METHODS FOR CROSS COMPONENT DEPENDENCY REDUCTION

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] This application is based on International Patent Application No. PCT/CN2020/086111, filed on April 22, 2020, which claims the priority to and benefit of International Patent Application No. PCT/CN2019/083846, filed on April 23, 2019. All the aforementioned patent applications are hereby incorporated by reference in their entireties.

TECHNICAL FIELD

[0002] This patent document relates to video coding and decoding techniques, devices and systems.

BACKGROUND

[0003] In spite of the advances in video compression, digital video still accounts for the largest bandwidth use on the internet and other digital communication networks. As the number of connected user devices capable of receiving and displaying video increases, it is expected that the bandwidth demand for digital video usage will continue to grow.

SUMMARY

[0004] Devices, systems and methods related to digital video coding/decoding, and specifically, simplified linear model derivations for the cross-component linear model (CCLM) prediction mode in video coding/decoding are described. The described methods may be applied to both the existing video coding standards (e.g., High Efficiency Video Coding (HEVC)) and future video coding standards (e.g., Versatile Video Coding (VVC)) or codecs.

[0005] In one representative aspect, a method for visual media processing is disclosed. The method includes computing, during a conversion between a current video block of visual media data and a bitstream representation of the current video block, a cross-component linear model (CCLM) and/or a chroma residual scaling (CRS) factor for the current video block based, at least in part, on neighboring samples of a corresponding luma block which covers a top-left sample of a collocated luma block associated with the current video block, wherein one or more characteristics of the current video block are used for identifying the corresponding luma block.

[0006] In another representative aspect, a method for visual media processing is disclosed. The method includes using a rule to make a determination of selectively enabling or disabling a chroma residual scaling (CRS) on color components of a current video block of visual media data, wherein the rule is based on coding mode information of the current video block and/or coding mode information of one or

more neighbouring video blocks; and performing a conversion between the current video block and a bitstream representation, based on the determination.

[0007] In yet another representative aspect, a method for visual media processing is disclosed. The method includes using a single chroma residual scaling factor for at least one chroma block associated with video blocks in a slice or a tile group associated with a current video block of visual media data; and performing a conversion between the current video block and a bitstream representation of the current video block.

[0008] In another representative aspect, a method for visual media processing is disclosed. The method includes deriving a chroma residual scaling factor during a conversion between a current video block of visual media data and a bitstream representation of the current video block; storing the chroma residual scaling factor for use with other video blocks of the visual media data; and applying the chroma residual factor for the conversion of the current video block and the other video blocks into the bitstream representation.

[0009] In another representative aspect, a method for visual media processing is disclosed. The method includes during a conversion between a current video block of visual media data and a bitstream representation of the visual media data: computing a chroma residual factor of the current video block; storing, in a buffer, the chroma residual scaling factor for use with a second video block of the visual media data; and subsequent to the use, removing the chroma residual scaling factor from the buffer.

[0010] In yet another example aspect, a video encoder or decoder apparatus comprising a processor configured to implement an above described method is disclosed.

[0011] In another example aspect, a computer readable program medium is disclosed. The medium stores code that embodies processor executable instructions for implementing one of the disclosed methods.

[0012] In yet another representative aspect, the above-described method is embodied in the form of processor-executable code and stored in a computer-readable program medium.

BRIEF DESCRIPTION OF THE DRAWINGS

[0013] FIG. 1 shows an example of angular intra prediction modes in HEVC.

[0014] FIG. 2 shows an example of directional modes not in HEVC.

[0015] FIG. 3 shows an example in connection with the CCLM mode.

[0016] FIG. 4 shows an example of luma mapping with chroma scaling architecture.

[0017] FIG. 5 shows an example of luma block and chroma block in different color formats.

[0018] FIG. 6 shows an example of luma block and chroma block in same color formats.

[0019] FIG. 7 shows an example of collocated luma block covering multiple formats.

[0020] FIG. 8 shows an example of luma block within a larger luma block.

[0021] FIG. 9 shows an example of luma block within a larger luma block and within a bounding box.

[0022] FIG. 10 is a block diagram of an example of a hardware platform for implementing a visual media decoding or a visual media encoding technique described in the present document.

[0023] FIG. 11 shows a flowchart of an example method for linear model derivations for cross-component prediction in accordance with the disclosed technology.

[0024] FIG. 12 is a block diagram of an example video processing system in which disclosed techniques may be implemented.

[0025] FIG. 13 shows a flowchart of an example method for visual media processing.

[0026] FIG. 14 shows a flowchart of an example method for visual media processing.

[0027] FIG. 15 shows a flowchart of an example method for visual media processing.

[0028] FIG. 16 shows a flowchart of an example method for visual media processing.

[0029] FIG. 17 shows a flowchart of an example method for visual media processing.

DETAILED DESCRIPTION

2.1 A brief review on HEVC

2.1.1 Intra Prediction in HEVC/H.265

[0030] Intra prediction involves producing samples for a given TB (transform block) using samples previously reconstructed in the considered colour channel. The intra prediction mode is separately signalled for the luma and chroma channels, with the chroma channel intra prediction mode optionally dependent on the luma channel intra prediction mode via the 'DM_CHROMA' mode. Although the intra prediction mode is signalled at the PB (prediction block) level, the intra prediction process is applied at the TB level, in accordance with the residual quad-tree hierarchy for the CU, thereby allowing the coding of one TB to have an effect on the coding of the next TB within the CU, and therefore reducing the distance to the samples used as reference values.

[0031] HEVC includes 35 intra prediction modes – a DC mode, a planar mode and 33 directional, or 'angular' intra prediction modes. The 33 angular intra prediction modes are illustrated in FIG. 1.

[0032] For PBs associated with chroma colour channels, the intra prediction mode is specified as either planar, DC, horizontal, vertical, 'DM_CHROMA' mode or sometimes diagonal mode '34'.

[0033] Note for chroma formats 4:2:2 and 4:2:0, the chroma PB may overlap two or four (respectively) luma PBs; in this case the luma direction for DM_CHROMA is taken from the top left of these luma PBs.

[0034] The DM_CHROMA mode indicates that the intra prediction mode of the luma colour channel PB is applied to the chroma colour channel PBs. Since this is relatively common, the most-probable-mode coding scheme of the intra_chroma_pred_mode is biased in favor of this mode being selected.

2.2 Versatile Video Coding (VVC) algorithm description

2.2.1 VVC coding architecture

[0035] To explore the future video coding technologies beyond HEVC, Joint Video Exploration Team (JVET) was founded by VCEG and MPEG jointly in 2015. The JVET meeting is concurrently held once every quarter, and the new coding standard is targeting at 50% bitrate reduction as compared to HEVC. The new video coding standard was officially named as Versatile Video Coding (VVC) in the April 2018 JVET meeting, and the first version of VVC test model (VTM) was released at that time. As there are continuous effort contributing to VVC standardization, new coding techniques are being adopted to the VVC standard in every JVET meeting. The VVC working draft and test model VTM are then updated after every meeting. The VVC project is now aiming for technical completion (FDIS) at the July 2020 meeting.

[0036] 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. The picture partitioning structure 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.

2.2.2 Dual/separate tree partition in VVC

[0037] Luma component and chroma component can have separate partition trees for I slices. Separate tree partitioning is under 64x64 block level instead of CTU level. In VTM software, there is an SPS flag to control the dual-tree on and off.

2.2.3 Intra prediction in VVC

2.2.3.1 67 intra prediction modes

[0038] To capture the arbitrary edge directions presented in natural video, the number of directional intra modes in VTM4 is extended from 33, as used in HEVC, to 65. The new directional modes not in HEVC

are depicted as red dotted arrows in FIG. 2, 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.

2.2.3.2 Cross-component linear model prediction (CCLM)

[0039] To reduce the cross-component redundancy, a cross-component linear model (CCLM) prediction mode is used in the VTM4, for which the chroma samples are predicted based on the reconstructed luma samples of the same CU by using a linear model as follows:

$$\mathrm{pred_C}(i,j) = \alpha \cdot \mathrm{rec_L}'(i,j) + \beta$$

where $\mathrm{pred_C}(i,j)$ represents the predicted chroma samples in a CU and $\mathrm{rec_L}(i,j)$ represents the downsampled reconstructed luma samples of the same CU. Linear model parameter α and β are derived from the relation between luma values and chroma values from two samples, which are luma sample with minimum sample value and with maximum sample sample inside the set of downsampled neighboring luma samples, and their corresponding chroma samples. The linear model parameters α and β are obtained according to the following equations.

$$\alpha = \frac{Y_a - Y_b}{X_a - X_b}$$

$$\beta = Y_b - \alpha \cdot X_b$$

[0040] Where Y_a and X_a represent luma value and chroma value of the luma sample with maximum luma sample value. And X_b and Y_b represent luma value and chroma value of the luma sample with minimum luma sample, respectively. FIG. 3 shows an example of the location of the left and above samples and the sample of the current block involved in the CCLM mode.

[0041] The division operation to calculate parameter α is implemented with a look-up table. To reduce the memory required for storing the table, the *diff* value (difference between maximum and minimum values) and the parameter α are expressed by an exponential notation. For example, *diff* is approximated with a 4-bit significant part and an exponent. Consequently, the table for 1/diff is reduced into 16 elements for 16 values of the significand as follows:

DivTable [ ] = { 0, 7, 6, 5, 5, 4, 4, 3, 3, 2, 2, 1, 1, 1, 1, 0 }

[0042] This would have a benefit of both reducing the complexity of the calculation as well as the memory size required for storing the needed tables

[0043] Besides the above template and left template can be used to calculate the linear model coefficients together, they also can be used alternatively in the other 2 LM modes, called LM_A, and LM_L modes.

[0044] In LM_A mode, only the above template are used to calculate the linear model coefficients. To get more samples, the above template are extended to (W+H). In LM_L mode, only left template are

used to calculate the linear model coefficients. To get more samples, the left template are extended to (H+W).

[0045] For a non-square block, the above template are extended to W+W, the left template are extended to H+H.

[0046] To match the chroma sample locations for 4:2:0 video sequences, two types of downsampling filter are applied to luma samples to achieve 2 to 1 downsampling ratio in both horizontal and vertical directions. The selection of downsampling filter is specified by a SPS level flag. The two downsampling filters are as follows, which are corresponding to "type-0" and "type-2" content, respectively.

$$\mathrm{rec_L}'(i,j) = \begin{bmatrix} \mathrm{rec_L}(2i-1,2j-1) + 2\cdot \mathrm{rec_L}(2i-1,2j-1) + \mathrm{rec_L}(2i+1,2j-1) + \\ \mathrm{rec_L}(2i-1,2j) + 2\cdot \mathrm{rec_L}(2i,2j) + \mathrm{rec_L}(2i+1,2j) + 4 \end{bmatrix} \gg 3$$

$$\mathrm{rec_L}'(i,j) = \begin{bmatrix} \mathrm{rec_L}(2i,2j-1) + \mathrm{rec_L}(2i-1,2j) + 4\cdot \mathrm{rec_L}(2i,2j) \\ +\mathrm{rec_L}(2i+1,2j) + \mathrm{rec_L}(2i,2j+1) + 4 \end{bmatrix} \gg 3$$

[0047] Note that only one luma line (general line buffer in intra prediction) is used to make the downsampled luma samples when the upper reference line is at the CTU boundary.

[0048] This parameter 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.

[0049] For chroma intra mode coding, a total of 8 intra modes are allowed for chroma intra mode coding. Those modes include five traditional intra modes and three cross-component linear model modes (CCLM, LM_A, and LM_L). Chroma mode coding directly depends on the intra prediction mode of the corresponding luma block. Since separate block partitioning structure for luma and chroma components is enabled in I slices, one chroma block may correspond to multiple luma blocks. Therefore, for Chroma DM mode, the intra prediction mode of the corresponding luma block covering the center position of the current chroma block is directly inherited.

2.2.3.2.1 Corresponding modified working draft (JVET-N0271)

[0050] The following spec is based on the modified working draft of JVET-M1001 and the adoption in JVET-N0271. The modifications of the adopted JVET-N0220 are shown in bold and underlining.

Syntax table

Sequence parameter set RBSP syntax

sps_dmvr_enabled_flag	***u(1)***
sps_cclm_enabled_flag	***u(1)***
if(sps_cclm_enabled_flag && chroma_format_idc == 1)	
sps_cclm_colocated_chroma_flag	***u(1)***
sps_mts_enabled_flag	***u(1)***

Semantics

sps_cclm_enabled_flag equal to 0 specifies that the cross-component linear model intra prediction from luma component to chroma component is disabled. sps_cclm_enabled_flag equal to 1 specifies that the cross-component linear model intra prediction from luma component to chroma componenent is enabled.

Decoding process

In 8.4.4.2.8 Specification of INTRA_LT_CCLM, INTRA_L_CCLM and INTRA_T_CCLM intra prediction mode

Inputs to this process are:

- *the intra prediction mode predModeIntra,*
- *a sample location (xTbC, yTbC) of the top-left sample of the current transform block relative to the top-left sample of the current picture,*
- *a variable nTbW specifying the transform block width,*
- *a variable nTbH specifying the transform block height,*
- *chroma neighbouring samples p[x][y], with x = −1, y = 0..2 * nTbH − 1 and x = 0.. 2 * nTbW − 1, y = − 1.*

Output of this process are predicted samples predSamples[x][y], with x = 0..nTbW − 1, y = 0..nTbH − 1.

