**INTERNATIONAL ORGANIZATION FOR STANDARDIZATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

**ISO/IEC JTC1/SC29/WG11**

**CODING OF MOVING PICTURES AND AUDIO**

**ISO/IEC JTC1/SC29/WG11 N17733**

**July 2018, Ljubljana, SI**

|  |  |
| --- | --- |
| **Source** | **JVET** |
| **Status** | **Approved** |
| **Title** | **Test Model 2 of Versatile Video Coding (VTM 2)** |

|  |  |
| --- | --- |
| Joint Video Experts Team (JVET)  of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11  11th Meeting: Ljubljana, SI, 10–18 July 2018 | Document: JVET-K1002-v2 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **Algorithm description for Versatile Video Coding and Test Model 2 (VTM 2)** | | |
| *Status:* | Output document of JVET | | |
| *Purpose:* | Algorithm description for Versatile Video Coding and Test Model 2 | | |
| *Author(s) or Contact(s):* | Jianle Chen Yan Ye Seung Hwan Kim | Email: | [jianle.chen@huawei.com](mailto:jianle.chen@huawei.com) [Yan.Ye@InterDigital.com](mailto:Yan.Ye@InterDigital.com) [seunghwan3.kim@lge.com](mailto:seunghwan3.kim@lge.com) |
| *Source:* | Editors | | |

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

# Abstract

The JVET established the Versatile Video Coding (VVC) working draft 2 and the VVC Test Model 2 (VTM2) algorithm description and encoding method at its 11th meeting (10–18 July, 2018, Ljubljana, SI). This document serves as a source of general tutorial information on the VVC design and also provides an encoder-side description of VTM2.

VVC Test Model 2 (VTM2) algorithm description and encoding method

Ed. Notes:

* Incorporated JVET-K0230: Separate trees for intra slices (without multi-DMs) with an implicit split to 64x64;
* Incorporated JVET-K0556: Prohibit ternary split of something bigger than 64 in width or height (and not send the bit to indicate ternary type at that level).
* Incorporated JVET-K0351 (test c): Keep only the TT restriction (preventing binary split with same orientation in center partition of the ternary split)
* Incorporated JVET-K0554: Implicit splitting at picture boundaries and ensure MinQTSize at boundary splits
* Incorporated JVET-K0063: Position dependent intra prediction combination (PDPC)
* Incorporated JVET-K0190: CCLM only (test 4.1.8)
* Incorporated JVET-K0122: DC prediction bug fix
* Incorporated JVET-K0529: 67 modes with 3MPM and FLC for non-MPM
* Incorporated JVET-K0500: Wide-angle intra prediction for non-square block
* Incorporated MTS (AMT) modification: Multiple transform selection (MTS)
* Incorporated sub-block TMVP
* Incorporated adaptive motion vector resolution
* Incorporated 8x8 and 1/16 pel motion field storage
* Incorporated affine motion

# Introduction

At the 10th JVET meeting (April 10–20, 2018, San Diego, US), JVET defined the first draft of Versatile Video Coding (VVC) and the VVC Test Model 1 (VTM1) encoding method. It was decided to include a quadtree with nested multi-type tree using binary and ternary splits coding block structure as the initial new coding feature of VVC. Draft reference software to implement the VTM1 encoding method (and the draft VVC decoding process) has also been developed. At the 11th meeting (10–18 July, 2018, Ljubljana, SI), the Versatile Video Coding (VVC) working draft 2 and the VVC Test Model 2 (VTM2) algorithm description and encoding method were established with the inclusion of a group of new coding features as well as some of HEVC coding elements.

# Scope

The normative decoding process for Versatile Video Coding is specified in the VVC draft 2 text specification document [1]. The VTM2 reference software is provided to demonstrate a reference implementation of non-normative encoding techniques and the normative decoding process for VVC. The reference software can be accessed via

https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware\_VTM.git

This document provides an algorithm description as well as an encoder-side description of the VVC Test Model 2, which serves as a tutorial for the algorithm and encoding model implemented in the VTM2.0 software. The purpose of this document is to share a common understanding of the coding features of VVC and the reference encoding methods supported in the VTM2.0 software, in order to facilitate the assessment of the technical impact of new technologies during the standardization process. Common test conditions and software reference configurations that should be used for experimental work for conventional standard-dynamic range rectangular video content are described in JVET-K1010 [1]. Common test conditions specific to video content with high dynamic range and wide colour gamut are described in JVET-K1011 [3]. Common test conditions specific to video content for 360° omnidirectional video applications are described in JVET-K1012 [4]. When encoding and decoding 360° omnidirectional video, an additional software package called the 360Lib needs to be used together with using the VTM2.0 software to process, encode/decode and compute the spherical quality metrics. The 360Lib software is available at:

https://jvet.hhi.fraunhofer.de/svn/svn\_360Lib/

Additionally, document JVET-K1004 [5] describes the algorithms used in 360Lib to process, code, and measure quality of 360° omnidirectional video.

# Algorithm description of Versatile Video Coding

## VVC coding architecture

As in most preceding standards, VVC has a block-based hybrid coding architecture, combining inter-picture and intra-picture prediction and transform coding with entropy coding. Figure 1 shows a general block diagram of the VVC2 encoder.

