matching MV refinement is applied for the merge MV candidate and AMVP MVP as a starting point. Otherwise, if template matching functionality is enabled, template matching MV refinement is applied to the merge predictor or the AMVP predictor which has a higher template matching cost.

AMVP part of the mode is signaled as a regular uni-directional AMVP, i.e. reference index and MVD are signaled, and it has a derived MVP index if template matching is used or MVP index is signaled when template matching is disabled.

For AMVP direction LX, X can be 0 or 1, the merge part in the other direction (1 – LX) is implicitly derived by minimizing the bilateral matching cost between the AMVP predictor and a merge predictor, i.e. for a pair of the AMVP and a merge motion vectors. For every merge candidate in the merge candidate list which has that other direction (1 – LX) motion vector, the bilateral matching cost is calculated using the merge candidate MV and the AMVP MV. The merge candidate with the smallest cost is selected. The bilateral matching refinement is applied to the coding block with the selected merge candidate MV and the AMVP MV as a starting point.

The third pass of multi pass DMVR which is 8x8 sub-PU BDOF refinement of the multi-pass DMVR is enabled to AMVP-merge mode coded block.

The mode is indicated by a flag, if the mode is enabled AMVP direction LX is further indicated by a flag.

## Transform and coefficient coding

### Dependent quantization with 8-states

The coding efficiency of trellis-coded quantization in VVC increased by increasing the number of quantization states (at the cost of a higher encoder complexity). Dependent quantization with 8 quantization states in addition to the current variant of dependent quantization with 4 quantization state is supported (JVET-Q0243).

For supporting both variants of dependent quantization (4 and 8 states) in a unified framework, the decoding process for the VVC variant of dependent quantization is re-written.

The state transition table (sec. 7.4.12.11 in VVC) is modified from

QStateTransTable[ ][ ] = { { 0, 2 }, { 2, 0 }, { 1, 3 }, { 3, 1 } }

to

QStateTransTable[ ][ ] = { { 0, 1 }, { 2, 3 }, { 1, 0 }, { 3, 2 } }

There are three aspects that depend on the quantization state QState: (a) the mapping of transmitted transform coefficient levels to intermediate quantization indexes (part of the dequantization specified in the syntax); (b) the context selection for the sig\_coeff\_flag; (c) the derivation of the mapping parameter ZeroPos[ ] for transform coefficient levels coded in bypass mode. All three aspects are re-written in order to reflect the swapping of quantization states:

1. The mapping of transmitted transform coefficient levels to intermediate quantization indexes (see syntax structure residual\_coding() in VVC) is modified from

TransCoeffLevel[ x0 ][ y0 ][ cIdx ][ xC ][ yC ] =  
 ( 2 \* AbsLevel[ xC ][ yC ] − ( QState > 1 ? 1 : 0 ) ) \* ( 1 − 2 \* coeff\_sign\_flag[ n ] )

to

TransCoeffLevel[ x0 ][ y0 ][ cIdx ][ xC ][ yC ] =  
 ( 2 \* AbsLevel[ xC ][ yC ] − ( QState & 1 ) ) \* ( 1 − 2 \* coeff\_sign\_flag[ n ] )

1. The context selection of the sig\_coeff\_flag (see sec. 9.3.4.2.8 in VVC) depends on a parameter (context set id) that is derived based on the quantization state. In VVC, this parameter is given by

Max( 0, QState – 1 )

With the relabelling of the quantization states, this parameter can be derived according to

ctxSetId[ QState & 3 ] with ctxSetId[ ] = { 0, 1, 0, 2 }

It should be noted that for the 4-state version, the result of (QState & 3) is equal to QState. The masking is only required for the 8-state version of dependent quantization.

1. The derivation of the mapping parameter ZeroPos[ ] for transform coefficient levels coded in bypass mode is modified from

ZeroPos[ n ] = ( QState < 2 ? 1 : 2 )  <<  cRiceParam

to

ZeroPos[ n ] = ( 1 + ( QState & 1 ) )  <<  cRiceParam

### Maximum Transform Size and Zeroing-out of Transform Coefficients

Both CTU size and maximum transform size (i.e., all MTS transform kernels) are extended to 256, where the maximum intra coded block can have a size of 128x128. The maximum CTU size is set to 256 for UHD sequences and it is set to 128, otherwise. In the primary transformation process, there is no normative zeroing out operation applied on transform coefficients. However, if LFNST is applied, the primary transform coefficients outside the LFNST region are normatively zeroed-out.