The current luma location (xTbY, yTbY) is derived as follows:

$$(xTbY, yTbY) = (xTbC << 1, yTbC << 1) \quad (8\text{-}156)$$

The variables availL, availT and availTL are derived as follows:

- *The availability of left neighbouring samples derivation process for a block as specified in clause* ***6.4.X [Ed. (BB): Neighbouring blocks availability checking process tbd]*** *is invoked with the current chroma location (xCurr, yCurr) set equal to (xTbC, yTbC) and the neighbouring chroma location (xTbC − 1, yTbC) as inputs, and the output is assigned to availL.*
- *The availability of top neighbouring samples derivation process for a block as specified in clause* ***6.4.X [Ed. (BB): Neighbouring blocks availability checking process tbd]*** *is invoked with the current chroma location (xCurr, yCurr) set equal to (xTbC, yTbC) and the neighbouring chroma location (xTbC, yTbC − 1) as inputs, and the output is assigned to availT.*
- *The availability of top-left neighbouring samples derivation process for a block as specified in clause* ***6.4.X [Ed. (BB): Neighbouring blocks availability checking process tbd]*** *is invoked with the current chroma location (xCurr, yCurr) set equal to (xTbC, yTbC) and the neighbouring chroma location (xTbC − 1, yTbC − 1) as inputs, and the output is assigned to availTL.*
- *The number of available top-right neighbouring chroma samples numTopRight is derived as follows:*
  - *The variable numTopRight is set equal to 0 and availTR is set equal to TRUE.*

- *When predModeIntra is equal to INTRA_T_CCLM, the following applies for x = nTbW..2 * nTbW − 1 until availTR is equal to FALSE or x is equal to 2 * nTbW − 1:*
  - *The availability derivation process for a block as specified in clause **6.4.X [Ed. (BB): Neighbouring blocks availability checking process tbd]** is invoked with the current chroma location (xCurr, yCurr) set equal to (xTbC , yTbC) and the neighbouring chroma location (xTbC + x, yTbC − 1) as inputs, and the output is assigned to availableTR*
  - *When availableTR is equal to TRUE, numTopRight is incremented by one.*
- *The number of available left-below neighbouring chroma samples numLeftBelow is derived as follows:*
  - *The variable numLeftBelow is set equal to 0 and availLB is set equal to TRUE.*
  - *When predModeIntra is equal to INTRA_L_CCLM, the following applies for y = nTbH..2 * nTbH − 1 until availLB is equal to FALSE or y is equal to 2 * nTbH − 1:*
    - *The availability derivation process for a block as specified in clause **6.4.X [Ed. (BB): Neighbouring blocks availability checking process tbd]** is invoked with the current chroma location (xCurr, yCurr) set equal to (xTbC , yTbC) and the neighbouring chroma location (xTbC − 1, yTbC + y) as inputs, and the output is assigned to availableLB*
    - *When availableLB is equal to TRUE, numLeftBelow is incremented by one.*

The number of available neighbouring chroma samples on the top and top-right numTopSamp and the number of available neighbouring chroma samples on the left and left-below nLeftSamp are derived as follows:

- *If predModeIntra is equal to INTRA_LT_CCLM, the following applies:*

$$numSampT = availT\ ?\ nTbW\ :\ 0 \quad (8\text{-}157)$$

$$numSampL = availL\ ?\ nTbH\ :\ 0 \quad (8\text{-}158)$$

- *Otherwise, the following applies:*

$$numSampT = (availT\ \&\&\ predModeIntra == INTRA_T_CCLM)\ ?\ (nTbW + \mathbf{Min}(numTopRight, \mathbf{nTbH}))\ :\ 0 \quad (8\text{-}159)$$

$$numSampL = (availL\ \&\&\ predModeIntra == INTRA_L_CCLM)\ ?\ (nTbH + \mathbf{Min}(numLeftBelow, \mathbf{nTbW}))\ :\ 0 \quad (8\text{-}160)$$

The variable bCTUboundary is derived as follows:

$$\mathbf{bCTUboundary = (yTbC\ \&\ (1 << (CtbLog2SizeY - 1) - 1)\ ==\ 0)\ ?\ TRUE\ :\ FALSE. \quad (8\text{-}161)}$$

The variable cntN and array pickPosN[] with N being replaced by L and T, are derived as follows:

- ***The variable numIs4N is set equal to ((availT && availL && predModeIntra == INTRA_LT_CCLM) ? 0 : 1).***
- ***The variable startPosN is set equal to numSampN >> (2 + numIs4N).***

– ***The variable pickStepN is set equal to Max(1, numSampN >> (1 + numIs4N)).***

– ***If availN is equal to TRUE and predModeIntra is equal to INTRA_LT_CCLM or INTRA_N_CCLM, cntN is set equal to Min(numSampN, (1 + numIs4N) << 1), and pickPosN[pos] is set equal to (startPosN + pos * pickStepN), with pos = 0..(cntN – 1).***

– ***Otherwise, cntN is set equal to 0.***

The prediction samples predSamples[x][y] with x = 0..nTbW − 1, y = 0..nTbH − 1 are derived as follows:

– *If both numSampL and numSampT are equal to 0, the following applies:*

$$predSamples[x][y] = 1 << (BitDepth_C - 1) \quad (8\text{-}162)$$

– *Otherwise, the following ordered steps apply:*

1. *The collocated luma samples pY[x][y] with x = 0..nTbW * 2 − 1, y= 0..nTbH * 2 − 1 are set equal to the reconstructed luma samples prior to the deblocking filter process at the locations (xTbY + x, yTbY + y).*
2. *The neighbouring luma samples samples pY[x][y] are derived as follows:*
   – *When numSampL is greater than 0, the neighbouring left luma samples pY[x][y] with x = −1..−3, y = 0..2 * numSampL − 1, are set equal to the reconstructed luma samples prior to the deblocking filter process at the locations (xTbY + x , yTbY +y).*
   – *When numSampT is greater than 0, the neighbouring top luma samples pY[x][y] with x = 0..2 * numSampT − 1, y = −1, −2, are set equal to the reconstructed luma samples prior to the deblocking filter process at the locations (xTbY+ x, yTbY + y).*
   – *When availTL is equal to TRUE, the neighbouring top-left luma samples pY[x][y] with x = −1, y = −1, −2, are set equal to the reconstructed luma samples prior to the deblocking filter process at the locations (xTbY+ x, yTbY + y).*
3. *The down-sampled collocated luma samples pDsY[x][y] with x = 0..nTbW − 1, y = 0..nTbH − 1 are derived as follows:*
   – *If sps_cclm_colocated_chroma_flag is equal to 1, the following applies:*
     – *pDsY[x][y] with x = 1..nTbW − 1, y = 1..nTbH − 1 is derived as follows:*

$$pDsY[x][y] = (pY[2 * x][2 * y - 1] + pY[2 * x - 1][2 * y] + 4 * pY[2 * x][2 * y] + pY[2 * x + 1][2 * y] + pY[2 * x][2 * y + 1] + 4) >> 3 \quad (8\text{-}163)$$

   – *If availL is equal to TRUE, pDsY[0][y] with y = 1..nTbH − 1 is derived as follows:*

$$pDsY[0][y] = (pY[0][2 * y - 1] + pY[-1][2 * y] + 4 * pY[0][2 * y] + pY[1][2 * y] + pY[0][2 * y + 1] + 4) >> 3 \quad (8\text{-}164)$$

– *Otherwise, pDsY[0][y] with y = 1..nTbH − 1 is derived as follows:*

$$pDsY[\,0\,][\,y\,] = (\,pY[\,0\,][\,2*y-1\,] + 2*pY[\,0\,][\,2*y\,] + pY[\,0\,][\,2*y+1\,] + 2\,) >> 2 \quad (8\text{-}165)$$

– *If availT is equal to TRUE, pDsY[x][0] with x = 1..nTbW − 1 is derived as follows:*

$$pDsY[\,x\,][\,0\,] = (\,pY[\,2*x\,][\,-1\,] + pY[\,2*x-1\,][\,0\,] + 4*pY[\,2*x\,][\,0\,] + pY[\,2*x+1\,][\,0\,] + pY[\,2*x\,][\,1\,] + 4\,) >> 3 \quad (8\text{-}166)$$

– *Otherwise, pDsY[x][0] with x = 1..nTbW − 1 is derived as follows:*

$$pDsY[\,x\,][\,0\,] = (\,pY[\,2*x-1\,][\,0\,] + 2*pY[\,2*x\,][\,0\,] + pY[\,2*x+1\,][\,0\,] + 2\,) >> 2 \quad (8\text{-}167)$$

– *If availL is equal to TRUE and availT is equal to TRUE, pDsY[0][0] is derived as follows:*

$$pDsY[\,0\,][\,0\,] = (\,pY[\,0\,][\,-1\,] + pY[\,-1\,][\,0\,] + 4*pY[\,0\,][\,0\,] + pY[\,1\,][\,0\,] + pY[\,0\,][\,1\,] + 4\,) >> 3 \quad (8\text{-}168)$$

– *Otherwise if availL is equal to TRUE and availT is equal to FALSE, pDsY[0][0] is derived as follows:*

$$pDsY[\,0\,][\,0\,] = (\,pY[\,-1\,][\,0\,] + 2*pY[\,0\,][\,0\,] + pY[\,1\,][\,0\,] + 2\,) >> 2 \quad (8\text{-}169)$$

– *Otherwise if availL is equal to FALSE and availT is equal to TRUE, pDsY[0][0] is derived as follows:*

$$pDsY[\,0\,][\,0\,] = (\,pY[\,0\,][\,-1\,] + 2*pY[\,0\,][\,0\,] + pY[\,0\,][\,1\,] + 2\,) >> 2 \quad (8\text{-}170)$$

– *Otherwise (availL is equal to FALSE and availT is equal to FALSE), pDsY[0][0] is derived as follows:*

$$pDsY[\,0\,][\,0\,] = pY[\,0\,][\,0\,] \quad (8\text{-}171)$$

– *Otherwise, the following applies:*

– *pDsY[x][y] with x = 1..nTbW − 1, y = 0..nTbH − 1 is derived as follows:*

$$pDsY[\,x\,][\,y\,] = (\,pY[\,2*x-1\,][\,2*y\,] + pY[\,2*x-1\,][\,2*y+1\,] + 2*pY[\,2*x\,][\,2*y\,] + 2*pY[\,2*x\,][\,2*y+1\,] + pY[\,2*x+1\,][\,2*y\,] + pY[\,2*x+1\,][\,2*y+1\,] + 4\,) >> 3 \quad (8\text{-}172)$$

– *If availL is equal to TRUE, pDsY[0][y] with y = 0..nTbH − 1 is derived as follows:*

$$pDsY[\,0\,][\,y\,] = (\,pY[\,-1\,][\,2*y\,] + pY[\,-1\,][\,2*y+1\,] + 2*pY[\,0\,][\,2*y\,] + 2*pY[\,0\,][\,2*y+1\,] + pY[\,1\,][\,2*y\,] + pY[\,1\,][\,2*y+1\,] + 4\,) >> 3 \quad (8\text{-}173)$$

– *Otherwise, pDsY[0][y] with y = 0..nTbH − 1 is derived as follows:*

$$pDsY[\,0\,][\,y\,] = (\,pY[\,0\,][\,2*y\,] + pY[\,0\,][\,2*y+1\,] + 1\,) >> 1 \qquad (8\text{-}174)$$

4. *When numSampL is greater than 0,* ***the selected neighbouring left chroma samples pSelC[idx] are set equal to p[-1][pickPosL[idx]] with idx = 0..(cntL – 1), and*** *the* ***selected*** *down-sampled neighbouring left luma samples* ***pSelDsY[idx]*** *with* ***idx = 0..(cntL-1)*** *are derived as follows:*

   – ***The variable y is set equal to pickPosL[idx].***

   – *If sps_cclm_colocated_chroma_flag is equal to 1, the following applies:*

      – ***If y > 0 || availTL == TRUE,***

$$pSelDsY[\,idx\,] = (\,pY[\,-2\,][\,2*y-1\,] + pY[\,-3\,][\,2*y\,] + 4*pY[\,-2\,][\,2*y\,] + pY[\,-1\,][\,2*y\,] + pY[\,-2\,][\,2*y+1\,] + 4\,) >> 3 \qquad (8\text{-}175)$$

      – *Otherwise,*

$$pSelDsY[\,idx\,] = (\,pY[\,-3\,][\,0\,] + 2*pY[\,-2\,][\,0\,] + pY[\,-1\,][\,0\,] + 2\,) >> 2 \qquad (8\text{-}177)$$

   – *Otherwise, the following applies:*

$$pSelDsY[\,idx\,] = (\,pY[\,-1\,][\,2*y\,] + pY[\,-1\,][\,2*y+1\,] + 2*pY[\,-2\,][\,2*y\,] + 2*pY[\,-2\,][\,2*y+1\,] + pY[\,-3\,][\,2*y\,] + pY[\,-3\,][\,2*y+1\,] + 4\,) >> 3 \qquad (8\text{-}178)$$

5. *When numSampT is greater than 0,* ***the selcted neighbouring top chroma samples pSelC[idx] are set equal to p[pickPosT[idx – cntL]][-1] with idx = cntL..(cntL + cntT – 1), and*** *the down-sampled neighbouring top luma samples* ***pSelDsY[idx] with idx = cntL..(cntL + cntT – 1)*** *are specified as follows:*

   – ***The variable x is set equal to pickPosT[idx – cntL].***

   – *If sps_cclm_colocated_chroma_flag is equal to 1, the following applies:*

      – *If x > 0:*

         – *If bCTUboundary is equal to FALSE, the following applies:*

$$pSelDsY[\,idx\,] = (\,pY[\,2*x\,][\,-3\,] + pY[\,2*x-1\,][\,-2\,] + 4*pY[\,2*x\,][\,-2\,] + pY[\,2*x+1\,][\,-2\,] + pY[\,2*x\,][\,-1\,] + 4\,) >> 3 \qquad (8\text{-}179)$$

         – *Otherwise (bCTUboundary is equal to TRUE), the following applies:*

$$pSelDsY[\,idx\,] = (\,pY[\,2*x-1\,][\,-1\,] + 2*pY[\,2*x\,][\,-1\,] + pY[\,2*x+1\,][\,-1\,] + 2\,) >> 2 \qquad (8\text{-}180)$$

      – ***Otherwise:***

         – *If availTL is equal to TRUE and bCTUboundary is equal to FALSE, the following applies:*

$$pSelDsY[idx] = (pY[0][-3] + pY[-1][-2] + 4*pY[0][-2] + pY[1][-2] + pY[0][-1] + 4) >> 3 \quad (8\text{-}181)$$

– *Otherwise if availTL is equal to TRUE and bCTUboundary is equal to TRUE, the following applies:*

$$pSelDsY[idx] = (pY[-1][-1] + 2*pY[0][-1] + pY[1][-1] + 2) >> 2 \quad (8\text{-}182)$$

– *Otherwise if availTL is equal to FALSE and bCTUboundary is equal to FALSE, the following applies:*

$$pSelDsY[idx] = (pY[0][-3] + 2*pY[0][-2] + pY[0][-1] + 2) >> 2 \quad (8\text{-}183)$$