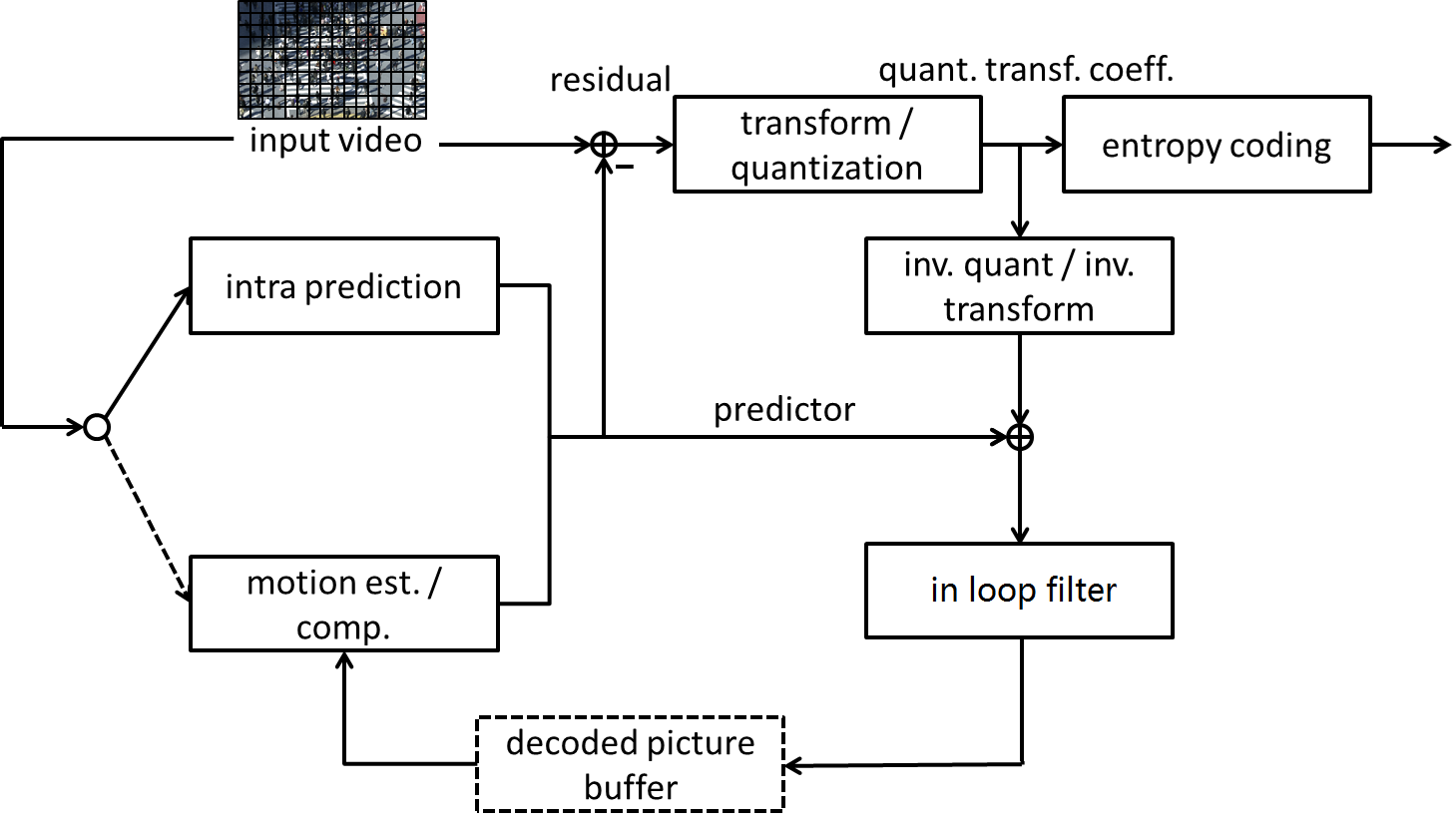


Figure 1 – General block diagram of VTM2 encoder

The picture partitioning structure, which is further described in section 3.2, divides the input video into blocks called coding tree units (CTUs). A CTU is split using a quadtree with nested multi-type tree structure into coding units (CUs), with a leaf coding unit (CU) defining a region sharing the same prediction mode (e.g. intra or inter). In this document, the term ‘unit’ defines a region of an image covering all colour components; the term ‘block’ is used to define a region covering a particular colour component (e.g. luma), and may differ in spatial location when considering the chroma sampling format such as 4:2:0.

The other features of VTM2, including intra prediction processes, inter picture prediction processes, transform and quantization processes, entropy coding processes and in-loop filter processes, are covered in sections 3.3 to 3.7, with an emphasis on how a feature is different from the corresponding HEVC process.As agreed in the 11th JVET meeting, the following features have been included in the VVC test model 2 on top of the bock tree structure.

* Intra prediction
  + 67 intra mode with wide angles mode extension
  + Position dependent intra prediction combination (PDPC)
  + Cross component linear model intra prediction
* Inter-picture prediction
  + Affine motion inter prediction
  + Advanced temporal motion vector prediction
  + Adaptive motion vector resolution
  + 8x8 block based motion compression
  + High precision (1/16 pel) motion vector storage and motion compensation
* Transform and quantization
  + Multiple primary transform selection with DCT2, DST7 and DCT8
  + Dependent quantization with max QP increased from 51 to 63
  + Modified transform coefficient coding with sign data hiding
* Adaptive Loop Filter
* Simple high-level syntax

## Partitioning

### Partitioning of the picture into CTUs

Pictures are divided into a sequence of coding tree units (CTUs). The CTU concept is same to that of the HEVC [5][6]. For a picture that has three sample arrays, a CTU consists of an N×N block of luma samples together with two corresponding blocks of chroma samples. Figure 1 shows the example of a picture divided into CTUs.

The maximum allowed size of the luma block in a CTU is specified to be 128×128 (although the maximum size of the luma transform blocks is 64×64).



Figure 2 – Example of a picture divided into CTUs

### Partitioning of the CTUs using a tree structure

In HEVC, a CTU is split into CUs by using a quaternary-tree structure denoted as coding tree to adapt to various local characteristics. The decision whether to code a picture area using inter-picture (temporal) or intra-picture (spatial) prediction is made at the leaf CU level. Each leaf CU can be further split into one, two or four PUs according to the PU splitting type. Inside one PU, the same prediction process is applied and the relevant information is transmitted to the decoder on a PU basis. After obtaining the residual block by applying the prediction process based on the PU splitting type, a leaf CU can be partitioned into transform units (TUs) according to another quaternary-tree structure similar to the coding tree for the CU. One of key feature of the HEVC structure is that it has the multiple partition conceptions including CU, PU, and TU.

In VVC, a quadtree with nested multi-type tree using binary and ternary splits segmentation structure replaces the concepts of multiple partition unit types, i.e. it removes the separation of the CU, PU and TU concepts except as needed for CUs that have a size too large for the maximum transform length, and supports more flexibility for CU partition shapes. In the coding tree structure, a CU can have either a square or rectangular shape. A coding tree unit (CTU) is first partitioned by a quaternary tree (a.k.a. quadtree) structure. Then the quaternary tree leaf nodes can be further partitioned by a multi-type tree structure. As shown in Figure 3, there are four splitting types in multi-type tree structure, vertical binary splitting (SPLIT\_BT\_VER), horizontal binary splitting (SPLIT\_BT\_HOR), vertical ternary splitting (SPLIT\_TT\_VER), and horizontal ternary splitting (SPLIT\_TT\_HOR). The multi-type tree leaf nodes are called coding units (CUs), and unless the CU is too large for the maximum transform length, this segmentation is used for prediction and transform processing without any further partitioning. This means that, in most cases, the CU, PU and TU have the same block size in the quadtree with nested multi-type tree coding block structure. The exception occurs when maximum supported transform length is smaller than the width or height of the colour component of the CU.



Figure 3 – Multi-type tree splitting modes

Figure 4 illustrates the signalling mechanism of the partition splitting information in quadtree with nested multi-type tree coding tree structure. A coding tree unit (CTU) is treated as the root of a quaternary tree and is first partitioned by a quaternary tree structure. Each quaternary tree leaf node (when sufficiently large to allow it) is then further partitioned by a multi-type tree structure. In the multi-type tree structure, a first flag (mtt\_split\_cu\_flag) is signalled to indicate whether the node is further partitioned; when a node is further partitioned, a second flag (mtt\_split\_cu\_vertical\_flag) is signalled to indicate the splitting direction, and then a third flag (mtt\_split\_cu\_binary\_flag) is signalled to indicate whether the split is a binary split or a ternary split. Based on the values of mtt\_split\_cu\_vertical\_flag and mtt\_split\_cu\_binary\_flag, the multi-type tree slitting mode (MttSplitMode) of a CU is derived as shown in Table 3‑1.