### Enhanced MTS for intra coding

In the current VVC design [1], for MTS, only DST7 and DCT8 transform kernels are utilized which are used for intra and inter coding.

Additional primary transforms including DCT5, DST4, DST1, and identity transform (IDT) are employed. Also MTS set is made dependent on the TU size and intra mode information. 16 different TU sizes are considered, and for each TU size 5 different classes are considered depending on intra-mode information. For each class, 4 different transform pairs are considered, the same as that of VVC. Note, although a total of 80 different classes are considered, some of those different classes often share exactly same transform set. So there are 58 (less than 80) unique entries in the resultant LUT.

For angular modes, a joint symmetry over TU shape and intra prediction is considered. So, a mode i (i > 34) with TU shape AxB will be mapped to the same class corresponding to the mode j=(68 – i) with TU shape BxA. However, for each transform pair the order of the horizontal and vertical transform kernel is swapped. For example, for a 16x4 block with mode 18 (horizontal prediction) and a 4x16 block with mode 50 (vertical prediction) are mapped to the same class. However, the vertical and horizontal transform kernels are swapped. For the wide-angle modes the nearest conventional angular mode is used for the transform set determination. For example, mode 2 is used for all the modes between -2 and -14. Similarly, mode 66 is used for mode 67 to mode 80.

MTS index [0,3] is signalled with 2 bit fixed-length coding.

### Secondary Transformation: LFNST extension with large kernel

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]
* Three different kernels, LFNST4, LFNST8, and LFNST16, are defined to indicate LFNST kernel sets, which are applied to 4xN/Nx4 (N4), 8xN/Nx8 (N8), and MxN (M, N16), respectively.

The kernel dimensions are specified by:

(LFSNT4, LFNST8\*, LFNST16\*) = (16x16, 32x64, 32x96)

The forward LFNST is applied to top-left low frequency region, which is called Region-Of-Interest (ROI). When LFNST is applied, primary-transformed coefficients that exist in the region other than ROI are zeroed out, which is not changed from the VVC standard.

The ROI for LFNST16 is depicted in Figure 10109. It consists of six 4x4 sub-blocks, which are consecutive in scan order. Since the number of input samples is 96, transform matrix for forward LFNST16 can be Rx96. R is chosen to be 32 in this contribution, 32 coefficients (two 4x4 sub-blocks) are generated from forward LFNST16 accordingly, which are placed following coefficient scan order.



Figure 10 The ROI for LFNST16

The ROI for LFNST8 is shown in Figure 111110. The forward LFNST8 matrix can be Rx64 and R is chosen to be 32. The generated coefficients are located in the same manner as with LFNST16.

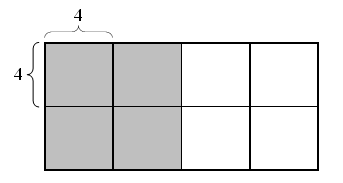


Figure 11. The ROI for LFNST8

The mapping from intra prediction modes to these sets is shown in Table 1,

Table 5. Mapping of intra prediction modes to LFNST set index



### Sign prediction

The basic idea of the coefficient sign prediction method (JVET-D0031 and JVET-J0021) is to calculate reconstructed residual for both negative and positive sign combinations for applicable transform coefficients 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 121211. It is measured for all hypotheses, and the one with the smallest cost is selected as a predictor for coefficient signs.

A picture containing text

Description automatically generated

Figure 12. 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.

## Adaptive loop filter

### ALF simplification removal

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, three different classifiers (C0, C1 and C2) and three different sets of filters (F0, F1 and F2) are used. 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 65 (a); otherwise, diagonal edges are dominant, the is derived by using Table 65 (b).

Table 6. 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.

### Alternative 2x2 ALF classifier

Classification in ALF is extended with an additional alternative classifier. For a signalled luma filter set, a flag is signalled to indicate whether the alternative classifier is applied. Geometrical transformation is not applied to the alternative band classifier. When the band-based classifier is applied, the sum of sample values of a 2x2 luma block is calculated at first. Then the class index is calculated as below,

class\_index = (sum \* 25) >> (sample bit depth + 2).

### CCALF with long tap filter

The CCALF process uses a linear filter to filter luma sample values and generate a residual correction for the chroma samples. A 25-tap large filter is used in CCALF process, which is illustrated in Figure 13. For a given slice, the encoder can collect the statistics of the slice, analyze them and can signal up to 16 filters through APS.