– *Otherwise (availTL is equal to FALSE and bCTUboundary is equal to TRUE), the following applies:*

$$pSelDsY[idx] = pY[0][-1] \quad (8\text{-}184)$$

– *Otherwise, the following applies:*

– ***If x > 0:***

– *If bCTUboundary is equal to FALSE, the following applies:*

$$pSelDsY[idx] = (pY[2*x-1][-2] + pY[2*x-1][-1] + 2*pY[2*x][-2] + 2*pY[2*x][-1] + pY[2*x+1][-2] + pY[2*x+1][-1] + 4) >> 3 \quad (8\text{-}185)$$

– *Otherwise (bCTUboundary is equal to TRUE), the following applies:*

$$pSelDsY[idx] = (pY[2*x-1][-1] + 2*pY[2*x][-1] + pY[2*x+1][-1] + 2) >> 2 \quad (8\text{-}186)$$

– ***Otherwise:***

– *If availTL is equal to TRUE and bCTUboundary is equal to FALSE, the following applies:*

$$pSelDsY[idx] = (pY[-1][-2] + pY[-1][-1] + 2*pY[0][-2] + 2*pY[0][-1] + pY[1][-2] + pY[1][-1] + 4) >> 3 \quad (8\text{-}187)$$

– *Otherwise if availTL is equal to TRUE and bCTUboundary is equal to TRUE, the following applies:*

$$pSelDsY[idx] = (pY[-1][-1] + 2*pY[0][-1] + pY[1][-1] + 2) >> 2 \quad (8\text{-}188)$$

– *Otherwise if availTL is equal to FALSE and bCTUboundary is equal to FALSE, the following applies:*

$$pSelDsY[\,idx\,] = (\,pY[\,0\,][\,-2\,] + pY[\,0\,][\,-1\,] + 1\,) >> 1 \qquad (8\text{-}189)$$

– *Otherwise (availTL is equal to FALSE and bCTUboundary is equal to TRUE), the following applies:*

$$pSelDsY[\,idx\,] = pY[\,0\,][\,-1\,] \qquad (8\text{-}190)$$

6. ***When cntT+ cntL is not equal to 0, the variables minY, maxY, minC and maxC are derived as follows:***

– ***When cntT+cntL is equal to 2, set pSelComp[3] equal to pSelComp[0], pSelComp[2] equal to pSelComp[1], pSelComp[0] equal to pSelComp[1], and pSelComp[1] equal to pSelComp[3], with Comp being replaced by DsY and C.***

– ***The arrays minGrpIdx[] and maxGrpIdx[] are set as: minGrpIdx[0] = 0, minGrpIdx[1] = 2, maxGrpIdx[0] = 1, maxGrpIdx[1] = 3.***

– ***If pSelDsY[minGrpIdx[0]] > pSelDsY[minGrpIdx[1]], Swap(minGrpIdx[0], minGrpIdx[1]).***

– ***If pSelDsY[maxGrpIdx[0]] > pSelDsY[maxGrpIdx[1]], Swap(maxGrpIdx[0], maxGrpIdx[1]).***

– ***If pSelDsY[minGrpIdx[0]] > pSelDsY[maxGrpIdx[1]], Swap(minGrpIdx, maxGrpIdx).***

– ***If pSelDsY[minGrpIdx[1]] > pSelDsY[maxGrpIdx[0]], Swap(minGrpIdx[1], maxGrpIdx[0]).***

– ***maxY = (pSelDsY[maxGrpIdx[0]] + pSelDsY[maxGrpIdx[1]] + 1) >> 1.***

– ***maxC = (pSelC[maxGrpIdx[0]] + pSelC[maxGrpIdx[1]] + 1) >> 1.***

– ***minY = (pSelDsY[minGrpIdx[0]] + pSelDsY[minGrpIdx[1]] + 1) >> 1.***

– ***minC = (pSelC[minGrpIdx[0]] + pSelC[minGrpIdx[1]] + 1) >> 1.***

7. *The variables a, b, and k are derived as follows:*

– *If numSampL is equal to 0, and numSampT is equal to 0, the following applies:*

$$k = 0 \qquad (8\text{-}208)$$

$$a = 0 \qquad (8\text{-}209)$$

$$b = 1 << (\,BitDepth_C - 1) \qquad (8\text{-}210)$$

– *Otherwise, the following applies:*

$$diff = maxY - minY \qquad (8\text{-}211)$$

–If diff is not equal to 0, the following applies:

$$diffC = maxC - minC \qquad (8\text{-}212)$$

$$x = Floor(\,Log2(\,diff\,)\,) \qquad (8\text{-}213)$$

$$normDiff = (\,(\,diff << 4\,) >> x\,) \,\&\, 15 \qquad (8\text{-}214)$$

$$x \mathrel{+}= (normDiff \mathrel{!}= 0) \ ? \ 1 \ : \ 0 \quad (8\text{-}215)$$

$$y = Floor(Log2(Abs (diffC))) + 1 \quad (8\text{-}216)$$

$$a = (diffC * (divSigTable[normDiff] \mid 8) + 2^{y-1}) >> y \quad (8\text{-}217)$$

$$k = ((3 + x - y) < 1) \ ? \ 1 \ : \ 3 + x - y \quad (8\text{-}218)$$

$$a = ((3 + x - y) < 1) \ ? \ Sign(a) * 15 \ : \ a \quad (8\text{-}219)$$

$$b = minC - ((a * minY) >> k) \quad (8\text{-}220)$$

where divSigTable[ ] is specified as follows:

$$divSigTable[ ] = \{ 0, 7, 6, 5, 5, 4, 4, 3, 3, 2, 2, 1, 1, 1, 1, 0 \} \quad (8\text{-}221)$$

–Otherwise (diff is equal to 0), the following applies:

$$k = 0 \quad (8\text{-}222)$$

$$a = 0 \quad (8\text{-}223)$$

$$b = minC \quad (8\text{-}224)$$

8. *The prediction samples predSamples[x][y] with x = 0..nTbW − 1, y = 0.. nTbH − 1 are derived as follows:*

$$predSamples[x][y] = Clip1C(((pDsY[x][y] * a) >> k) + b) \quad (8\text{-}225)$$

2.2.3.3 Miscellaneous intra prediction aspects

[0051] VTM4 includes many intra coding tools which are different from HEVC, for example, the following features have been included in the VVC test model 3 on top of the bock tree structure.

- 67 intra mode with wide angles mode extension
- Block size and mode dependent 4 tap interpolation filter
- Position dependent intra prediction combination (PDPC)
- Cross component linear model intra prediction
- Multi-reference line intra prediction
- Intra sub-partitions

2.2.4 Inter prediction in VVC

2.2.4.1 Combined inter and intra prediction (CIIP)

[0052] In VTM4, when a CU is coded in merge mode, and if the CU contains at least 64 luma samples (that is, CU width times CU height is equal to or larger than 64), an additional flag is signalled to indicate if the combined inter/intra prediction (CIIP) mode is applied to the current CU.

[0053] In order to form the CIIP prediction, an intra prediction mode is first derived from two additional syntax elements. Up to four possible intra prediction modes can be used: DC, planar, horizontal, or vertical. Then, the inter prediction and intra prediction signals are derived using regular intra and inter decoding processes. Finally, weighted averaging of the inter and intra prediction signals is performed to obtain the CIIP prediction.

2.2.4.2 Miscellaneous inter prediction aspects

[0054] VTM4 includes many inter coding tools which are different from HEVC, for example, the following features have been included in the VVC test model 3 on top of the bock tree structure.

- Affine motion inter prediction
- sub-block based temporal motion vector prediction
- Adaptive motion vector resolution
- 8x8 block based motion compression for temporal motion prediction
- High precision (1/16 pel) motion vector storage and motion compensation with 8-tap interpolation filter for luma component and 4-tap interpolation filter for chroma component
- Triangular partitions
- Combined intra and inter prediction
- Merge with MVD (MMVD)
- Symmetrical MVD coding
- Bi-directional optical flow
- Decoder side motion vector refinement
- Bi-predictive weighted averaging

2.2.5 In-loop filters

[0055] There are totally three in-loop filters in VTM4. Besides deblocking filter and SAO (the two loop filters in HEVC), adaptive loop filter (ALF) are applied in the VTM4. The order of the filtering process in the VTM4 is the deblocking filter, SAO and ALF.

[0056] In the VTM4, the SAO and deblocking filtering processes are almost same as those in HEVC.

[0057] In the VTM4, a new process called the luma mapping with chroma scaling was added (this process was previously known as the adaptive in-loop reshaper). This new process is performed before deblocking.

2.2.6 Luma mapping with chroma scaling (LMCS, aka. in-loop reshaping)

[0058] In VTM4, a coding tool called the luma mapping with chroma scaling (LMCS) is added as a new processing block before the loop filters. LMCS has two main components: 1) in-loop mapping of the luma component based on adaptive piecewise linear models; 2) for the chroma components, luma-dependent chroma residual scaling is applied. FIG. 4 shows the LMCS architecture from decoder's perspective. The light-blue shaded blocks in FIG. 4 indicate where the processing is applied in the mapped domain; and these include the inverse quantization, inverse transform, luma intra prediction and adding of the luma prediction together with the luma residual. The unshaded blocks in FIG. 4 indicate where the processing is applied in the original (i.e., non-mapped) domain; and these include loop filters such as deblocking, ALF, and SAO, motion compensated prediction, chroma intra prediction, adding of the chroma prediction together with the chroma residual, and storage of decoded pictures as reference pictures. The light-yellow shaded blocks in FIG. 4 are the new LMCS functional blocks, including forward and inverse mapping of the luma signal and a luma-dependent chroma scaling process. Like most other tools in VVC, LMCS can be enabled/disabled at the sequence level using an SPS flag.

2.2.6.1 Luma mapping with piecewise linear model

[0059] The in-loop mapping of the luma component adjusts the dynamic range of the input signal by redistributing the codewords across the dynamic range to improve compression efficiency. Luma mapping makes use of a forward mapping function, *FwdMap*, and a corresponding inverse mapping function, *InvMap*. The *FwdMap* function is signalled using a piecewise linear model with 16 equal pieces. *InvMap* function does not need to be signalled and is instead derived from the *FwdMap* function.

[0060] The luma mapping model is signalled at the tile group level. A presence flag is signalled first. If luma mapping model is present in the current tile group, corresponding piecewise linear model parameters are signalled. The piecewise linear model partitions the input signal's dynamic range into 16 equal pieces, and for each piece, its linear mapping parameters are expressed using the number of codewords assigned to that piece. Take 10-bit input as an example. Each of the 16 pieces will have 64 codewords assigned to it by default. The signalled number of codewords is used to calculate the scaling factor and adjust the mapping function accordingly for that piece. At the tile group level, another LMCS enable flag is signalled to indicate if the LMCS process as depicted in FIG. 4 is applied to the current tile group.

[0061] Each i-th piece, i = 0 … 15, of the *FwdMap* piecewise linear model is defined by two input pivot points InputPivot[] and two output (mapped) pivot points MappedPivot[].

[0062] The InputPivot[] and MappedPivot[] are computed as follows (assuming 10-bit video):

1) OrgCW = 64
2) For i = 0:16, InputPivot[i] = i * OrgCW

3) For i=0:16, MappedPivot[i] is calculated as follows:

MappedPivot[0] = 0;

for(i = 0; i <16 ; i++)

MappedPivot[i + 1] = MappedPivot[i] + SignalledCW[i]

where SignalledCW[i] is the signalled number of codewords for the i-th piece.

[0063] As shown in FIG. 4, for an inter-coded block, motion compensated prediction is performed in the mapped domain. In other words, after the motion-compensated prediction block Y_{pred} is calculated based on the reference signals in the DPB, the *FwdMap* function is applied to map the luma prediction block in the original domain to the mapped domain, $Y'_{pred} = FwdMap(Y_{pred})$. For an intra-coded block, the *FwdMap* function is not applied because intra prediction is performed in the mapped domain. After reconstructed block Y_r is calculated, the *InvMap* function is applied to convert the reconstructed luma values in the mapped domain back to the reconstructed luma values in the original domain ($\hat{Y}_i = InvMap(Y_r)$). The *InvMap* function is applied to both intra- and inter-coded luma blocks.

[0064] The luma mapping process (forward and/or inverse mapping) can be implemented using either look-up-tables (LUT) or using on-the-fly computation. If LUT is used, then $FwdMapLUT$ and $InvMapLUT$ can be pre-calculated and pre-stored for use at the tile group level, and forward and inverse mapping can be simply implemented as $FwdMap(Y_{pred}) = FwdMapLUT[Y_{pred}]$ and $InvMap(Y_r) = InvMapLUT[Y_r]$, respectively. Alternatively, on-the-fly computation may be used. Take forward mapping function *FwdMap* as an example. In order to figure out the piece to which a luma sample belongs, the sample value is right shifted by 6 bits (which corresponds to 16 equal pieces). Then, the linear model parameters for that piece are retrieved and applied on-the-fly to compute the mapped luma value. Let i be the piece index, a1, a2 be InputPivot[i] and InputPivot[i+1], respectively, and b1, b2 be MappedPivot[i] and MappedPivot[i+1], respectively. The FwdMap function is evaluated as follows:

$$FwdMap(Y_{pred}) = ((\mathrm{b2} - \mathrm{b1})/(\mathrm{a2} - \mathrm{a1})) * (Y_{pred} - \mathrm{a1}) + \mathrm{b1}$$

[0065] The InvMap function can be computed on-the-fly in a similar manner, except that conditional checks need to be applied instead of a simple right bit-shift when figuring out the piece to which the sample value belongs, because the pieces in the mapped domain are not equal sized.

2.2.6.2 Luma-dependent chroma residual scaling

[0066] Chroma residual scaling is designed to compensate for the interaction between the luma signal and its corresponding chroma signals. Whether chroma residual scaling is enabled or not is also signalled at the tile group level. If luma mapping is enabled and if dual tree partition (also known as separate chroma tree) is not applied to the current tile group, an additional flag is signalled to indicate if luma-dependent chroma residual scaling is enabled or not. When luma mapping is not used, or when dual tree

partition is used in the current tile group, luma-dependent chroma residual scaling is disabled. Further, luma-dependent chroma residual scaling is always disabled for the chroma blocks whose area is less than or equal to 4.

[0067] Chroma residual scaling depends on the average value of the corresponding luma prediction block (for both intra- and inter-coded blocks). Denote $avgY'$ as the average of the luma prediction block. The value of $C_{ScaleInv}$ is computed in the following steps:

1) Find the index Y_{Idx} of the piecewise linear model to which $avgY'$ belongs based on the *InvMap* function.