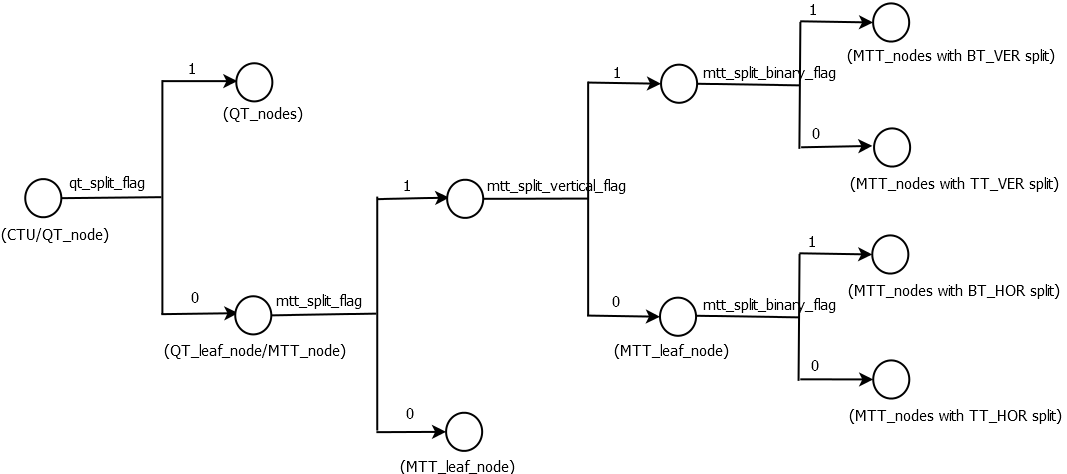


Figure 4 – Splitting flags signalling in quadtree with nested multi-type tree coding tree structure

**Table 3‑1 – MttSplitMode derviation based on multi-type tree syntax elements**

|  |  |  |
| --- | --- | --- |
| **MttSplitMode** | **mtt\_split\_cu\_vertical\_flag** | **mtt\_split\_cu\_binary\_flag** |
| SPLIT\_TT\_HOR | 0 | 0 |
| SPLIT\_BT\_HOR | 0 | 1 |
| SPLIT\_TT\_VER | 1 | 0 |
| SPLIT\_BT\_VER | 1 | 1 |

Figure 5 shows a CTU divided into multiple CUs with a quadtree and nested multi-type tree coding block structure, where the bold block edges represent quadtree partitioning and the remaining edges represent multi-type tree partitioning. The quadtree with nested multi-type tree partition provides a content-adaptive coding tree structure comprised of CUs. The size of the CU may be as large as the CTU or as small as 4×4 in units of luma samples. For the case of the 4:2:0 chroma format, the maximum chroma CB size is 64×64 and the minimum chroma CB size is 2×2.

In VVC, the maximum supported luma transform size is 64×64 and the maximum supported chroma transform size is 32×32. When the width or height of the CB is larger the maximum transform width or height, the CB is automatically split in the horizontal and/or vertical direction to meet the transform size restriction in that direction.

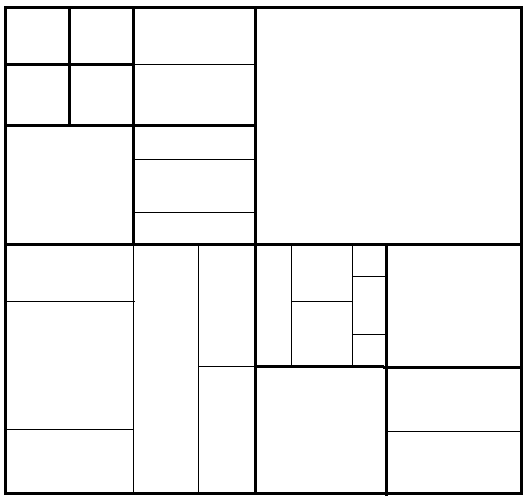


Figure 5– Example of quadtree with nested multi-type tree coding block structure

The following parameters are defined and specified by SPS syntax elements for the quadtree with nested multi-type tree coding tree scheme.

– CTU size: the root node size of a quaternary tree

– *MinQTSize*: the minimum allowed quaternary tree leaf node size

– *MaxBtSize*: the maximum allowed binary tree root node size

– *MaxTtSize*: the maximum allowed ternary tree root node size

– *MaxMttDepth*: the maximum allowed hierarchy depth of multi-type tree splitting from a quadtree leaf

– *MinBtSize*: the minimum allowed binary tree leaf node size

– *MinTtSize*: the minimum allowed ternary tree leaf node size

In one example of the quadtree with nested multi-type tree coding tree structure, the CTU size is set as 128×128 luma samples with two corresponding 64×64 blocks of 4:2:0 chroma samples, the *MinQTSize* is set as 16×16, the *MaxBtSize* is set as 128×128and *MaxTtSize* is set as 64×64, the *MinBtSize* and *MinTtSize* (for both width and height) is set as 4×4, and the *MaxMttDepth* is set as 4. The quaternary tree partitioning is applied to the CTU first to generate quaternary tree leaf nodes. The quaternary tree leaf nodes may have a size from 16×16 (i.e., the *MinQTSize*) to 128×128 (i.e., the CTU size). If the leaf QT node is 128×128, it will not be further split by the binary tree since the size exceeds the *MaxBtSize* and *MaxTtSize* (i.e., 64×64). Otherwise, the leaf qdtree node could be further partitioned by the multi-type tree. Therefore, the quaternary tree leaf node is also the root node for the multi-type tree and it has multi-type tree depth (mttDepth) as 0. When the multi-type tree depth reaches *MaxMttDepth* (i.e., 4), no further splitting is considered. When the multi-type tree node has width equal to *MinBtSize* and smaller or equal to 2 \* *MinTtSize*, no further horizontal splitting is considered. Similarly, when the multi-type tree node has height equal to *MinBtSize and* smaller or equal to 2 \* *MinTtSize*, no further vertical splitting is considered.

To allow 64×64 Luma block and 32×32 Chroma pipelining design in VVC hardware decoders, TT split is forbidden when either width or height of a luma coding block is larger than 64 , as shown in Figure 6. TT split is also forbidden when either width or height of a chroma coding block is larger than 32.