Background pattern

Description automatically generated

Figure . 25-tap long filter.

## Bilateral filter

The filter is carried out in the sample adaptive offset (SAO) loop-filter stage, as shown in Figure 141412. Both the bilateral filter (BIF) and SAO are using samples from deblocking as input. Each filter creates an offset per sample, and these are added to the input sample and then clipped, before proceeding to ALF.

Diagram

Description automatically generated

Figure 14. Both BIF and SAO use 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 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.

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. When crossing horizontal CTU borders, the bilateral filter can access the same samples as SAO is accessing. As an example, if the center sample (see Figure 151513) is located on the top line of a CTU, , and are read from the CTU above, just like SAO does, but is padded, so no extra line buffer is needed compared to JVET-P0073.

The samples surrounding the center sample are denoted according to Figure 151513, where A, B, L and R stands for above, below, left and right and where NW, NE, SW, SE stands for north-west etc. Likewise, AA stands for above-above, BB for below-below etc. This diamond shape is different from JVET-P0073 which used a square filter support, not using , , , or .

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

Figure 15. Naming convention for samples surrounding the center sample, I\_C.

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 = clip(0, 25, QP + bilateral\_filter\_qp\_offset-17):

{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, }, if qpb = 0  
{ 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, }, if qpb = 1  
{ 0, 2, 2, 2, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, }, if qpb = 2  
{ 0, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, -1, }, if qpb = 3  
{ 0, 3, 3, 3, 2, 2, 1, 2, 1, 1, 1, 1, 0, 1, 1, -1, }, if qpb = 4  
{ 0, 4, 4, 4, 3, 2, 1, 2, 1, 1, 1, 1, 0, 1, 1, -1, }, if qpb = 5  
{ 0, 5, 5, 5, 4, 3, 2, 2, 2, 2, 2, 1, 0, 1, 1, -1, }, if qpb = 6  
{ 0, 6, 7, 7, 5, 3, 3, 3, 3, 2, 2, 1, 1, 1, 1, -1, }, if qpb = 7  
{ 0, 6, 8, 8, 5, 4, 3, 3, 3, 3, 3, 2, 1, 2, 2, -2, }, if qpb = 8  
{ 0, 7, 10, 10, 6, 4, 4, 4, 4, 3, 3, 2, 2, 2, 2, -2, }, if qpb = 9  
{ 0, 8, 11, 11, 7, 5, 5, 4, 5, 4, 4, 2, 2, 2, 2, -2, }, if qpb = 10  
{ 0, 8, 12, 13, 10, 8, 8, 6, 6, 6, 5, 3, 3, 3, 3, -2, }, if qpb = 11  
{ 0, 8, 13, 14, 13, 12, 11, 8, 8, 7, 7, 5, 5, 4, 4, -2, }, if qpb = 12  
{ 0, 9, 14, 16, 16, 15, 14, 11, 9, 9, 8, 6, 6, 5, 6, -3, }, if qpb = 13  
{ 0, 9, 15, 17, 19, 19, 17, 13, 11, 10, 10, 8, 8, 6, 7, -3, }, if qpb = 14  
{ 0, 9, 16, 19, 22, 22, 20, 15, 12, 12, 11, 9, 9, 7, 8, -3, }, if qpb = 15  
{ 0, 10, 17, 21, 24, 25, 24, 20, 18, 17, 15, 12, 11, 9, 9, -3, }, if qpb = 16  
{ 0, 10, 18, 23, 26, 28, 28, 25, 23, 22, 18, 14, 13, 11, 11, -3, }, if qpb = 17  
{ 0, 11, 19, 24, 29, 30, 32, 30, 29, 26, 22, 17, 15, 13, 12, -3, }, if qpb = 18  
{ 0, 11, 20, 26, 31, 33, 36, 35, 34, 31, 25, 19, 17, 15, 14, -3, }, if qpb = 19  
{ 0, 12, 21, 28, 33, 36, 40, 40, 40, 36, 29, 22, 19, 17, 15, -3, }, if qpb = 20  
{ 0, 13, 21, 29, 34, 37, 41, 41, 41, 38, 32, 23, 20, 17, 15, -3, }, if qpb = 21  
{ 0, 14, 22, 30, 35, 38, 42, 42, 42, 39, 34, 24, 20, 17, 15, -3, }, if qpb = 22  
{ 0, 15, 22, 31, 35, 39, 42, 42, 43, 41, 37, 25, 21, 17, 15, -3, }, if qpb = 23  
{ 0, 16, 23, 32, 36, 40, 43, 43, 44, 42, 39, 26, 21, 17, 15, -3, }, if qpb = 24  
{ 0, 17, 23, 33, 37, 41, 44, 44, 45, 44, 42, 27, 22, 17, 15, -3, }, if qpb = 25

This is different from JVET-P0073 where 5 such tables were used, and the same table was reused for several qp-values.