2) $C_{ScaleInv}$ = cScaleInv[Y_{Idx}], where cScaleInv[] is a pre-computed 16-piece LUT.

If the current block is coded as intra, CIIP, or intra block copy (IBC, a.k.a. current picture referencing or CPR) modes, **$avgY'$ is computed as the average of the intra-, CIIP-, or IBC- predicted luma values**; **otherwise, $avgY'$ is computed as the average of the forward mapped inter predicted luma values** (Y'_{pred} in FIG. 4). Unlike luma mapping, which is performed on the sample basis, $C_{ScaleInv}$ is a constant value for the entire chroma block. With $C_{ScaleInv}$, chroma residual scaling is applied as follows:

Encoder side: $C_{ResScale} = C_{Res} * C_{Scale} = C_{Res}/C_{ScaleInv}$

Decoder side: $C_{Res} = C_{ResScale}/C_{Scale} = C_{ResScale} * C_{ScaleInv}$

2.2.6.3 Corresponding working draft in JVET-M1001_v7 with the adoption in JVET-N0220

[0068] The following spec is based on the modified working draft of JVET-M1001 and the adoption in JVET-N0220. The modification in the adopted JVET-N0220 is shown in bold and underlining.

Syntax tables

In 7.3.2.1 Sequence parameter set RBSP syntax

sps_triangle_enabled_flag	*u(1)*
sps_lmcs_enabled_flag	*u(1)*
sps_ladf_enabled_flag	*u(1)*

In 7.3.4.1 General tile group header syntax

if (sps_lmcs_enabled_flag) {	
tile_group_lmcs_model_present_flag	*u(1)*
if (tile_group_lmcs_model_present_flag)	
lmcs_data( )	
tile_group_lmcs_enabled_flag	*u(1)*
if (tile_group_lmcs_enabled_flag && !(qtbtt_dual_tree_intra_flag && tile_group_type = = I))	
tile_group_chroma_residual_scale_flag	*u(1)*

}	
if(NumTilesInCurrTileGroup > 1) {	
offset_len_minus1	ue(v)
for(i = 0; i < NumTilesInCurrTileGroup − 1; i++)	
entry_point_offset_minus1[i]	u(v)
}	

In 7.3.4.4 Luma mapping with chroma scaling data syntax

lmcs_data () {	Descriptor
lmcs_min_bin_idx	ue(v)
lmcs_delta_max_bin_idx	ue(v)
lmcs_delta_cw_prec_minus1	ue(v)
for (i = lmcs_min_bin_idx; i <= LmcsMaxBinIdx; i++) {	
lmcs_delta_abs_cw[i]	u(v)
if (lmcs_delta_abs_cw[i]) > 0)	-
lmcs_delta_sign_cw_flag[i]	u(1)
}	
}	

Semantics

In 7.4.3.1 Sequence parameter set RBSP semantics

sps_lmcs_enabled_flag *equal to 1 specifies that luma mapping with chroma scaling is used in the CVS. sps_lmcs_enabled_flag equal to 0 specifies that luma mapping with chroma scaling is not used in the CVS.*

tile_group_lmcs_model_present_flag *equal to 1 specifies that lmcs_data() is present in the tile group header. tile_group_lmcs_model_present_flag equal to 0 specifies that lmcs_data() is not present in the tile group header. When tile_group_lmcs_model_present_flag is not present, it is inferred to be equal to 0.*

tile_group_lmcs_enabled_flag *equal to 1 specifies that luma mappin with chroma scaling is enabled for the current tile group. tile_group_lmcs_enabled_flag equal to 0 specifies that luma mapping with chroma scaling is not enabled for the current tile group. When tile_group_lmcs_enabled_flag is not present, it is inferred to be equal to 0.*

tile_group_chroma_residual_scale_flag *equal to 1 specifies that chroma residual scaling is enabled for the current tile group. tile_group_ chroma_residual_scale_flag equal to 0 specifies that chroma residual scaling is not enabled for the current tile group. When tile_group_ chroma_residual_scale_flag is not present, it is inferred to be equal to 0.*

In 7.4.5.4 Luma mapping with chroma scaling data semantics

lmcs_min_bin_idx *specifies the minimum bin index used in the luma mapping with chroma scaling construction process. The value of lmcs_min_bin_idx shall be in the range of 0 to 15, inclusive.*

lmcs_delta_max_bin_idx *specifies the delta value between 15 and the maximum bin index LmcsMaxBinIdx used in the luma mapping with chroma scaling construction process. The value of lmcs_delta_max_bin_idx shall be in the range of 0 to 15, inclusive. The value of LmcsMaxBinIdx is set equal to 15 − lmcs_delta_max_bin_idx. The value of LmcsMaxBinIdx shall be larger than or equal to lmcs_min_bin_idx.*

lmcs_delta_cw_prec_minus1 *plus 1 specifies the number of bits used for the representation of the syntax lmcs_delta_abs_cw[i]. The value of lmcs_delta_cw_prec_minus1 shall be in the range of 0 to BitDepthY − 2, inclusive.*

lmcs_delta_abs_cw*[i] specifies the absolute delta codeword value for the ith bin.*

lmcs_delta_sign_cw_flag*[i] specifies the sign of the variable lmcsDeltaCW[i] as follows:*

- *If lmcs_delta_sign_cw_flag[i] is equal to 0, lmcsDeltaCW[i] is a positive value.*
- *Otherwise (lmcs_delta_sign_cw_flag[i] is not equal to 0), lmcsDeltaCW[i] is a negative value.*

When lmcs_delta_sign_cw_flag[i] is not present, it is inferred to be equal to 0.

The variable OrgCW is derived as follows:

$$OrgCW = (1 << BitDepth_Y) / 16 \quad (7\text{-}70)$$

The variable lmcsDeltaCW[i], with i = lmcs_min_bin_idx..LmcsMaxBinIdx, is derived as follows:

$$lmcsDeltaCW[i] = (1 - 2 * lmcs_delta_sign_cw_flag[i]) * lmcs_delta_abs_cw[i] \quad (7\text{-}71)$$

The variable lmcsCW[i] is derived as follows:

- *For i = 0.. lmcs_min_bin_idx − 1, lmcsCW[i] is set equal 0.*
- *For i = lmcs_min_bin_idx..LmcsMaxBinIdx, the following applies:*

$$lmcsCW[i] = OrgCW + lmcsDeltaCW[i] \quad (7\text{-}72)$$

The value of lmcsCW[i] shall be in the range of (OrgCW>>3) to (OrgCW<<3 − 1), inclusive.

- *For i = LmcsMaxBinIdx + 1..15, lmcsCW[i] is set equal 0.*

It is a requirement of bitstream conformance that the following condition is true:

$$\sum_{i=0}^{15} lmcsCW[i] <= (1 << BitDepth_Y) - 1 \quad (7\text{-}73)$$

The variable InputPivot[i], with i = 0..16, is derived as follows:

*InputPivot[i] = i * OrgCW* (7-74)

The variable LmcsPivot[i] with i = 0..16, the variables ScaleCoeff[i] and InvScaleCoeff[i] with i = 0..15, are derived as follows:

LmcsPivot[0] = 0;
for(i = 0; i <= 15; i++) {
LmcsPivot[i + 1] = LmcsPivot[i] + lmcsCW[i]
ScaleCoeff[i] = (lmcsCW[i] * (1 << 11) + (1 << (Log2(OrgCW) − 1))) >> (Log2(OrgCW)) ***(7-75)***
if (lmcsCW[i] = = 0)
InvScaleCoeff[i] = 0
else
InvScaleCoeff[i] = OrgCW * (1 << 11) / lmcsCW[i]
}

The variable ChromaScaleCoeff[i], with i = 0... 15, is derived as follows:

if (lmcsCW[i] = = 0)
ChromaScaleCoeff[i] = (1 << 11)
else {
ChromaScaleCoeff[i] = InvScaleCoeff[i]
}

The variables ClipRange, LmcsMinVal, and LmcsMaxVal are derived as follows:

ClipRange = ((lmcs_min_bin_idx > 0) && (LmcsMaxBinIdx < 15) (7-77)

$LmcsMinVal = 16 << (BitDepth_Y - 8)$ (7-78)

$LmcsMaxVal = 235 << (BitDepth_Y - 8)$ (7-79)

NOTE – Arrays InputPivot[i] and LmcsPivot[i], ScaleCoeff[i], and InvScaleCoeff[i], ChromaScaleCoeff[i], ClipRange, LmcsMinVal and LmcsMaxVal, are updated only when tile_group_lmcs_model_present_flag is equal to 1. Thus, the lmcs model may be sent with an IRAP picture, for example, but lmcs is disabled for that IRAP picture.

3. Drawbacks of existing implementations

[0069] The current design of LMCS/CCLM may have the following problems:

1. In LMCS coding tool, the chroma residual scaling factor is derived by the average value of the collocated luma prediction block, which results in a latency for processing the chroma samples in LMCS chroma residual scaling.

a) In case of single/shared tree, the latency is caused by (a) waiting for all the prediction samples of the whole luma block available, and (b) averaging all the luma prediction samples obtained by (a).

b) In case of dual/separate tree, the latency is even worse since separate block partitioning structure for luma and chroma components is enabled in I slices. Therefore, one chroma block may correspond to multiple luma blocks, and one 4x4 chroma block may correspond to a 64x64 luma block. Thus the worst case is that the chroma residual scaling factor of current 4x4 chroma block may need to wait until all the prediction samples in the whole 64x64 luma block are available. In a word, the latency issue in dual/separate tree would be much more serious.

2. In CCLM coding tool, the CCLM model computation for intra chroma prediction depends on the left and above reference samples of both luma block and chroma block. And the CCLM prediction for a chroma block depends on the collocated luma reconstructed samples of the same CU. This would cause high latency in dual/separate tree.

- In case of dual/separate tree, one 4x4 chroma block may correspond to a 64x64 luma block. Thus the worst case is that the CCLM process for the current chroma block may need wait until the corresponding whole 64x64 luma block being reconstructed. This latency issue is similar as LMCS chroma scaling in dual/separate tree.

4. Example techniques and embodiments

[0070] To tackle the problems, we propose several methods to remove/reduce/restrict the cross-component dependency in luma-dependent chroma residual scaling, CCLM, and other coding tools that rely on information from a different colour component.

[0071] The detailed embodiments described below should be considered as examples to explain general concepts. These embodiments should not be interpreted narrowly way. Furthermore, these embodiments can be combined in any manner.

[0072] It is noted that although the bullets described below explicitly mention LMCS/CCLM, the methods may be also applicable to other coding tools that rely on information from a different colour component. In addition, the term 'luma' and 'chroma' mentioned below may be replaced by 'a first color component' and 'a second color component' respectively, such as 'G component' and 'B/R component' in the RGB color format.

[0073] In the following discussion, the definition a "collocated sample/block" aligns with the definition of collocated sample/block in VVC working draft JVET-M1001. To be more specific, in 4:2:0 colour format, suppose the top-left sample of a chroma block is at position (xTbC, yTbC), then the top-left sample of the collocated luma block location (xTbY, yTbY) is derived as follows:

(xTbY, yTbY) = (xTbC << 1, yTbC << 1). As illustrated in FIG. 5, the top-left sample of the current chroma block is located at (x=16,y=16) in the chroma picture, then the top-left sample of its collocated luma block is located at (x=32,y=32) in the luma picture, regardless of the block partition of collocated luma block in the luma picture. For another example, saying in the same color component, the location of the top-left sample of the collocated block in the reference frame should be same with the location of the top-left sample of the current block in the current frame, as illustrated in FIG. 6, suppose the top-left sample of the current block is (x,y) in the current frame, then the top-left sample of the collocated block of the current block have the same location (x,y) in the reference frame.

[0074] In the following discussion, a "corresnponding block" may have different location with the current block. For an example, there might be a motion shift between the current block and its corresponding block in the reference frame. As illustrated in FIG. 6, suppose the current block is located at (x,y) in the current frame and it has a motion vector (mv_x, mv_y), then a corresponding block of the current block may be located at ($x+mv_x$,$y+mv_y$) in the reference frame. And for an IBC coded block, the collocated luma block (pointed by zero vector) and the corresponding luma block (pointed by non-zero-BV) may locate in different places of the current frame. For another example, when the partition of luma block doesn't align with the partition of chroma block (in dual tree partition of I slices), the collocated luma block of the current chroma block may belong to a larger luma block which depends on the partition size of the overlapped luma coding block covering the top-left sample of the collocated luma block. As illustrated in FIG. 5, assume the bold rectangle denotes the partitions of the block, so that a 64x64 luma block is firstly split by an BT and then the right part of the 64x64 luma block is further split by a TT, which results in three luma blocks of sizes equal to 32x16, 32x32, 32x16, respectively. Thus looking at the top-left sample (x=32, y=32) of the collocated luma block of the current chroma block, it belongs to the center 32x32 luma block of the TT partition. In this case, we call the corresnponding luma block that covers the top-left sample of the collocated luma block as a "corresponding luma block". Therefore, in this example, the top-left sample of the corresponding luma block is located at (x=32, y=16).

[0075] Hereinafter, DMVD (decoder-side motion vector derivation) is used to represent BDOF (a.k.a BIO) or/and DMVR (decode-side motion vector refinement) or/and FRUC (frame rate up-conversion) or/and other method that refines motion vector or/and prediction sample value at decoder.

Removal of the chroma scaling latency of LMCS and model computation of CCLM

1. It is proposed that for an inter-coded block, one or multiple reference samples of the current block in reference frames may be used to derive the chroma residual scaling factor in the LMCS mode.
    a) In one example, reference luma samples may be directly used to derive the chroma residual scaling factor.

i. Alternatively, interpolation may be firstly applied to reference samples and the interpolated samples may be used to derive the chroma residual scaling factor.

ii. Alternatively, reference samples in different reference frames may be utilized to derive the final reference samples that are used for the chroma residual scaling factor derivation.

1) In one example, for bi-prediction coded blocks, the above method may be applied.

iii. In one example, the intensities of reference samples may be converted to reshaping domain before being used to derive the chroma residual scaling factor.

iv. In one example, linear combination of the reference samples may be used to derive the chroma residual scaling factor.