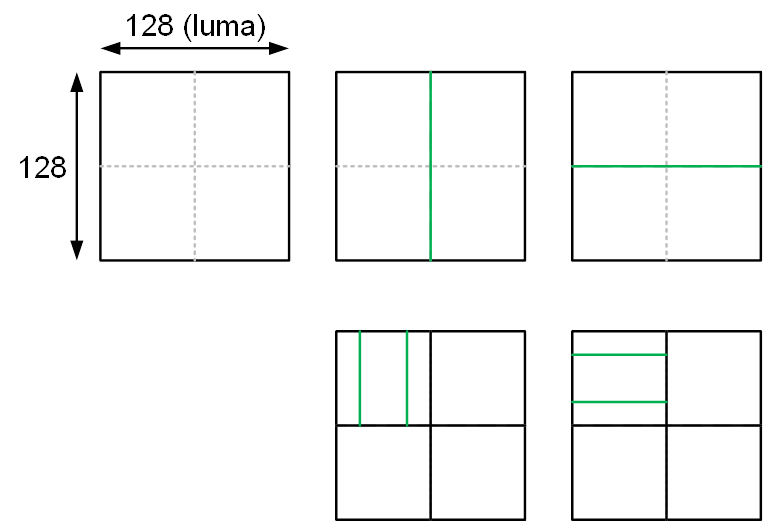


Figure 6– No TT split for 128×128 coding block

[Ed. (JC): Currently the maximum BT size and the maximum TT size are fixed in the reference software. The maximum BT size is 32x32 luma samples and corresponding chroma samples for I-slices and 128x128 luma samples and corresponding chroma samples for P/B-slices. The maximum TT size 32×32 luma samples and corresponding chroma samples for I-slices and 64x64 luma samples and corresponding chroma samples for P/B-slices]

In VTM2, the coding tree scheme supports the ability for the luma and chroma to have a separate block tree structure. Currently, for P and B slices, the luma and chroma CTBs in one CTU have to share the same coding tree structure. However, for I slices, the luma and chroma can have separate block tree structures. When separate block tree mode is applied, luma CTB is partitioned into CUs by one coding tree structure, and the chroma CTBs are partitioned into chroma CUs by another coding tree structure. This means that a CU in an I slice may consist of a coding block of the luma component or coding blocks of two chroma components, and a CU in a P or B slice always consists of coding blocks of all three colour components unless the video is monochrome.

### CU splits on picture boundaries

As done in HEVC, when a portion of a tree node block exceeds the bottom or right picture boundary, the tree node block is forced to be split until the all samples of every coded CU are located inside the picture boundaries. The following splitting rules are applied in the VTM2:

– If a portion of a tree node block exceeds both the bottom and the right picture boundaries,

* + If the block is a QT node and the size of the block is larger than the minimum QT size, the block is forced to be split with QT split mode.
  + Otherwise, the block is forced to be split with SPLIT\_BT\_HOR mode

– Otherwise if a portion of a tree node block exceeds the bottom picture boundaries,

* + If the block is a QT node, and the size of the block is larger than the minimum QT size, and the size of the block is larger than the maximum BT size, the block is forced to be split with QT split mode.
  + Otherwise, if the block is a QT node, and the size of the block is larger than the minimum QT size and the size of the block is smaller than or equal to the maximum BT size, the block is forced to be split with QT split mode or SPLIT\_BT\_HOR mode.
  + Otherwise (the block is a BTT node or the size of the block is smaller than or equal to the minimum QT size), the block is forced to be split with SPLIT\_BT\_HOR mode.

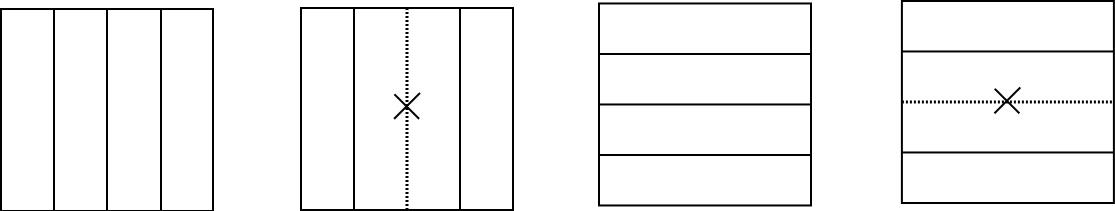
– Otherwise if a portion of a tree node block exceeds the right picture boundaries,

* + If the block is a QT node, and the size of the block is larger than the minimum QT size, and the size of the block is larger than the maximum BT size, the block is forced to be split with QT split mode.
  + Otherwise, if the block is a QT node, and the size of the block is larger than the minimum QT size and the size of the block is smaller than or equal to the maximum BT size, the block is forced to be split with QT split mode or SPLIT\_BT\_VER mode.
  + Otherwise (the block is a BTT node or the size of the block is smaller than or equal to the minimum QT size), the block is forced to be split with SPLIT\_BT\_VER mode.

### Restrictions on redundant CU splits

The quadtree with nested multi-type tree coding block structure provides a highly flexible block partitioning structure. Due to the types of splits supported the multi-type tree, different splitting patterns could potentially result in the same coding block structure. In VVC, some of these redundant splitting patterns are disallowed.

Figure 7 illustrates the redundant splitting patterns of binary tree splits and ternary tree splits. As shown in Figure 7, two levels of consecutive binary splits in one direction could have the same coding block structure as a ternary tree split followed by a binary tree split of the central partition. In this case, the binary tree split (in the given direction) for the central partition of a ternary tree split is prevented by the syntax. This restriction applies for CUs in all pictures.



**Figure 7–Redundant splitting patterns of binary tree split and ternary tree split cases**

When the splits are prohibited as described above, signalling of the corresponding syntax elements is modified to account for the prohibited cases. For example, when any case in Figure 7 is identified (i.e. the binary split is prohibited for a CU of a central partition), the syntax element mtt\_split\_cu\_binary\_flag which specifies whether the split is a binary split or a ternary split is not signalled and is instead inferred to be equal to 0 by the decoder.

## Intra prediction

### Intra mode coding with 67 intra prediction modes

To capture the arbitrary edge directions presented in natural video, the number of directional intra modes in VTM2 is extended from 33, as used in HEVC, to 65. The new directional modes not in HEVC are depicted as red dotted arrows in Figure 8, and the planar and DC modes remain the same. These denser directional intra prediction modes apply for all block sizes and for both luma and chroma intra predictions.

In VTM2, several conventional angular intra prediction modes are adaptively replaced with wide-angle intra prediction modes for the non-square blocks. Wide angle intra prediction is described in 3.3.1.2.

In HEVC, every intra-coded block has a square shape and the length of each of its side is a power of 2. Thus, no division operations are required to generate an intra-predictor using DC mode. In VTM2, blocks can have a rectangular shape that necessitates the use of a division operation per block in the general case. To avoid division operations for DC prediction, only the longer side is used to compute the average for non-square blocks.

#### Intra mode coding



Figure 8– 67 intra prediction modes

To keep the complexity of the most probable mode (MPM) list generation low, an intra mode coding method with 3 MPMs is used. The following three aspects are used to construct the MPM list:

* + - Neighbour intra modes
    - Derived intra modes
    - Default intra modes

For neighbor intra modes (A and B), two neighbouring blocks, located in left and above are considered. An initial MPM list is formed by performing pruning process for two neighboring intra modes. If two neighboring modes are different from each other, one of the default modes (i.e., PLANA (0), DC (1), ANGULAR50 (50)) is added to the MPM list after the pruning check with the existing two MPMs. Otherwise, if the two neighboring modes are the same, either the default modes or the derived modes are added to the MPM list after the pruning check.

Pruning is used to remove duplicated modes so that only unique modes can be included into the MPM list. For entropy coding of the 64 non-MPM modes, a 6-bit Fixed Length Code (FLC) is used.