As described in JVET-N0493 section 3.1.3, these values can be stored using six bits per entry resulting in 26\*16\*6/8=312 bytes or 300 bytes if excluding the first row which is all zeros.

The modifier values for , and are calculated from , and in the same way. For diagonal samples , , ,, and the samples two steps away , , and , 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 and two-steps-away 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- and two-steps-away modifier values. This means that in a hardware implementation, it is sufficient to calculate the six values , , , , and and the remaining six 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 76:

Table 7. Obtaining the c parameter from the minimum size D = min(width, height) of the block.

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

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.

## Bilateral inloop filter on chroma

Same as BIF-luma, proposed BIF-chroma is also performed in parallel with the SAO and CCSAO process as shown in Figure 16. BIF-chroma, CCSAO and SAO use the same chroma samples produced by the deblocking filter as input and generate three offsets per chroma sample in parallel. Then these three offsets are added to the input chroma sample to obtain a sum, which is then clipped to form the final output chroma sample value. The proposed BIF-chroma provides an on/off control mechanism on CTU level and slice level.

Diagram

Description automatically generated

Figure . Filtering stage of BIF-Chroma.

The filtering process of BIF-chroma is similar to that of BIF-luma. For a chroma sample, a 5×5 diamond shape filter is used for generating the filtering offset. The difference between the central sample and each surrounding sample is calculated first. The coefficient for each reference sample is extracted from a pre-defined look-up-table based on the calculated difference directly. The coefficients used for chroma components are retrained, different from those from BIF-luma. In the BIF-luma design, the block-level filtering strength parameter is determined based on luma TU size and CU mode. While in the BIF-chroma design, the parameter for chroma components is determined based the chroma TU size and mode when dual-tree partitioning is enabled for the current slice and based on the corresponding luma TU size and mode when dual-tree partitioning is disabled.

## Cross-Component Sample Adaptive Offset (CCSAO)

Cross-component Sample Adaptive Offset (CCSAO) is used to refine reconstructed chroma samples. Similarly to SAO, the CCSAO classifies the reconstructed samples into different categories, derives one offset for each category and adds the offset to the reconstructed samples in that category. However, different from SAO which only uses one single luma/chroma component of current sample as input, the CCSAO utilizes all three components to classify the current sample into different categories. To facilitate the parallel processing, the output samples from the de-blocking filter are used as the input of the CCSAO. Figure 17 shows the diagram of the decoding workflow when the CCSAO is applied.