1) For example, $a \times S + b$ may be used to derive the chroma residual scaling factor, where S is a reference sample, a and b are parameters. In one example, a and b may be derived by Localized Illuminate Compensation (LIC).

b) In one example, the location of the reference luma samples in the reference frame may depend on the current block's motion vector(s).

i. In one example, a reference sample belongs to a reference luma block, which is in a reference picture, and with the same width and height as the current luma block. The position of the reference luma sample in the reference picture may be calculated as the position of its corresponding luma sample in the current picture, adding a motion vector.

ii. In one example, the position of the reference luma samples may be derived by the position of top-left (or center, or bottom-right) sample of the current luma block and current block's motion vector, referred as a corresponding luma sample in the reference frame.

1) In one example, an integer motion vector may be used to derive the corresponding luma sample in the reference frame. In one example, the motion vector associated with one block may be either rounded toward zero, or rounded away from zero to derive the integer motion vector.

2) Alternatively, a fractional motion vector may be used to derive the corresponding luma sample in the reference frame, so that the interpolation process may be required to derive the fractional reference samples.

iii. Alternatively, the position of the reference luma samples may be derived by the position of top-left (or center, or bottom-right) sample of current luma block.

iv. Alternatively, multiple corresponding luma samples at some pre-defined positions in the reference frame may be picked to calculate the chroma residual scaling factor.

c) In one example, the median or average value of the multiple reference luma samples may be used to derive the chroma residual scaling factor.

d) In one example, the reference luma samples in pre-defined reference frames may be used to derive the chroma residual scaling factor.

i. In one example, the pre-defined reference frame may be the one with reference index equal to 0 of reference picture list 0.

ii. Alternatively, the reference index and/or reference picture list for the pre-defined reference frame may be signaled in sequence/picture/tile group/slice/tile/CTU row/video unit level.

iii. Alternatively, the reference luma samples in multiple reference frames may be derived and the averaged or weighted average values may be utilized to get the chroma residual scaling factor.

2. It is proposed that whether and how to derive the chroma residual scaling factor from luma samples in the LMCS mode may depend on whether the current block applies bi-prediction.

a) In one example, the chroma residual scaling factor is derived for each prediction direction individually.

3. It is proposed that whether and how to derive the chroma residual scaling factor from luma samples in the LMCS mode may depend on whether the current block applies sub-block-based prediction.

a) In one example, the sub-block-based prediction is affine prediction;

b) In one example, the sub-block-based prediction is Alternative Temporal Motion Vector Prediction (ATMVP).

c) In one example, the chroma residual scaling factor is derived for each sub-block individually.

d) In one example, the chroma residual scaling factor is derived for the whole block even if it is predicted by sub-blocks.

i. In one example, motion vector of one selected sub-block (e.g., top-left sub-block) may be used to identify the reference samples of current block as described in bullet 1.

4. It is proposed that the luma prediction values used to derive the chroma residual scaling factor may be intermediate luma prediction value instead of the final luma prediction value.

a) In one example, the luma prediction values before the process of Bi-Directional Optical Flow (BDOF, a.k.a. BIO) may be used to derive the chroma residual scaling factor.

b) In one example, the luma prediction values before the process of Decoder-side Motion Vector Refinement (DMVR) may be used to derive the chroma residual scaling factor.

c) In one example, the luma prediction values before the process of LIC may be used to derive the chroma residual scaling factor.

d) In one example, the luma prediction values before the process of Prediction Refinement Optical Flow (PROF) as proposed in JVET-N0236 may be used to derive the chroma residual scaling factor.

5. Intermediate motion vectors may be used to identify the reference samples.

a) In one example, motion vector before the process of BDOF or/and DMVR or/and other DMVD methods may be used to identify the reference samples.

b) In one example, the motion vector before the process of Prediction Refinement Optical Flow (PROF) as proposed in JVET-N0236 may be used to identify the reference samples.

6. The above methods may be applicable when the current block is coded with inter mode.

7. It is proposed that for an IBC-coded block, one or multiple reference samples in reference block of current frame may be used to derive the chroma residual scaling factor in the LMCS mode. When the block IBC-coded, the term "motion vector" may also be referred as "block vector", where the reference picture is set as the current picture.

a) In one example, a reference sample belongs to a reference block, which is in the current picture, and with the same width and height as the current block. The position of the reference sample may be calculated as the position of its corresponding sample adding a motion vector.

b) In one example, the position of the reference luma samples may be derived by the position of top-left (or center, or bottom-right) sample of current luma block adding a motion vector.

c) Alternatively, the position of the reference luma samples may be derived by the position of top-left (or center, or bottom-right) sample of current luma block adding current block's block vector.

d) Alternatively, multiple corresponding luma samples at some pre-defined positions in the reference region of current luma block may be picked to calculate the chroma residual scaling factor.

e) In one example, multiple corresponding luma samples may be computed with a function to derive the chroma residual scaling factor.

i. For example, the median or average value of multiple corresponding luma samples may be computed to derive the chroma residual scaling factor.

f) In one example, the intensities of reference samples may be converted to reshaping domain before being used to derive the chroma residual scaling factor.

i. Alternatively, the intensities of reference samples may be converted to original domain before being used to derive the chroma residual scaling factor

8. It is proposed that one or multiple prediction/reconstructed samples which are located at the identified location(s) of the current luma block in the current frame may be used to derive the chroma residual scaling factor for the current chroma block in the LMCS mode.

a) In one example, if current block is inter-coded, the luma prediction (or reconstruction) sample located in the center of the current luma block may be picked to derive the chroma residual scaling factor.

b) In one example, the average value of the first MxN luma prediction (or reconstruction) samples may be picked to derive the chroma residual scaling factor, where MxN could be smaller than collocated luma block size widthxheight.

9. It is proposed that the whole or partial of the procedure used to calculate the CCLM model may be used for the chroma residual scaling factor derivation of current chroma block in the LMCS mode.

a) In one example, reference samples which are located at the identified locations of neighboring luma samples of the collocated luma block in CCLM model parameter derivation process may be utilized to derive chroma residual scaling factor.

i. In one example, those reference samples may be directly used.

ii. Alternatively, downsampling may be applied to those reference samples, and downsampled reference samples may be applied.

b) In one example, K out of S reference samples selected for CCLM model computation may be used for chroma residual scaling factor derivation in the LMCS mode. E.g., K is equal to 1 and S is equal to 4.

c) In one example, the average/minimum/maximum value of the reference samples of the collocated luma block in CCLM mode may be used for chroma residual scaling factor derivation in the LMCS mode.

10. How to select samples for derivation of chroma residual scaling factors may be dependent on the coded information of current block.

a) The coded information may include QP, coding mode, POC, intra-prediction mode, motion information and so on.

b) In one example, for IBC coded or Non-IBC coded blocks, the way to select samples may be different.

c) In one example, the way to select samples may be different based on the reference picture information, such as POC distance between reference pictures and current picture.

11. It is proposed that the chroma residual scaling factor and/or the model computation of CCLM may depend on neighboring samples of a corresponding luma block which covers the top-left sample of the collocated luma block. In this invention, a "coding block" may refer to a video coding region such as CU/TU/PU as specified in HEVC specification or in the VVC working draft.

a) The "corresponding luma coding block" may be defined as the coding block which covers the top-left position of the collocated luma coding block.

i. FIG. 5 shows an example, where for an intra-coded chroma block in dual tree case, the CTU partition of chroma component may be different from the CTU partition of luma component. Firstly, a "corresponding luma coding block" covering the top-left sample of the collocated luma block of current chroma block is retrieved. Then by using the block size information of the "corresponding luma coding block", the top-left sample of the "corresponding luma coding block" can be derived, the top-left luma sample of the "corresponding luma coding block" covering the top-left sample of the collocated luma block is located at (x=32,y=16).

b) In one example, the block size/partition/location/coordination may be required to derive the location of "corresponding luma coding block" that covers the top-left sample of the collocated luma coding block.

i. In one example, the block size, and/or block partition, and/or block coordination may be stored for each block of a specific color component, such as the luma component.

ii. In one example, "corresponding luma coding block" and current block may be always inside the same CTU or CTU row, thus there may be no storage of block size/partition/position/ coordination in the line buffer.

c) In one example, the reconstructed samples not in the "corresponding luma coding block" may be used to derive the chroma residual scaling factor and/or model computation of CCLM.

i. In one example, the reconstructed samples adjacent to the "corresponding luma coding block" may be used to derive the chroma residual scaling factor and/or model computation of CCLM.

1) In one example, N samples located at the left neighboring columns and/or the above neighboring rows of the "corresponding luma coding block" may be used to derive

the chroma residual scaling factor and /or the model computation of CCLM, where N=1…2W+2H, W and H are the width and height of the "corresponding luma coding block".

a) Suppose the top-left sample of the "corresponding luma coding block" is (xCb, yCb), then in one example, the above neighboring luma sample may locate at (xCb + W/2, yCb - 1), or (xCb -1, yCb - 1). In an alternative example, the left neighboring luma sample may locate at (xCb+W-1, yCb-1).

b) In one example, the location(s) of the neighboring sample(s) may be fixed, and/or in a pre-defined checking order.

2) In one example, 1 out of N neighboring samples may be selected to derive the chroma residual scaling factor and /or the model computation of CCLM. Assume N=3, and the checking order of the three neighbor samples (xCb -1, yCb - H -1), (xCb + W/2, yCb - 1), (xCb -1, yCb - 1), then the first available neighboring sample in the checking list may be selected to derive the chroma residual scaling factor.

3) In one example, the median or average value of N samples located at the left neighboring columns and/or the above neighboring rows of the "corresponding luma coding block" may be used to derive the chroma residual scaling factor and /or the model computation of CCLM, where N=1…2W+2H, W and H are the width and height of the "corresponding luma coding block".

d) In one example, whether to perform the chroma residual scaling may depend on the "available" neighbouring samples of a corresponding luma block.

i. In one example, the "availability" of neighbouring samples may depend on the encoding mode of the current block/sub-block or/and encoding mode of the neighbouring sample.

1) In one example, for a block coded in inter mode, neighbouring samples coded in intra mode or/and IBC mode or/and CIIP mode or/and LIC mode may be considered as "unavailable".

2) In one example, for a block coded in inter mode, neighbouring samples employs diffusion filter or/and bilateral filter or/and Hadamard transform filter may be considered as "unavailable".

ii. In one example, the "availability" of the neighbouring samples may depend on the width and/or height of the current picture/tile/tile group/VPDU/slice.

1) In one example, if the neighbouring block locates outside the current picture, then it is treated as "unavailable".

iii. In one example, when there is no "available" neighbouring sample, chroma residual scaling may be disallowed.

iv. In one example, when the number of "available" neighbouring samples is smaller than K (K >= 1), chroma residual scaling may be disallowed.

v. Alternatively, the unavailable neighbouring sample may be filled by a default fixed value, or padding, or substitution, so that the chroma residual scaling may always be applied.

1) In one example, if the neighbouring sample is not available, then it may be filled by 1 << (bitDepth − 1), where bitDepth specifies the bit depth of the samples of the luma/chroma components.

2) Alternatively, if the neighbouring sample is not available, then it may be filled by padding from the surrounding samples located in the left/right/top/bottom neighbour.

3) Alternatively, if the neighbouring sample is not available, then it may be substituted by the first available adjacent sample at a pre-defined checking order.

4) Alternatively, if the neighbouring sample is not available, then it may be filled by a predefined filtered/mapped value (e.g., filtered/mapped value of 1 << (bitDepth − 1), where bitDepth specifies the bit depth of the samples of the luma/chroma components).

a) In one example, the filtering/mapping process may be LUT indexing of the forward mapping of LMCS.

e) In one example, whether and how to perform the chroma residual scaling may depend on the coding mode of current block and/or the coding modes of neighbour blocks.

i. The "current block" may refer to the current chroma block, or it may refer to the collocated luma block, or the corresponding luma block which covers at least one sample of the collocated chroma block. The "neighbour blocks" (adjacent or non-adjacent) may refer to chroma blocks neighbouring to the current chroma block, or they may refer to luma blocks neighbouring to the current luma block.

ii. In one example, the coding mode of one luma neighbouring block may be utilized which covers a given position, such as (-1, -1) relatively to the top-left coordinate of current block.

iii. In one example, the coding modes of multiple neighbouring blocks may be utilized which cover multiple positions, such as (x, -1) (e.g., with x being 0.. block's width minus 1) relatively to the top-left coordinate of current block, and/or (-1, y) (e.g., with y being -1... block's height minus 1) relatively to the top-left coordinate of current block

iv. In one example, if the reconstruction of one neighbouring block requires to access samples in the current slice/tile group, such as it is X-coded, then chroma residual scaling is disabled.

1) For example, mode X may be intra mode;

2) For example, mode X may be CIIP mode;

3) For example, mode X may be IBC mode;

4) In one example, if current block is inter-coded and not CIIP-coded, and the neighbour block neighbouring the corresponding luma block is coded with mode X, then chroma residual scaling is disabled.

v. In one example, if the reconstruction of one neighbouring block requires to access samples in the current slice/tile group, such as it is X-coded, then a default value may be used to derive chroma residual scaling factor.

1) For example, mode X may be intra mode;

2) For example, mode X may be CIIP mode;

3) For example, mode X may be IBC mode;

4) In one example, if current block is inter-coded and not CIIP-coded, and the neighbour block of the corresponding luma block is coded with mode X, then a default value may be used to derive chroma residual scaling factor.

5) In one example, the default value may depend on the bit depth of the luma/chroma samples.

6) In one example, the default value may be set to a filtered/mapped value of 1 << (bitDepth − 1), where bitDepth specifies the bit depth of the samples of the luma/chroma components. In one example, the filtering/mapping process may be a LUT indexing of the forward mapping of LMCS.

f) In one example, the filtered/mapped reconstructed samples neighboring the "corresponding luma coding block" may be used to derive the chroma residual scaling factor and /or the model computation of CCLM.

i. In one example, the filtering/mapping process may include reference smoothing filtering for intra blocks, post-filtering such as bilateral filter, Hadamard transform based filter, forward mapping of reshaper domain and so on.

12. It is proposed that a fixed value may be used to derive the chroma residual scaling factor for numbers of chroma blocks (such as CUs or TUs) in the current slice/tile group.

a) In one example, the chroma residual scaling factor for N chroma blocks may be derived by a fix value, wherein N being 1…total number of chroma blocks in the current slice/tile group.

b) In one example, a fixed value may be used to find the index of the piecewise linear model to which the value belongs to, and the chroma residual scaling factor may be then calculated from the derived piecewise index. In one example, the fixed value may depend on the internal bit depth for luma samples.

c) In one example, a fixed value may be directly used to represent the chroma residual scaling factor.