#### Wide-angle intra prediction for non-square blocks

Conventional angular intra prediction directions are defined from 45 degrees to -135 degrees in clockwise direction. In VTM2, several conventional angular intra prediction modes are adaptively replaced with wide-angle intra prediction modes for non-square blocks. The replaced modes are signaled using the original mode indexes, which are remapped to the indexes of wide angular modes after parsing. The total number of intra prediction modes is unchanged, i.e., 67, and the intra mode coding method is unchanged.

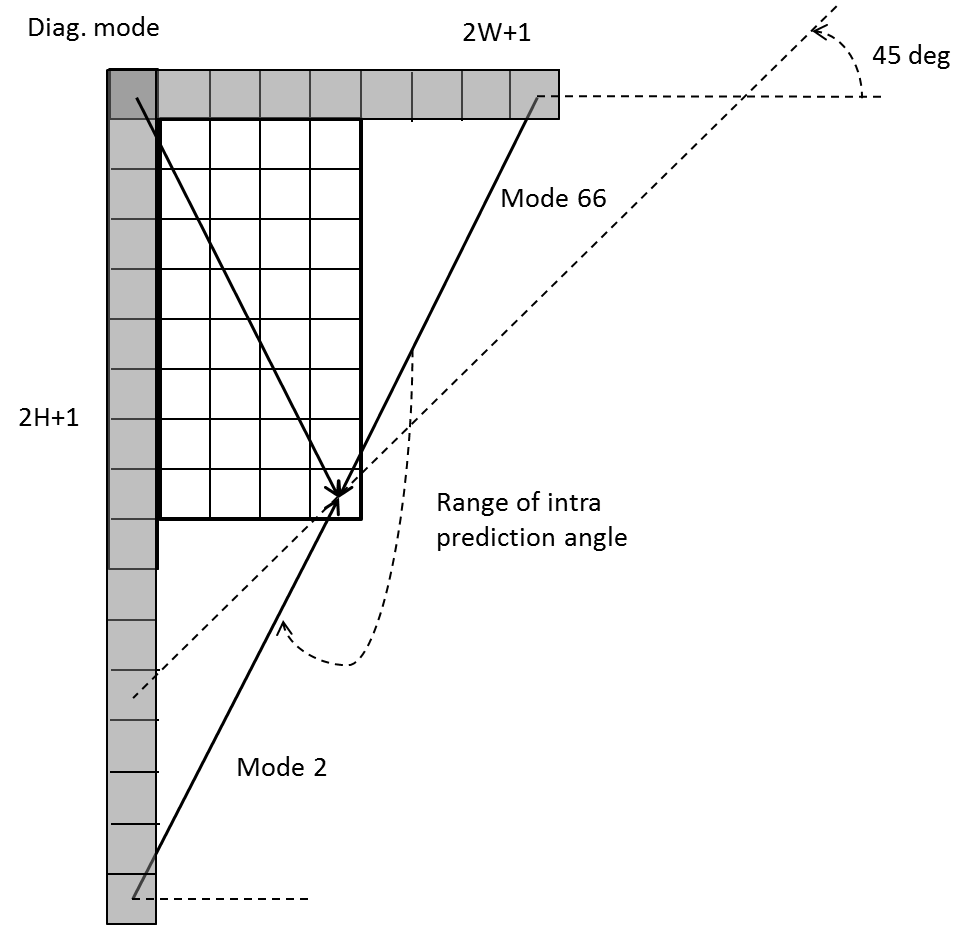
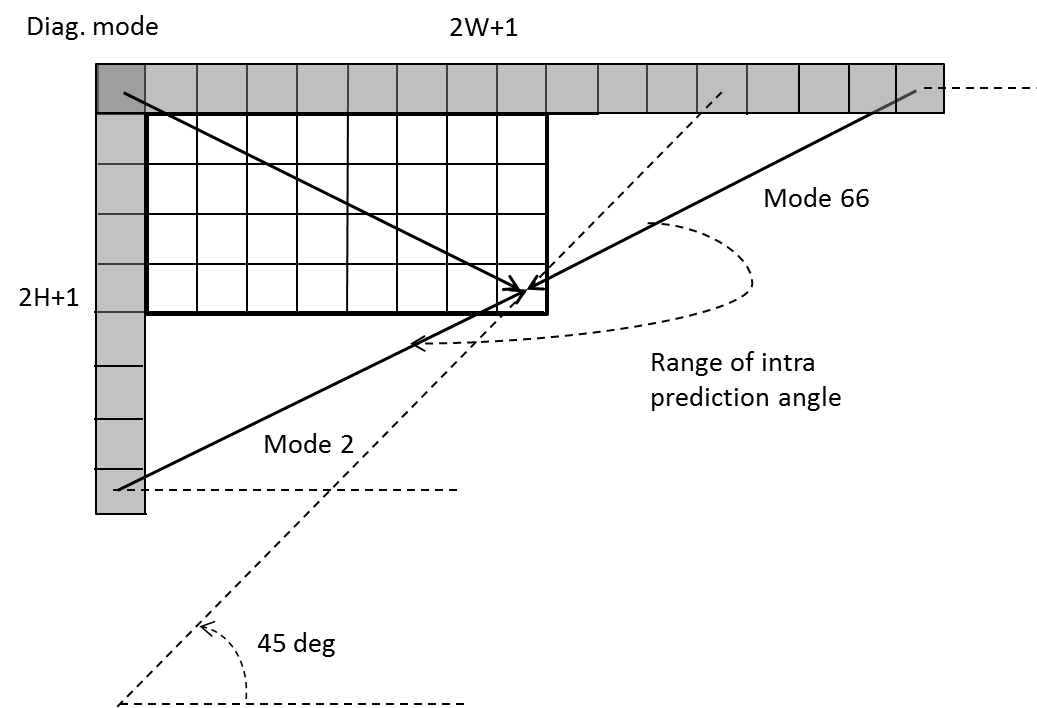


Figure 9 – Reference samples for wide-angular intra prediction

To support these prediction directions, the top reference with length 2W+1, and the left reference with length 2H+1, are defined as shown in Figure 9.

The number of replaced modex in wide-angular direction mode depends on the aspect ratio of a block. The replaced intra prediction modes are illustrated in Table 3‑2

Table 3‑2 - Intra prediction modes replaced by wide-angular modes

|  |  |
| --- | --- |
| Condition | Replaced intra prediction modes |
| W / H == 2 | Modes 2,3,4,5,6,7 |
| W / H > 2 | Modes 2,3,4,5,6,7,8,9,10,11 |
| W / H == 1 | None |
| H / W == 1/2 | Modes 61,62,63,64,65,66 |
| H / W < 1/2 | Mode 57,58,59,60,61,62,63,64,65,66 |



Figure 10 - Problem of discontinuity in case of directions beyond 45 degree

As shown in Figure 10, two vertically-adjacent predicted samples may use two non-adjacent reference samples in the case of wide-angle intra prediction. Hence, low-pass reference samples filter and side smoothing are applied to the wide-angle prediction to reduce the negative effect of the increased gap ∆pα.