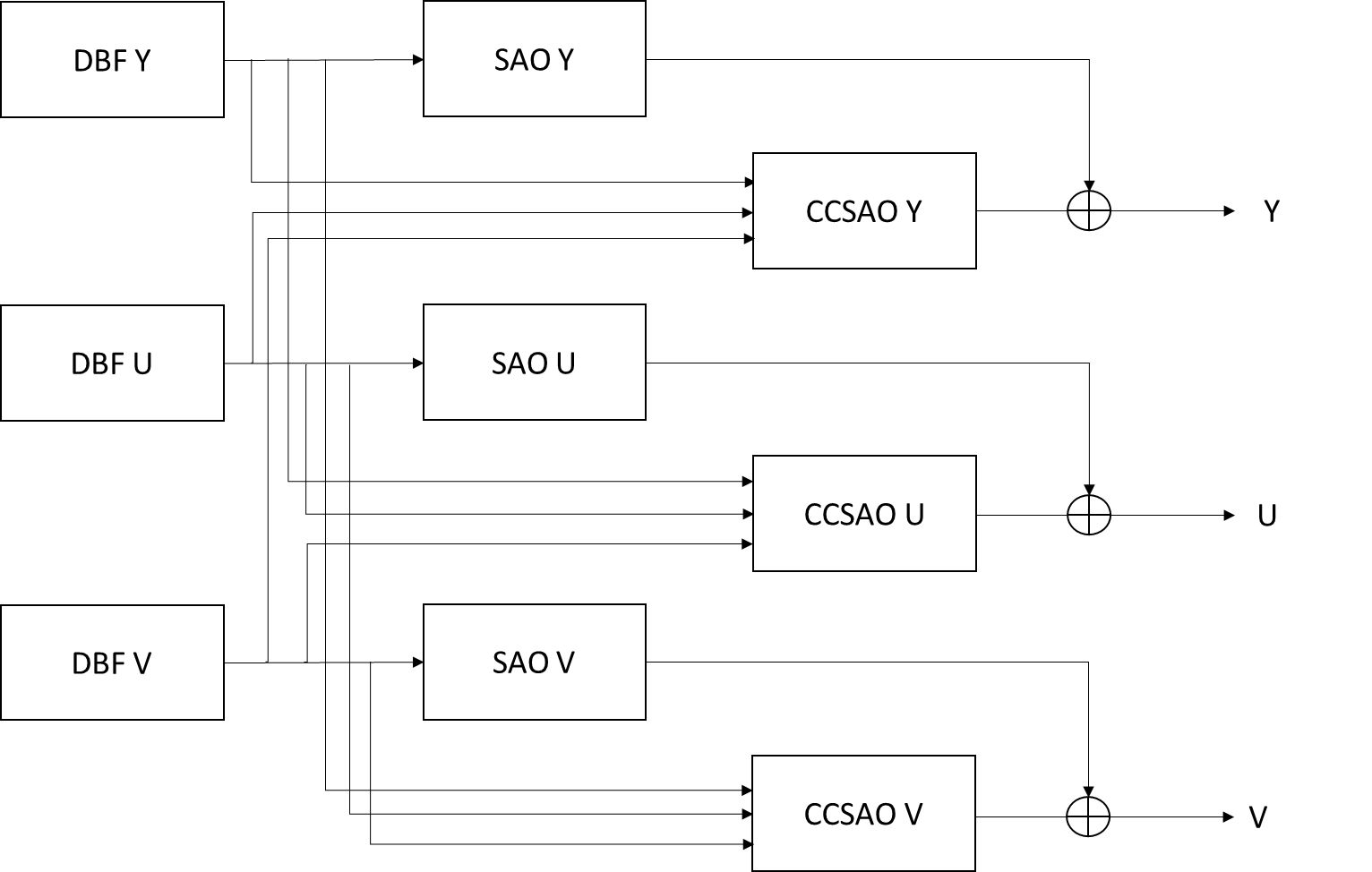


Figure : Modified SAO process when the proposed CCSAO is applied.

In CCSAO, only BO is used to enhance the quality of the reconstructed samples. For a given luma/chroma sample, three candidate samples are selected to classify the given sample into different categories: one collocated Y sample, one collocated U sample, and one collocated V sample. The sample values of these three selected samples are then classified into three different bands {}, and a joint index represents the category of the given sample. One offset is signaled and added to the reconstructed samples that fall into that category, which can be formulated as

where {*, ,* } are the three selected collocated samples used to classify current sample; {} are the numbers of equally divided bands applied to {*, ,* } full range respectively; is the internal coding bit-depth; and are the reconstructed samples before and after the CCSAO is applied; is the value of the CCSAO offset applied to *i*-th BO category. The collocated luma sample can be chosen from 9 candidate positions, while the collocated chroma sample positions are fixed, as depicted in Figure 18.

Similar to SAO, different classifiers can be applied to different local region to further enhance the whole picture quality. The parameters for each classifier (i.e., the position of , , , , and offsets) are signaled at picture level, and the classifier to be used is explicitly signaled and switched at CTB level. For each classifier, the maximum of {} is set to {16, 4, 4}, and offsets are constrained to be within the range [-15, 15]. At most 4 classifiers are used per frame.

Une image contenant texte, piscine à balles

Description générée automatiquement

Figure : Illustration of the candidate positions used for the CCSAO classifier.

SAO, Bilateral filter (BIF) and CCSAO offset are computed in parallel, added to the reconstructed chroma samples and jointly clipped, as shown in Figure 19.

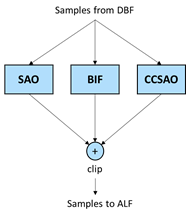


Figure : joint clipping after adding SAO/BIF/CCSAO offsets to the input sample.

## Entropy coding

### Extended precision

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

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.

The context initialization parameters and window sizes are retrained.

# References

1. B. Bross, J. Chen, S. Liu, and Y.-K. Wang "Versatile Video Coding (Draft 10)," document JVET-2001, 19th JVET meeting: by teleconference, 22 June – 1 July 2020.