Restriction on whether chroma residual scaling and/or CCLM is applied or not

13. It is proposed that whether the chroma residual scaling or CCLM is applied or not may depend on the partition of the corresponding and/or the collocated luma block.

a) In one example, whether to enable or disable tools with cross-component information may depend on the number of CU/PU/TUs within the collocated luma (e.g., Y or G component) block.

i. In one example, if the number of CU/PU/TUs within the collocated luma (e.g., Y or G component) block exceeds a number threshold, such tools may be disabled.

ii. Alternatively, whether to enable or disable tools with cross-component information may depend on the partition tree depth.

1) In one example, if the maximum (or minimum or average or other variation) quadtree depth of CUs within the collocated luma block exceeds a threshold, such tools may be disabled.

2) In one example, if the maximum (or minimum or average or other variation) BT and/or TT depth of CUs within the collocated luma block exceeds a threshold, such tools may be disabled.

iii. Alternatively, furthermore, whether to enable or disable tools with cross-component information may depend on the block dimension of the chroma block.

iv. Alternatively, furthermore, whether to enable or disable tools with cross-component information may depend on whether the collocated luma cross multiple VPDUs/pre-defined region sizes.

v. The thresholds in the above discussion may be fixed numbers, or may be signaled, or may be dependent on standard profiles/levels/tiers.

b) In one example, if the collocated luma block of current chroma block is divided by multiple partitions (e.g., in FIG. 7), then the chroma residual scaling and/or CCLM may be prohibited.

   i. Alternatively, if the collocated luma block of current chroma block is not split (e.g., within one CU/TU/PU), then the chroma residual scaling and/or CCLM may be applied.

c) In one example, if the collocated luma block of current chroma block contains more than M CUs/PUs/TUs, then the chroma residual scaling and/or CCLM may be prohibited.

   i. In one example, M may be an integer great than 1.

   ii. In one example, M may be dependent on whether it is CCLM or chroma residual scaling process.

   iii. M may be fixed numbers, or may be signaled, or may be dependent on standard profiles/levels/tiers

d) The above-mentioned CUs within the collocated luma block may be interpreted to be all CUs within the collocated luma block. Alternatively, CUs within the collocated luma block may be interpreted to be partial CUs within the collocated luma block, such as CUs along the boundary of the collocated luma block.

e) The above-mentioned CUs within the collocated luma block may be interpreted to be sub-CUs or sub-blocks.

   i. For example, sub-CUs or sub-blocks may be used in ATMVP;

   ii. For example, sub-CUs or sub-blocks may be used in affine prediction;

   iii. For example, sub-CUs or sub-blocks may be used in Intra Sub-Partitions (ISP) mode.

f) In one example, if the CU/PU/TU covering the top-left luma sample of the collocated luma block is larger than a pre-defined luma block size, then the chroma residual scaling and/or CCLM may be prohibited.

   i. An example is depicted in FIG. 8, the collocated luma block is 32x32 but it is within a corresponding luma block with size equal to 64x64, then if the pre-defined luma block size is 32x64, the chroma residual scaling and/or CCLM is prohibited in this case

   ii. Alternatively, if the collocated of current chroma block is not split, and the corresponding luma block covering the top-left luma sample of the collocated luma blocks is completely included within a pre-defined bounding box, then the chroma residual scaling and/or CCLM for current chroma block may be applied. The bounding box may be defined as a rectangle with width W and height H, denoted by WxH, as shown in FIG. 9, where the

corresponding luma block is with width 32 and height 64, and the bounding box is with width 40 and height 70.

1) In one example, the size WxH of the bounding box may be defined according to the CTU width and/or height, or according to the CU width and/or height, or according to arbitrary values.

g) In one example, if the collocated luma block of current chroma block is divided by multiple partitions, then only the prediction samples (or reconstructed samples) inside the pre-defined partition of the collocated luma block are used to derive the chroma residual scaling factor in LMCS mode.

i. In one example, the average of all the prediction samples (or reconstructed samples) in the first partition of the collocated luma block are used to derive the chroma residual scaling factor in the LMCS mode.

ii. Alternatively, the top-left prediction sample (or reconstructed sample) in the first partition of the collocated luma block is used to derive the chroma residual scaling factor in the LMCS mode.

iii. Alternatively, the center prediction sample (or reconstructed sample) in the first partition of the collocated luma block is used to derive the chroma residual scaling factor in the LMCS mode.

h) It is proposed that whether and how to apply the cross-component tools such as CCLM and LMCS may depend on the coding mode(s) of one or multiple luma CUs which cover at least one sample of the collocated luma block.

i. For example, the cross-component tools are disabled if one or multiple luma CUs which cover at least one sample of the collocated luma block are coded with affine mode;

ii. For example, the cross-component tools are disabled if one or multiple luma CUs which cover at least one sample of the collocated luma block are coded with bi-prediction;

iii. For example, the cross-component tools are disabled if one or multiple luma CUs which cover at least one sample of the collocated luma block are coded with BDOF;

iv. For example, the cross-component tools are disabled if one or multiple luma CUs which cover at least one sample of the collocated luma block are coded with DMVR;

v. For example, the cross-component tools are disabled if one or multiple luma CUs which cover at least one sample of the collocated luma block are coded with matrix affine prediction mode as proposed in JVET-N0217;

vi. For example, the cross-component tools are disabled if one or multiple luma CUs which cover at least one sample of the collocated luma block are coded with inter mode;

vii. For example, the cross-component tools are disabled if one or multiple luma CUs which cover at least one sample of the collocated luma block are coded with ISP mode;

viii. In one example, "one or multiple luma CUs which cover at least one sample of the collocated luma block" may refer the corresponding luma block.

i) When CCLM/LMCS is prohibited, signalling of the indication of usage of CCLM/LMCS may be skipped.

j) In this disclosure, CCLM may refer to any variants modes of CCLM, including LM mode, LM-T mode, and LM-L mode.

14. It is proposed that whether and how to apply the cross-component tools such as CCLM and LMCS may be performed on part of a chroma block.

a) In one example, whether and how to apply the cross-component tools such as CCLM and LMCS at chroma subblock level.

i. In one example, a chroma subblock is defined as a 2x2 or 4x4 block in a chroma CU.

ii. In one example, for a chroma subblock, when the corresponding luma coding block of the current chroma CU covers all samples of the corresponding block of the subblock, CCLM may be applied.

iii. In one example, for a chroma subblock, when not all samples of the corresponding block are covered by the corresponding luma coding block of the current chroma CU, CCLM is not applied.

iv. In one example, the parameters of CCLM or LMCS are derived for each chroma subblock as treating the subblock as a chroma CU.

v. In one example, when CCLM or LMCS are applied for a chroma subblock, the collocated block's samples may be used.

Applicability of chroma residual scaling in LMCS mode

15. It is proposed that whether luma dependent chroma residual scaling can be applied may be signalled at other syntax level in addition to the tile group header as specified in JVET-M1001.

a) For example, a **chroma_residual_scale_flag** may be signalled at sequence level (e.g. in SPS), at picture level (e.g. in PPS or picture header), at slice level (e.g. in slice header), at tile level, at CTU row level, at CTU level, at CU level. **chroma_residual_scale_flag** equal to 1 specifies

that chroma residual scaling is enabled for the CUs below the signalled syntax level. **chroma_residual_scale_flag** equal to 0 specifies that chroma residual scaling is not enabled for below the signalled syntax level. When chroma_residual_scale_flag is not present, it is inferred to be equal to 0.

b) In one example, if chroma residual scaling is constrained at a partition node level. Then **chroma_residual_scale_flag** may not be signalled and inferred to be 0 for CUs covered by the partition node. In one example, a partition node may be a CTU (CTU is treated as the root node of quaternary tree partition).

c) In one example, if chroma residual scaling is constrained for chroma block size equal or smaller than 32x32, then **chroma_residual_scale_flag** may not be signalled and inferred to be 0 for chroma block size equal or smaller than 32x32.

Applicability of CCLM mode

16. It is proposed that whether CCLM mode can be applied may be signalled at other syntax levels in addition to the sps level as specified in JVET-M1001.

a) For example, it may be signalled at picture level (e.g. in PPS or picture header), at slice level (e.g. in slice header), at tile group level (e.g. in tile group header), at tile level, at CTU row level, at CTU level, at CU level.

b) In one example, **cclm_flag** may not be signalled and inferred to be 0 if CCLM cannot be applied.

i. In one example, if chroma residual scaling is constrained for chroma block size equal or smaller than 8x8, then **cclm_flag** may not be signalled and inferred to be 0 for chroma block size equal or smaller than 8x8.

Unification of chroma residual scaling factor derivation for intra mode and inter mode

17. Chroma residual scaling factor may be derived after encoding/decoding a luma block and may be stored and used for following coded blocks.

a) In one example, certain prediction samples or/and intermediate prediction samples or/and reconstructed samples or/and reconstructed samples before loop filtering (e.g., before processed by deblocking filter or/and SAO filter or/and bilateral filter or/and Hadamard transform filter or/and ALF filter) in the luma block may be used for derivation of the chroma residual scaling factor.

i. For example, partial samples in the bottom row or/and right column of the luma block may be used for derivation of the chroma residual scaling factor.

b) In single tree case, when encoding a block coded in intra mode or/and IBC mode or/and inter mode, derived chroma residual scaling factor of neighboring blocks may be used for deriving scaling factor of the current block.

   i. In one example, certain neighboring blocks may be checked in order, and the first available chroma residual scaling factor may be used for the current block.

   ii. In one example, certain neighboring blocks may be checked in order, and a scaling factor may be derived based on the first K available neighboring chroma residual scaling factors.

   iii. In one example, for a block coded in inter mode or/and CIIP mode, if a neighboring block is coded in intra mode or/and IBC mode or/and CIIP mode, chroma residual scaling factor of the neighboring block may be considered as "unavailable".

   iv. In one example, neighboring blocks may be checked in order of left (or above left) -> above (or above right).

      1) Alternatively, neighboring blocks may be checked in order of above (or above right) -> left (or above left).

c) In separate tree case, when encoding a chroma block, the corresponding luma block may be first identified. Then, derived chroma residual scaling factor of its (e.g., the corresponding luma block) neighboring blocks may be used for deriving scaling factor of the current block.

   i. In one example, certain neighboring blocks may be checked in order, and the first available chroma residual scaling factor may be used for the current block.

   ii. In one example, certain neighboring blocks may be checked in order, and a scaling factor may be derived based on the first K available neighboring chroma residual scaling factors.

d) Neighboring blocks may be checked in a predefined order.

   i. In one example, neighboring blocks may be checked in order of left (or above left) -> above (or above right)

   ii. In one example, neighboring blocks may be checked in order of above (or above right) -> left (or above left).

   iii. In one example, neighboring blocks may be checked in order of below left -> left -> above right -> above -> above left.

   iv. In one example, neighboring blocks may be checked in order of left -> above -> above right -> below left -> above left.

e) In one example, whether to apply chroma residual scaling may depend on the "availability" of neighbouring block.

i. In one example, when there is no "available" neighbouring block, chroma residual scaling may be disallowed.

ii. In one example, when the number of "available" neighbouring blocks is smaller than K (K >= 1), chroma residual scaling may be disallowed.

iii. Alternatively, when there is no "available" neighbouring block, chroma residual scaling factor may be derived by a default value.

1) In one example, a default value 1<< (BitDepth -1) may be used to derive the chroma residual scaling factor.

f) In one example, the chroma residual scaling factor of current chroma block may be stored and used for following coded blocks.

g) In one example, the storage of chroma residual scaling factors may be removed from line buffer.

i. In one example, when the current block and a neighboring (adjacent or non-adjacent) block to be accessed are in different regions, its chroma residual scaling factor may be considered as "unavailable" and may not be used for the derivation of chroma residual scaling factor of the current block.

1) A region may be a slice, a tile, a tile group, a CTU row, or a CTU.

2) Alternatively, its chroma residual scaling factor may be considered as a default value in such a case.

3) Alternatively, chroma residual scaling cannot be applied in such a case.

h) In one example, the chroma residual scaling factor of current chroma block may be updated on-the-fly and may be saved in a history table for scaling factor derivation of the following blocks.

i. The history table may be updated in a FIFO (first-in first-out) way.

ii. After decoding/encoding a chroma block, a chroma residual scaling factor may be derived (e.g., according to the luma values) and may be stored in the FIFO history table.

iii. In one example, the FIFO history table may contain at most 1 entry. In this case, derived chroma residual scaling factor of the latest decoded block is used for the current block.

iv. In one example, the history table is refreshed before encoding/decoding a picture, and/or a slice, and/or a tile group, and/or a tile, and/or a CTU row and/or a CTU.

1) In one example, a default chroma residual scaling factor may be put into the history table when the history table is refreshed.

2) In one example, the history table is set to empty when the FIFO history table is refreshed.

5. Embodiments

5.1 Embodiment #1

The embodiment below is for the method in item 11 of the example embodiments in Section 4 of this document.

Newly added parts are highlighted in bolded, underlined, italicized font, and the deleted parts from VVC working draft are highlighted in capitalized font. The modifications are based on the latest VVC working draft (JVET-M1007-v7) and the new adoption in JVET-N220-v3.

8.7.5.4 Picture reconstruction with luma dependent chroma residual scaling process for chroma samples

Inputs to this process are:

- a location (xCurr, yCurr) of the top-left sample of the current transform block relative to the top-left sample of the current picture,
- a variable nCurrSw specifying the transform block width,
- a variable nCurrSh specifying the transform block height,
- an (nCurrSw)x(nCurrSh) array predSamples specifying the chroma prediction samples of the current block,
- an (nCurrSw)x(nCurrSh) array resSamples specifying the chroma residual samples of the current block.

Output of this process is a reconstructed chroma picture sample array recSamples.