#### Cross-component linear model prediction

To reduce the cross-component redundancy, a cross-component linear model (CCLM) prediction mode is used in the VTM2, for which the chroma samples are predicted based on the reconstructed luma samples of the same CU by using a linear model as follows:

(3-1)

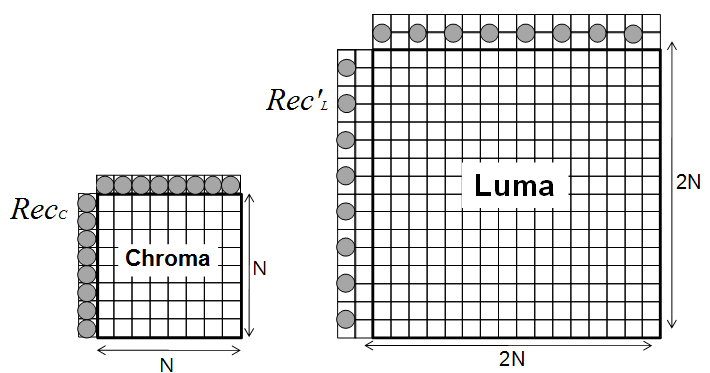
where represents the predicted chroma samples in a CU and represents the downsampled reconstructed luma samples of the same CU. Parameters and are derived by minimizing the regression error between the neighbouring reconstructed luma and chroma samples around the current block as follows:

(3-2)

(3-3)

where represents the down-sampled top and left neighbouring reconstructed luma samples, represents the top and left neighbouring reconstructed chroma samples, and value of is equal to twice of the minimum of width and height of the current chroma coding block. For a coding block with a square shape, the above two equations are applied directly. For a non-square coding block, the neighbouring samples of the longer boundary are first subsampled to have the same number of samples as for the shorter boundary. Figure 11 shows the location of the left and above samples and the sample of the current block involved in the CCLM mode.

This regression error minimization computation is performed as part of the decoding process, and is not just as an encoder search operation. As a result, no syntax is used to convey the α and β values to the decoder.



**Figure 11 - Locations of the samples used for the derivation of α and β**

For chroma intra mode coding, a total of 6 intra modes are allowed for chroma intra mode coding. Those modes include five traditional intra modes and one cross-component linear model mode (CCLM). Chroma mode signalling and derivation process are shown in Table 3-3.

Table 3‑3 – Derivation of chroma prediction mode from luma mode when cclm\_is enabled

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Chroma prediction mode | Corresponding luma intra prediction mode | | | | |
| 0 | 50 | 18 | 1 | X ( 0  <=  X  <=  66 ) |
| 0 | 66 | 0 | 0 | 0 | 0 |
| 1 | 50 | 66 | 50 | 50 | 50 |
| 2 | 18 | 18 | 66 | 18 | 18 |
| 3 | 1 | 1 | 1 | 66 | 1 |
| 4 | 77 | 77 | 77 | 77 | 77 |
| 5 | 0 | 50 | 18 | 1 | X |

### Position dependent intra prediction combination

In the VTM2, the results of intra prediction of planar mode are further modified by a position dependent intra prediction combination (PDPC) method. PDPC is an intra prediction method which invokes a combination of the un-filtered boundary reference samples and HEVC style intra prediction with filtered boundary reference samples. PDPC is applied to the following intra modes without signalling: planar, DC, horizontal, vertical, bottom-left angular mode and its *eight* adjacent angular modes, and top-right angular mode and its *eight* adjacent angular modes.

The prediction sample *pred*(*x*,*y*) is predicted using an intra prediction mode (DC, planar, angular) and a linear combination of reference samples according to the Equation 3-4 as follows:

*pred*(*x*,*y*)=(*wL*×*R*-1*,y* + *wT*×*Rx,*-1 – *wTL* ×*R*-1*,*-1+(64 – *wL* – *wT*+*wTL*)×*pred*(*x*,*y*) + 32 )>>6 (3-4)

where *Rx,*-1, *R*-1*,y* represent the reference samples located at the top and left of current sample (*x*, *y*), respectively, and *R*-1*,*-1 represents the reference sample located at the top-left corner of the current block.

If PDPC is applied to DC, planar, horizontal, and vertical intra modes, additional boundary filters are not needed, as required in the case of HEVC DC mode boundary filter or horizontal/vertical mode edge filters.

Figure 11 illustrates the definition of reference samples (*Rx,*-1, *R*-1*,y* and *R*-1*,*-1) for PDPC applied over various prediction modes. The prediction sample *pred* (*x’*, *y’*) is located at (*x’*, *y’*) within the prediction block. The coordinate *x* of the reference sample *Rx,*-1 is given by: *x* = *x’* + *y’* + 1, and the coordinate *y* of the reference sample *R*-1*,y* is similarly given by: *y* = *x’* + *y’* + 1.

|  |  |
| --- | --- |
| 1. Diagonal top-right mode | 1. Diagonal bottom-left mode |
| (c) Adjacent diagonal top-right mode | 1. Adjacent diagonal bottom-left mode |

**Figure 12 - Definition of samples used by PDPC applied to diagonal and adjacent angular intra modes.**

The PDPC weights are dependent on prediction modes and are shown in 4.

**Table 3‑4 - Example of PDPC weights according to prediction modes**

|  |  |  |  |
| --- | --- | --- | --- |
| Prediction modes | wT | wL | wTL |
| Diagonal top-right | 16 >> ( ( *y’*<<1 ) >> *shift*) | 16 >> ( ( *x’*<<1 ) >> *shift*) | 0 |
| Diagonal bottom-left | 16 >> ( ( *y’*<<1 ) >> *shift* ) | 16 >> ( ( *x’*<<1 ) >> *shift* ) | 0 |
| Adjacent diagonal top-right | 32 >> ( ( *y’*<<1 ) >> *shift* ) | 0 | 0 |
| Adjacent diagonal bottom-left | 0 | 32 >> ( ( *x’*<<1 ) >> *shift* ) | 0 |

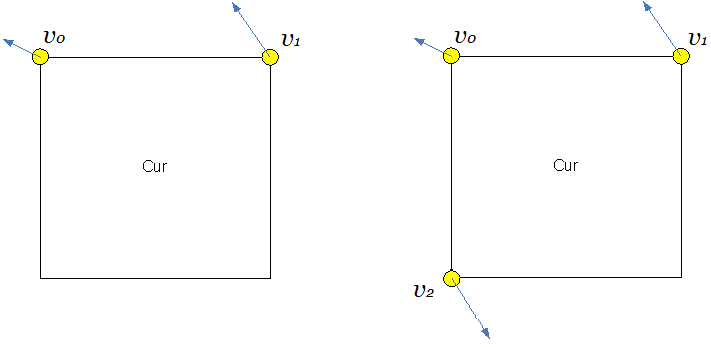
## Inter picture prediction

The VTM2.0 includes a number of new and refined inter prediction coding tools that are different from HEVC:

* Affine motion compensated prediction
* Subblock-based temporal motion vector prediction (SbTMVP)
* Adaptive motion vector resolution (AMVR)
* Motion field storage: 1/16th luma sample MV storage and 8x8 motion field compression

### Affine motion compensated prediction

In HEVC, only translation motion model is applied for motion compensation prediction (MCP). While in the real world, there are many kinds of motion, e.g. zoom in/out, rotation, perspective motions and the other irregular motions. In the VTM, a block-based affine transform motion compensation prediction is applied. As shown Figure 13, the affine motion field of the block is described by two control point (4-parameter) or three control point motion vectors (6-parameter).