The current luma location (xTbY, yTbY) is set to (xCurr << 1, yCurr << 1)

The corresponding luma location (xTbCorrY, yTbCorrY) is derived as follows:

- ***If tile_group_type is equal to 2 (I tile group) and qtbtt_dual_tree_intra_flag is equal to 1, the corresponding luma location (xTbCorrY, yTbCorrY) is set to the top-left sample location of the corresponding CU that covers the top-left sample of the current luma location (xTbY, yTbY).***
- ***Otherwise, the corresponding luma location (xTbCorrY, yTbCorrY) is set to the current luma location (xTbY, yTbY)***

The variable availL is derived as follows:

- *The availability of left neighbouring sample derivation process for a block as specified in clause 6.4.X [Ed. (BB): Neighbouring blocks availability checking process tbd] is invoked with the corresponding luma location (xTbCorrY, yTbCorrY) and the neighbouring luma location (xTbCorrY − 1, yTbCorrY) as inputs, and the output is assigned to availL.*

When availL is equal to TRUE, the neighbouring top-left luma sample pY[x][y] with x = −1, y = 0, is set equal to the reconstructed luma samples prior to the deblocking filter process at the location (xTbCorrY + x, yTbCorrY + y). The mapped value of the neighbouring top-left luma sample pYMap [x][y] with x = −1, y = −1, is derived as follows:

idxY = pY[x][y] >> Log2(OrgCW)

pYMap[x][y]=Clip1$_Y$(LmcsPivot[idxY]
*+ (ScaleCoeff[idxY] * (pY[x][y] − InputPivot[idxY]) + (1 << 10)) >> 11)*

The reconstructed chroma picture sample recSamples is derived as follows for i = 0..nCurrSw − 1, j = 0..nCurrSh − 1:

- IF TILE_GROUP_CHROMA_RESIDUAL_SCALE_FLAG IS EQUAL TO 0 OR NCURRSW * NCURRSH IS LESS THAN OR EUQAL TO 4, THE FOLLOWING APPLIES:

  RECSAMPLES[XCURR + I][YCURR + J] = CLIP1$_C$ (PREDSAMPLES[I][J] + RESSAMPLES[I][J]) (8-1063)

- *If one of the following conditions is true, recSamples[xCurr + i][yCurr + j] is set equal to Clip1$_C$ (predSamples[i][j] + resSamples[i][j])*
  - *tile_group_chroma_residual_scale_flag is equal to 0*
  - *nCurrSw * nCurrSh is less than or equal to 4*
  - *availL is equal to false*
  - *CuPredMode[xCurr][yCurr] is equal to MODE_INTER and ciip_flag[xCurr][yCurr] is equal to 0, and CuPredMode[xTbCorrY − 1][yTbCorrY] is euqal to MODE_INTRA*
  - *CuPredMode[xCurr][yCurr] is equal to MODE_INTER and ciip_flag[xCurr][yCurr] is equal to 0, and CuPredMode[xTbCorrY − 1][yTbCorrY] is euqal to MODE_IBC*
  - *CuPredMode[xCurr][yCurr] is equal to MODE_INTER and ciip_flag[xCurr][yCurr] is equal to 0, and CuPredMode[xTbCorrY − 1][yTbCorrY] is euqal to MODE_INTER and ciip_flag[xCurr][yCurr] is equal to 1*

- Otherwise (tile_group_chroma_residual_scale_flag is equal to 1 and nCurrSw * nCurrSh is greater than 4), the following applies:

– For the derivation of the variable varScale the following ordered steps apply:

1. The variable invAvgLuma is derived as follows:

$$\text{INVAVGLUMA} = \text{CLIP1}_Y((\sum_{K=0}^{2*\text{NCURRSW}-1} \sum_{L=0}^{2*\text{NCURRSH}-1} \text{PREDMAPSAMPLES}[K][L] + \text{NCURRSW} * \text{NCURRSH} * 2) / (\text{NCURRSW} * \text{NCURRSH} * 4)) \quad (8\text{-}1064)$$

invAvgLuma = Clip1$_Y$(pYMap[-1][0])

The variable idxYInv is derived by invoking the identification of piece-wise function index as specified in clause 8.7.5.3.2 with invAvgLuma as the input and idxYInv as the output.

2. The variable varScale is derived as follows:

$$\text{varScale} = \text{ChromaScaleCoeff}[\text{idxYInv}] \quad (8\text{-}1065)$$

– The recSamples is derived as follows:

– If tu_cbf_cIdx [xCurr][yCurr] equal to 1, the following applies:

$$\text{resSamples}[i][j] = \text{Clip3}(-(1 << \text{BitDepth}_C), 1 << \text{BitDepth}_C - 1, \text{resSamples}[i][j])$$

$$\text{recSamples}[\text{xCurr} + i][\text{yCurr} + j] = \text{ClipCidx1}(\text{predSamples}[i][j] + \text{Sign}(\text{resSamples}[i][j]) * ((\text{Abs}(\text{resSamples}[i][j]) * \text{varScale} + (1 << 10)) >> 11)) \quad (8\text{-}1066)$$

– Otherwise (tu_cbf_cIdx[xCurr][yCurr] equal to 0), the following applies:

$$\text{recSamples}[\text{xCurr} + i][\text{yCurr} + j] = \text{ClipCidx1}(\text{predSamples}[i][j]) \quad (8\text{-}1067)$$

5.2 Embodiment #2

The embodiment below is for the method in item 11 of the example embodiments in Section 4 of this document.

Newly added parts are highlighted in bolded, underlined, italicized font, and the deleted parts from VVC working draft are highlighted in capitalized font. The modifications are based on the latest VVC working draft (JVET-M1007-v7) and the new adoption in JVET-N220-v3.

The differences between Embodiment #2 and #1 are listed as follows:

– *Multiple neighbor luma samples are checked to derive the chroma residual scaling factor.*

- *When the neighbor luma samples are not available, or when neighbor luma is coded in INTRA/CIIP/IBC mode while current is coded in INTER mode, #2 uses a default value for chroma residual scaling factor derivation.*

8.7.5.4 Picture reconstruction with luma dependent chroma residual scaling process for chroma samples

Inputs to this process are:

- a location (xCurr, yCurr) of the top-left sample of the current transform block relative to the top-left sample of the current picture,
- a variable nCurrSw specifying the transform block width,
- a variable nCurrSh specifying the transform block height,
- an (nCurrSw)x(nCurrSh) array predSamples specifying the chroma prediction samples of the current block,
- an (nCurrSw)x(nCurrSh) array resSamples specifying the chroma residual samples of the current block.

Output of this process is a reconstructed chroma picture sample array recSamples.

The current luma location (xTbY, yTbY) is set to (xCurr << 1, yCurr << 1)

The corresponding luma location (xTbCorrY, yTbCorrY) is derived as follows:

- ***If tile group type is equal to 2 (I tile group) and qtbtt dual tree intra flag is equal to 1, the corresponding luma location (xTbCorrY, yTbCorrY) is set to the top-left sample location of the corresponding CU that covers the top-left sample of the current luma location (xTbY, yTbY).***
- ***Otherwise, the corresponding luma location (xTbCorrY, yTbCorrY) is set to the current luma location (xTbY, yTbY)***

The variable availTL is derived as follows:

- ***The availability of top-left neighbouring sample derivation process for a block as specified in clause 6.4.X [Ed. (BB): Neighbouring blocks availability checking process tbd] is invoked with the corresponding luma location (xTbCorrY, yTbCorrY) and the neighbouring luma location (xTbCorrY − 1, yTbCorrY − 1) as inputs, and the output is assigned to availTL.***

The variable availT is derived as follows:

- ***The availability of top neighbouring sample derivation process for a block as specified in clause 6.4.X [Ed. (BB): Neighbouring blocks availability checking process tbd] is invoked with the***

corresponding luma location (xTbCorrY, yTbCorrY) and the neighbouring luma location (xTbCorrY, yTbCorrY − 1) as inputs, and the output is assigned to availT.

The numAvailNeigh is set equal to availTL + availT.

If numAvailNeigh is larger or equal to 1, the neighbouring luma samples pY[x][y] with x = −1, 0, y = −1, are set equal to the average value of the reconstructed luma samples prior to the deblocking filter process at the location (xTbCorrY + x, yTbCorrY + y). The average value of neighbouring luma samples pY_{Avg} is derived as follows:

$$pY_{Avg} = ((availTL ? pY[-1][-1] : 0) + (availT ? pY[0][-1] : 0)) >> (numAvailNeigh > 1 ? 1 : 0)$$

Otherwise, if one of the following conditions is true, pY_{Avg} is set equal to (1 << ($BitDepth_Y$ -1)).

- *availTL is equal to false*
- *CuPredMode[xCurr][yCurr] is equal to MODE_INTER and ciip_flag[xCurr][yCurr] is equal to 0, and CuPredMode[xTbCorrY − 1][yTbCorrY – 1] is euqal to MODE_INTRA*
- *CuPredMode[xCurr][yCurr] is equal to MODE_INTER and ciip_flag[xCurr][yCurr] is equal to 0, and CuPredMode[xTbCorrY − 1][yTbCorrY – 1] is euqal to MODE_IBC*
- *CuPredMode[xCurr][yCurr] is equal to MODE_INTER and ciip_flag[xCurr][yCurr] is equal to 0, and CuPredMode[xTbCorrY − 1][yTbCorrY – 1] is euqal to MODE_INTER and ciip_flag[xCurr][yCurr] is equal to 1*

The mapped value of the neighbouring top-left luma sample $pYMap_{Avg}$, is derived as follows:

$$idxY = pY_{Avg} >> Log2(OrgCW)$$

$$pYMap_{Avg} = Clip1_Y(LmcsPivot[idxY] + (ScaleCoeff[idxY] * (pY_{Avg} − InputPivot[idxY]) + (1 << 10)) >> 11)$$

The reconstructed chroma picture sample recSamples is derived as follows for i = 0..nCurrSw − 1, j = 0..nCurrSh − 1:

- If tile_group_chroma_residual_scale_flag is equal to 0 or nCurrSw * nCurrSh is less than or euqal to 4, the following applies:

$$recSamples[xCurr + i][yCurr + j] = Clip1_C (predSamples[i][j] + resSamples[i][j]) \quad (8\text{-}1063)$$

- Otherwise (tile_group_chroma_residual_scale_flag is equal to 1 and nCurrSw * nCurrSh is greater than 4), the following applies:
  - For the derivation of the variable varScale the following ordered steps apply:
    - The variable invAvgLuma is derived as follows:

$$INVAVGLUMA = CLIP1_Y((\sum_{K=0}^{2*NCURRSW-1} \sum_{L=0}^{2*NCURRSH-1} PREDMAPSAMPLES[K][L] + NCURRSW * NCURRSH *2) / (NCURRSW * NCURRSH *4)) \quad (8\text{-}1064)$$

invAvgLuma = Clip1$_Y$(pYMap$_{Avg}$)

The variable idxYInv is derived by invoking the identification of piece-wise function index as specified in clause 8.7.5.3.2 with invAvgLuma as the input and idxYInv as the output.

The variable varScale is derived as follows:

$$varScale = ChromaScaleCoeff[idxYInv] \quad (8\text{-}1065)$$

- The recSamples is derived as follows:
  - If tu_cbf_cIdx [xCurr][yCurr] equal to 1, the following applies:

$$resSamples[i][j] = Clip3(-(1 << BitDepth_C), 1 << BitDepth_C - 1, resSamples[i][j])$$

$$recSamples[xCurr + i][yCurr + j] = ClipCidx1(predSamples[i][j] + Sign(resSamples[i][j]) * ((Abs(resSamples[i][j]) * varScale + (1 << 10)) >> 11)) \quad (8\text{-}1066)$$

  - Otherwise (tu_cbf_cIdx[xCurr][yCurr] equal to 0), the following applies:

$$recSamples[xCurr + i][yCurr + j] = ClipCidx1(predSamples[i][j]) \quad (8\text{-}1067)$$

6. Example implementations of the disclosed technology

[0076] FIG. 10 is a block diagram of a video processing apparatus 1000. The apparatus 1000 may be used to implement one or more of the methods described herein. The apparatus 1000 may be embodied in a smartphone, tablet, computer, Internet of Things (IoT) receiver, and so on. The apparatus 1000 may include one or more processors 1002, one or more memories 1004 and video processing hardware 1006. The processor(s) 1002 may be configured to implement one or more methods (including, but not limited to, methods 800 and 900) described in the present document. The memory (memories) 1004 may be used for storing data and code used for implementing the methods and techniques described herein. The video processing hardware 1006 may be used to implement, in hardware circuitry, some techniques described in the present document.

[0077] In some embodiments, the video coding methods may be implemented using an apparatus that is implemented on a hardware platform as described with respect to FIG. 10.

[0078] FIG. 11 shows a flowchart of an example method 1100 for linear model derivations for cross-component prediction in accordance with the disclosed technology. The method 1100 includes, at step 1110, performing a conversion between a current video block and a bitstream representation of the current video block, wherein, during the conversion, a second set of color component values of the current video block are derived from a first set of color component values included in one or more reference frames, wherein the first set of color component values are usable in a linear model of a video coding step.

[0079] Some embodiments may be described using the following clause-based format.

[0080] 1. A method for video processing, comprising:

[0081] performing a conversion between a current video block and a bitstream representation of the current video block, wherein, during the conversion, a second set of color component values of the current video block are derived from a first set of color component values included in one or more reference frames, wherein the first set of color component values are usable in a linear model of a video coding step.

[0082] 2. The method of clause 1, wherein the first set of color component values are interpolated prior to use in the linear model of the video coding step.

[0083] 3. The method of any one or more of clauses 1-2, wherein a linear combination of the first set of color component values are usable as parameters in the linear model.

[0084] 4. The method of clause 1, wherein locations of the first set of color component values included in the one or more reference frames are selected based, at least in part, on motion information of the current video block.

[0085] 5. The method of clause 4, wherein a position of a luma component value in the one or more reference frames is calculated from a position of a corresponding luma component value in the current video block and the motion information of the current video block.

[0086] 6. The method of clause 5, wherein the position of the corresponding luma component value is a top-left sample, a center sample, or a bottom-right sample in the current video block.

[0087] 7. The method of clause 6, wherein the motion information of the current video block corresponds to an integer motion vector or a fractional motion vector.

[0088] 8. The method of clause 7, wherein the fractional motion vector is derived using a fractional luma component value in the one or more reference frames.

[0089] 9. The method of clause 7, wherein the integer motion vector is derived by rounding towards zero or away from zero.

[0090] 10. The method of clause 1, wherein locations of the first set of color component values included in the one or more reference frames are pre-defined positions.

[0091] 11. The method of any one or more of clauses 1-10, wherein a median or an average of the first set of color component values are used to derive the second set of color component values of the current video block.

[0092] 12. The method of any one or more of clauses 1-11, wherein the one or more reference frames are pre-defined reference frames.