**(a)4 parameter affine model (b) 6 parameter affine model**

**Figure 13 – control point based affine motion model**

For 4-parameter affine motion model, motion vector at sample location (*x, y*) in a block is derived as:

(3-5)

For 6-parameter affine motion model, motion vector at sample location (*x, y*) in a block is derived as:

(3-5)

Where (*v0x*, *v0y*) is motion vector of the top-left corner control point, (*v1x*, *v1y*) is motion vector of the top-right corner control point, and (*v2x*, *v2y*) is motion vector of the bottom-left corner control point.

In order to simplify the motion compensation prediction, block based affine transform prediction is applied. To derive motion vector of each 4×4 sub-block, the motion vector of the center sample of each sub-block, as shown in Figure 14, is calculated according to above equations, and rounded to 1/16 fraction accuracy. Then the motion compensation interpolation filters are applied to generate the prediction of each sub-block with derived motion vector.



Figure 14 – Affine MVF per sub-block

In the VTM, there are two affine motion modes: AF\_INTER mode and AF\_MERGE mode.

AF\_MERGE mode can be applied for CUs with both width and height larger than or equal to 8, When a CU is applied in AF\_MERGE mode, and the CPMVP of a neighboring block coded with affine mode are extrapolated to attain the CPMVs of the current block. The selection order for the candidate block is from left, above, above right, left bottom to above left as shown in Figure 15.a. If the neighbour left bottom block A is coded in affine mode as shown in Figure 15.b, the motion vectors , and of the top left corner, above right corner and left bottom corner of the CU which contains the block A are attained. When block A is coded with 4-parameter affine model, the two CPMVs of the current CU are calculated according to , and . In case that block A is coded with 6-parameter affine model, the three CPMVs of the current CU are calculated according to , and . In order to identify whether the current CU is coded with AF\_MERGE mode, an affine flag is signalled in the bitstream when there is at least one neighbour block is coded in affine mode.



1. (b)

Figure 15 – Candidates for AF\_MERGE

AF\_INTER mode can be applied for CUs with both width and height larger than or equal to 16. An affine flag in CU level is signalled in the bitstream to indicate whether AF\_INTER mode is used and then another flag is signaled to indicate whether 4-parameter affine or 6-parameter affine. In this mode, the difference of the CPMV of current CU and their predictors CPMVP is signalled in the bitstream. The CPMVP candidates list size is 2 and it is generated by using the following three types of CPVM candidate in order:

1. CPMVs extrapolated from the CPMVs of the neighbour blocks
2. CPMVs constructed using the translational MVs of the neighbour blocks.
3. CPMVs generated by duplicating each of the AMVP candidates

### Subblock-based temporal motion vector prediction (SbTMVP)

VTM-2.0 supports the subblock-based temporal motion vector prediction (SbTMVP) method. Similar to the temporal motion vector prediction (TMVP) in HEVC, SbTMVP uses the motion field in the collocated picture to improve motion vector prediction and merge mode for CUs in the current picture. The same collocated picture used by TMVP is used for SbTVMP. SbTMVP differs from TMVP in the following two main aspects:

1. TMVP predicts motion at CU level but SbTMVP predicts motion at sub-CU level;
2. Whereas TMVP fetches the temporal motion vectors from the collocated block in the collocated picture (the collocated block is the bottom-right or center block relative to the current CU), SbTMVP applies a motion shift before fetching the temporal motion information from the collocated picture, where the motion shift is obtained from the motion vector from one of the spatial neighboring blocks of the current CU.

The SbTVMP process is illustrated in Figure 13. SbTMVP predicts the motion vectors of the sub-CUs within the current CU in two steps. In the first step, the spatial neighbors in Figure 13 (a) are examined in the order of A1, B1, B0 and A0. As soon as and the first spatial neighboring block that has a motion vector that uses the collocated picture as its reference picture is identified, this motion vector is selected to be the motion shift to be applied. If no such motion is identified from the spatial neighbors, then the motion shift is set to (0, 0).

In the second step, the motion shift identified in Step 1 is applied (i.e. added to the current block’s coordinates) to obtain sub-CU-level motion information (motion vectors and reference indices) from the collocated picture as shown in Figure 13 (b). The example in Figure 13 (b) assumes the motion shift is set to block A1’s motion. Then, for each sub-CU, the motion information of its corresponding block (the smallest motion grid that covers the center sample) in the collocated picture is used to derive the motion information for the sub-CU. After the motion information of the collocated sub-CU is identified, it is converted to the motion vectors and reference indices of the current sub-CU in a similar way as the TMVP process of HEVC, where temporal motion scaling is applied to align the reference pictures of the temporal motion vectors to those of the current CU.



1. **Spatial neighboring blocks used by ATVMP**



1. **Deriving sub-CU motion field by applying a motion shift from spatial neighbor and scaling the motion information from the corresponding collocated sub-CUs**

**Figure 13 – The SbTMVP process in VVC**

The SbTVMP mode is enabled/disabled by a sequence parameter set (SPS) flag. If the SbTMVP mode is enabled, the SbTMVP predictor is added to the list of merge candidates, increasing the maximum number of merge candidates from 5 in HEVC to 6 in VVC.

The size of the sub-CU is adaptive and is determined based on a slice header SbTMVP size indication and on the size of the current CU. Denote the slice level size indication as SbTMVP\_size, the width/height of the SbTMVP sub-CU is set to the minimum value among CU\_width, CU\_height and SbTMVP\_size. In VTM-2.0, the value of SbTMVP\_size can be either 4 or 8.

The encoding logic of the additional SbTMVP merge candidate is the same as for the other merge candidates, that is, for each CU in P or B slice, an additional RD check is performed to decide whether to use the SbTMVP candidate. In terms of how the encoder decides SbTMVP\_size for the current picture, the following simple encoder logic is used to set its value to either 4 or 8. Specifically, block size statistics in the last coded picture in the same temporal layer is considered. Assume the last coded picture in the same temporal layer contains CUs that are coded by the SbTMVP mode. Moreover, assume the sizes (i.e. areas) of those CUs are , , …, . The average size of the SbTMVP CUs is calculated as. For the current picture, the value of SbTMVP\_size is determined according to the following:

SbTVMP\_size

Where *thres* is set to 27x27 for non-low-delay pictures, and set to 75x75 for low-delay pictures. For the first picture in each temporal layer, SbTMVP\_size is set to 4.

### Adaptive motion vector resolution (AMVR)

In HEVC, motion vector differences (MVDs) (between the motion vector and predicted motion vector of a CU) are signalled in units of quarter-luma-sample when use\_integer\_mv\_flag is equal to 0 in the slice header. In VVC, a CU-level adaptive motion vector resolution (AMVR) scheme is introduced. AMVR allows MVD of the CU to be coded in units of quarter-luma-sample, integer-luma-sample or four-luma-sample. The CU-level MVD resolution indication is conditionally signalled if the current CU has at least one non-zero MVD component. If all MVD components (that is, both horizontal and vertical MVDs for reference list L0 and reference list L1) are zero, quarter-luma-sample MVD resolution is inferred.

For a CU that has at least one non-zero MVD component, a first flag is signalled to indicate whether quarter-luma-sample MVD precision is used for the CU. If the first flag is 0, no further signaling is needed and quarter-luma-sample MVD precision is used for the current CU. Otherwise, a second flag is signalled to indicate whether integer-luma-sample or four-luma-sample MVD precision is used. When a CU uses integer-luma-sample or four-luma-sample MVD precision, the motion vector predictors for the CU are rounded to the corresponding precision.

The encoder determines the motion vector resolution for the current CU using RD check. To avoid always performing CU-level RD check three times for each MVD resolution, in VTM-2.0, the RD check of four-luma-sample MVD resolution is only invoked conditionally. The RD cost of quarter-luma-sample MVD precision is computed first. Then, the RD cost of integer-luma-sample MVD precision is compared to that of quarter-luma-sample MVD precision to decide whether it is necessary to further check the RD cost of four-luma-sample MVD precision. When the RD cost for quarter-luma-sample MVD precision is much smaller than that of the integer-luma-sample MVD precision, the RD check of four-luma-sample MVD precision is skipped.

### Motion field storage

In VTM-2.0, the highest precision of explicitly signalled motion vectors is quarter-luma-sample. In some inter prediction modes such as the affine mode, motion vectors are derived at 1/16th-luma-sample precision and motion compensated prediction is performed at 1/16th-sample-precision. In terms of internal motion field storage, all motion vectors are stored at 1/16th-luma-sample precision.

For temporal motion field storage used by TMVP and ATVMP, motion field compression is performed at 8x8 granularity in contrast to the 16x16 granularity in HEVC.

## Transform and quantization

### Large block-size transforms with high-frequency zeroing

In VTM2, large block-size transforms, up to 64×64 in size, are enabled, which is primarily useful for higher resolution video, e.g., 1080p and 4K sequences. High frequency transform coefficients are zeroed out for the transform blocks with size (width or height, or both width and height) equal to 64, so that only the lower-frequency coefficients are retained. For example, for an M×N transform block, with M as the block width and N as the block height, when M is equal to 64, only the left 32 columns of transform coefficients are kept. Similarly, when N is equal to 64, only the top 32 rows of transform coefficients are kept. When transform skip mode is used for a large block, the entire block is used without zeroing out any values.

### Multiple transform selection (MTS) for core transform

In addition to DCT-II which has been employed in HEVC, a Multiple Transform Selection (MTS) scheme is used for residual coding both inter and intra coded blocks. It uses multiple selected transforms from the DCT8/DST7. The newly introduced transform matrices are DST-VII and DCT-VIII. Table 3‑5 shows the basis functions of the selected DST/DCT.

Table 3‑5 - Transform basis functions of DCT-II/ VIII and DSTVII for N-point input

|  |  |
| --- | --- |
| Transform Type | Basis function *Ti*(*j*), *i*, *j* = 0, 1,…, *N*−1 |
| DCT-II | where, |
| DCT-VIII |  |
| DST-VII |  |

In order to keep the orthogonality of the transform matrix, the transform matrices are quantized more accurately than the transform matrices in HEVC. To keep the intermediate values of the transformed coefficients within the 16-bit range, after horizontal and after vertical transform, all the coefficients are to have 10-bit.

In order to control MTS scheme, separate enabling flags are specified at SPS level for intra and inter, respectively. When MTS is enabled at SPS, a CU level flag is signalled to indicate whether MTS is applied or not. Here, MTS is applied only for luma. The MTS CU level flag is signalled when the following conditions are satisfied.

* + - Both width and height smaller than or equal to 32
    - CBF flag is equal to one

If MTS CU flag is equal to zero, then DCT2 is applied in both directions. However, if MTS CU flag is equal to one, then two other flags are additionally signalled to indicate the transform type for the horizontal and vertical directions, respectively. For intra CU, those two flags (i.e., MTS\_Hor\_flag and MTS\_Ver\_flag) are signalled when the number of non-zero coefficients is greater than two. However, for inter CU, regardless of the number of nonzero coefficients, those flags are signalled. For example, for intra CU with only 1 or 2 nonzero coefficients, DST 7 is used both horizontally and vertically without signalling the additional two flags when MTS CU flag is equal to one. Transform and signalling mapping table as shown in Table 3‑6.

Table 3‑6 - Transform and signalling mapping table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| MTS\_CU\_flag | MTS\_Hor\_flag | MTS\_Ver\_flag | Intra | | Inter | |
|  |  |  | Horizontal | Vertical | Horizontal | Vertical |
| 0 |  |  | DCT2 | | | |
| 1 | 0 | 0 | DST7 | DST7 | DCT8 | DCT8 |
| 0 | 1 | DCT8 | DST7 | DST7 | DCT8 |
| 1 | 0 | DST7 | DCT8 | DCT8 | DST7 |
| 1 | 1 | DCT8 | DCT8 | DST7 | DST7 |

As in HEVC, the residual of a block can be coded with transform skip mode. To avoid the redundancy of syntax coding, the transform skip flag is not signalled when the CU level MTS\_CU\_flag is not equal to zero. For inter prediction residual, of the signalling of DST-VII and DCT-VIII is applied reversely as shown in the Table 3‑6.

## Entropy coding

To be determined.

## In-loop filter

To be determined.

# Description of VTM2 encoder and encoding methods

### Derivation process of coding tree structure

To be added.

# References

1. B. Bross, J. Chen, S. Liu, “Versatile Video Coding (Draft 2),” document JVET-K1001, 11th JVET meeting: Ljubljana, SI, 10–18 July 2018.
2. F. Bossen, J. Boyce, X. Li, and V. Seregin, K. Sühring, “JVET common test conditions and software reference configurations for SDR video,” document JVET-K1010, 11th JVET meeting: Ljubljana, SI, 10–18 July 2018.
3. A. Segall, E. François, and D. Rusanovskyy, “JVET common test conditions and evaluation procedures for HDR/WCG video,” document JVET-K1011, 11th JVET meeting: Ljubljana, SI, 10–18 July 2018.
4. P. Hanhart, J. Boyce, and K. Choi, “JVET common test conditions and evaluation procedures for 360° video,” document JVET-K1012, 11th JVET meeting: Ljubljana, SI, 10–18 July 2018.
5. Y. Ye, J. Boyce, “Algorithm descriptions of projection format conversion and video quality metrics in 360Lib Version 7,” document JVET-K1004, 11th JVET meeting: Ljubljana, SI, 10–18 July 2018.
6. High Efficiency Video Coding (HEVC), Rec. ITU-T H.265 and ISO/IEC 23008-2, Jan. 2013 (and later editions).
7. G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, “Overview of the High Efficiency Video Coding (HEVC) Standard”, *IEEE Trans. Circuits and Systems for Video Technology*, Vol. 22, No. 12, pp. 1649‒1668, Dec. 2012.