[0093] 13. The method of clause 12, wherein the pre-defined reference frames include a frame with a reference index of a reference picture list.

[0094] 14. The method of clause 13, wherein the reference index is zero and the reference picture list is zero.

[0095] 15. The method of clause 13, wherein the reference index and/or the reference picture list is signaled in the bitstream representation associated with one or more of the following: a sequence, a picture, a tile, a group, a slice, a tile, a coding tree unit row, or a video block.

[0096] 16. The method of clause 1, wherein the second set of color component values of the current video block are derived from a mathematical mean or a weighted average of the first set of color component values included in the one or more reference frames.

[0097] 17. The method of clause 1, wherein the second set of color component values of the current video block are selectively derived from the first set of color component values included in the one or more reference frames, based on whether the current video block is a bi-prediction coded block.

[0098] 18. The method of clause 17, wherein the second set of color component values of the current video block are individually derived for each prediction direction of the first set of color component values.

[0099] 19. The method of clause 1, wherein the second set of color component values of the current video block are selectively derived from the first set of color component values included in the one or more reference frames, based on whether the current video block is associated with sub-block-based prediction.

[0100] 20. The method of clause 1, wherein the sub-block-based prediction corresponds to affine prediction or alternative temporal motion vector prediction (ATMVP).

[0101] 21. The method of any one or more of clauses 19-20, wherein the second set of color component values of the current video block are derived for individual sub-blocks.

[0102] 22. The method of any one or more of clauses 19-21, wherein the second set of color component values of the current video block are derived for an entirety of the current video block regardless of the sub-block-based prediction.

[0103] 23. The method of any one or more of clauses 19-22, wherein the first set of color component values included in one or more reference frames are selected based, at least in part on, a motion vector of a sub-block of the current video block.

[0104] 24. The method of any one or more of clauses 1-23, wherein the first set of color component values included in one or more reference frames are intermediate color component values.

[0105] 25. The method of any one or more of clauses 1-24, wherein the video coding step precedes another video coding step.

[0106] 26. The method of clause 25, wherein the first set of color component values included in the one or more reference frames are selected based, at least in part on, an intermediate motion vector of the current video block or a sub-block of the current video block, and wherein the intermediate motion vector is calculated prior to the another video coding step.

[0107] 27. The method of any one or more of clauses 24-26, wherein the another video coding step includes one or a combination of the following steps: a Bi-Directional Optical Flow (BDOF) step, a decoder-side motion vector refinement (DMVR) step, a prediction refinement optical flow (PROF) step.

[0108] 28. The method of any one or more of clauses 1-27, wherein the first set of color component values included in the one or more reference frames correspond to MxN luma component values associated with a corresponding luma block.

[0109] 29. The method of clause 28, wherein the corresponding luma block is a collocated luma block of the current video block.

[0110] 30. The method of clause 29, wherein a product of M and N is smaller than a product of a block width and a block height of the collocated luma block of the current video block.

[0111] 31. The method of any one or more of clauses 27-30, wherein the first set of color component values included in the one or more reference frames correspond to at least a portion of reference samples identified at positions of neighboring luma samples of the collocated luma block.

[0112] 32. The method of any one or more of clauses 1-31, wherein the first set of color component values are down sampled prior to use in the linear model of the video coding step.

[0113] 33. The method of clause 1, wherein the second set of color component values of the current video block are selected, based, at least in part on, one or more of the following information of the current video block: a quantization parameter, a coding mode, or a picture order count (POC).

[0114] 34. The method of clause 31, wherein the positions of the neighboring luma samples are such that a top left sample of the collocated luma block is covered.

[0115] 35. The method of clause 28, wherein the first set of color component values included in the one or more reference frames correspond to at least a portion of reference samples identified at positions external to the corresponding luma block.

[0116] 36. The method of clause 28, wherein the second set of color component values of the current video block are selectively derived from the first set of color component values included in the one or more reference frames, based on availability of neighboring samples of the corresponding luma block.

[0117] 37. The method of clause 28, wherein the availability of the neighboring samples of the corresponding luma block is based on one or more of: a use of a coding mode of the current video block, a use of a coding mode of the neighboring samples of the corresponding luma block, a use of a coding mode of the corresponding luma block, a use of a coding mode of one or more neighboring video blocks, a use of a type of a filter associated with the neighboring samples of the corresponding luma block, or a location of the neighboring samples of the corresponding luma block relative to the current video blocks or sub-blocks thereof.

[0118] 38. The method of clause 28, further comprising:

[0119] in response to a lack of the availability of the neighboring samples of the corresponding luma block, substituting, filling, or padding unavailable samples with other samples.

[0120] 39. The method of clause 28, further comprising:

[0121] applying a smoothing filter to samples neighboring the corresponding luma block.

[0122] 40. A method for video processing, comprising:

[0123] performing a conversion between a current video block and a bitstream representation of the current video block, wherein, during the conversion, a second set of color component values of the current video block are derived from a first set of color component values included in one or more reference frames, wherein the first set of color component values are usable in a linear model of a video coding step; and

[0124] in response to determining that the first set of color component values included in the one or more reference frames is a collocated luma block of the current video block, selectively enabling or disabling derivation of the second set of color component values of the current video block, based on one or more conditions associated with the collocated luma block of the current video block.

[0125] 41. The method of clause 40, wherein the one or more conditions associated with the collocated luma block of the current video block include: a partition size of the collocated luma block, a number of coding units of the collocated luma block achieving a threshold number, a top-left luma sample

of the collocated luma block achieving a threshold size, a partition tree depth of the collocated luma block, a corresponding luma block covering the top-left luma sample of the collocated luma block, a dimension of the collocated luma block or the current video block, or a corresponding luma block covering the top-left luma sample of the collocated luma block and additionally included within a bounding box of pre-defined size.

[0126] 42. The method of clause 40, wherein information indicating the selectively enabling or disabling the derivation is included in the bitstream representation.

[0127] 43. The method of clause 28, wherein the availability of neighboring samples of the corresponding luma block is associated with checking for the neighboring samples according to a pre-defined order.

[0128] 44. The method of clause 41, wherein the collocated luma block and the current video block are associated with a same coding tree unit or a same row of a coding tree unit.

[0129] 45. A method for video processing, comprising:

[0130] performing a conversion between a current video block and a bitstream representation of the current video block, wherein, during the conversion, a second set of color component values of the current video block are derived from a first set of color component values included in one or more reference frames, wherein the first set of color component values are usable in a linear model of a video coding step; and

[0131] in response to determining that one or more properties of the current video block or neighboring video blocks of the current video block are satisfied, selectively enabling or disabling derivation of the second set of color component values of the current video block.

[0132] 46. The method of clause 45, wherein the one or more properties of the current video block or neighboring video blocks of the current video block correspond to a neighboring luma block covering a spatial position relative to a spatial position of the current video block.

[0133] 47. The method of clause 45, wherein the one or more properties of the current video block or neighboring video blocks of the current video block correspond to spatial positions of the neighboring video blocks of the current video block relative to a spatial position of the current video block

[0134] 48. The method of clause 45, further comprising:

[0135] in response to determining that reconstructions of the neighboring video blocks is based, at least in part, on a coding mode of the current video block, disabling derivation of the second set of color component values of the current video block.

[0136] 49. The method of clause 45, further comprising:

[0137] in response to determining that the current video block is an inter-coded block and not a combined inter and intra prediction-coded block, and a block neighboring a corresponding luma block of the current video block is an intra-coded block, a combined inter and intra prediction (CIIP) block, or an intra block copy (IBC) coded block, disabling derivation of the second set of color component values of the current video block.

[0138] 50. The method of clause 45, further comprising:

[0139] in response to determining that the current video block is an inter-coded block and not a combined inter and intra prediction-coded block, and a block neighboring a corresponding luma block of the current video block is an intra-coded block, a combined inter and intra prediction (CIIP) block, or an intra block copy (IBC) coded block, enabling derivation of the second set of color component values of the current video block, wherein the first set of color component values are fixed values.

[0140] 51. The method of clause 45, wherein the first set of color component values are fixed values.

[0141] 52. The method of clause 51, wherein the fixed values correspond to a piecewise index of the linear model of the video coding step.

[0142] 53. The method of any one or more of clauses 1-52, wherein the neighboring samples may be adjacent or non-adjacent to the current video block.

[0143] 54. The method of any one or more of clauses 1-52, wherein the neighboring samples may be associated with chroma blocks neighboring the current video block or chroma blocks neighboring the current video block.

[0144] 55. The method of any one or more of clauses 1-54, wherein the current video block corresponds to a chroma block, the collocated luma block, or the corresponding luma block covering the top-left luma sample of the collocated chroma block.

[0145] 56. The method of any one or more of clause 1-54, wherein the second set of color component values of the current video block are stored for use in connection with one or more other video blocks.

[0146] 57. The method of any one or more of clauses 1-56, wherein the linear model corresponds to a cross-component linear model (CCLM) and the video coding step corresponds to a luma mapping with chroma scaling (LMCS) mode.

[0147] 58. The method of any one or more of clauses 1-57, wherein the current video block is an inter-coded block, a bi-prediction coded block, a combined inter and intra prediction (CIIP) block, or an intra block copy (IBC) coded block.

[0148] 59. The method of any one or more of clauses 1-58, wherein the second set of color component values of the current video block are stored for use associated with other video blocks in the bitstream representation.

[0149] 60. The method of clause 59, wherein the second set of color component values of the current video block are stored in a line buffer for selective availability or unavailability by a neighboring video block included in the other video blocks, wherein the current video block and the neighboring video block are associated with different slices, tiles, tile groups, coding tree units, or rows of coding tree units.

[0150] 61. The method of clause 60, wherein the second set of color component values of the current video block are fixed values.

[0151] 62. The method of clause 60, wherein the second set of color component values of the current video block are prevented from derivation.

[0152] 63. The method of clause 59, wherein the derivation of the second set of color component values of the current video block is prevented.

[0153] 64. The method of clause 59, wherein the second set of color component values of the current video block are stored in a table for selective availability or unavailability by a neighboring video block included in the other video blocks.

[0154] 65. The method of clause 64, wherein the second set of color component values of the current video block are dynamically updated.

[0155] 66. The method of clause 65, wherein the second set of color component values of the current video block are same as a previous video block in the other video blocks.

[0156] 67. The method of clause 64, wherein the second set of color component values of the current video block are dynamically updated.in a first in first out (FIFO) manner.

[0157] 68. The method of any one or more of clauses 1-67, wherein the first set of color component values correspond to luma sample values and the second set of color component values correspond to chroma scaling factors.

[0158] 69. An apparatus in a video system comprising a processor and a non-transitory memory with instructions thereon, wherein the instructions upon execution by the processor, cause the processor to implement the method in any one of clauses 1 to 68.

[0159] 70. A computer program product stored on a non-transitory computer readable media, the computer program product including program code for carrying out the method in any one of clauses 1 to 68.

[0160] FIG. 12 is a block diagram showing an example video processing system 1200 in which various techniques disclosed herein may be implemented. Various implementations may include some or all of the components of the system 1200. The system 1200 may include input 1202 for receiving video content. The video content may be received in a raw or uncompressed format, e.g., 8 or 10 bit multi-component pixel values, or may be in a compressed or encoded format. The input 1202 may represent a network

interface, a peripheral bus interface, or a storage interface. Examples of network interface include wired interfaces such as Ethernet, passive optical network (PON), etc. and wireless interfaces such as Wi-Fi or cellular interfaces.

[0161] The system 1200 may include a coding component 1204 that may implement the various coding or encoding methods described in the present document. The coding component 1204 may reduce the average bitrate of video from the input 1202 to the output of the coding component 1204 to produce a coded representation of the video. The coding techniques are therefore sometimes called video compression or video transcoding techniques. The output of the coding component 1204 may be either stored, or transmitted via a communication connected, as represented by the component 1206. The stored or communicated bitstream (or coded) representation of the video received at the input 1202 may be used by the component 1208 for generating pixel values or displayable video that is sent to a display interface 1210. The process of generating user-viewable video from the bitstream representation is sometimes called video decompression. Furthermore, while certain video processing operations are referred to as "coding" operations or tools, it will be appreciated that the coding tools or operations are used at an encoder and corresponding decoding tools or operations that reverse the results of the coding will be performed by a decoder.

[0162] Examples of a peripheral bus interface or a display interface may include universal serial bus (USB) or high definition multimedia interface (HDMI) or Displayport, and so on. Examples of storage interfaces include SATA (serial advanced technology attachment), PCI, IDE interface, and the like. The techniques described in the present document may be embodied in various electronic devices such as mobile phones, laptops, smartphones or other devices that are capable of performing digital data processing and/or video display.

[0163] FIG. 13 shows a flowchart of an example method for visual media processing. Steps of this flowchart are discussed in connection with example 11b in Section 4 of this document. At step 1302, the process computes, during a conversion between a current video block of visual media data and a bitstream representation of the current video block, a cross-component linear model (CCLM) and/or a chroma residual scaling (CRS) factor for the current video block based, at least in part, on neighboring samples of a corresponding luma block which covers a top-left sample of a collocated luma block associated with the current video block, wherein one or more characteristics of the current video block are used for identifying the corresponding luma block.

[0164] FIG. 14 shows a flowchart of an example method for visual media processing. Steps of this flowchart are discussed in connection with example 11e in Section 4 of this document. At step 1402, the process uses a rule to make a determination of selectively enabling or disabling a chroma residual scaling (CRS) on color components of a current video block of visual media data, wherein the rule is based on coding mode information of the current video block and/or coding mode information of one or more

neighbouring video blocks. At step 1404, the process performs a conversion between the current video block and a bitstream representation, based on the determination.

[0165] FIG. 15 shows a flowchart of an example method for visual media processing. Steps of this flowchart are discussed in connection with example 12 in Section 4 of this document. At step 1502, the process uses a single chroma residual scaling factor for at least one chroma block associated with video blocks in a slice or a tile group associated with a current video block of visual media data. At step 1504, the process performs a conversion between the current video block and a bitstream representation of the current video block.

[0166] FIG. 16 shows a flowchart of an example method for visual media processing. Steps of this flowchart are discussed in connection with example 17f in Section 4 of this document. At step 1602, the process derives a chroma residual scaling factor during a conversion between a current video block of visual media data and a bitstream representation of the current video block. At step 1604, the process stores the chroma residual scaling factor for use with other video blocks of the visual media data. At step 1606, the process applies the chroma residual factor for the conversion of the current video block and the other video blocks into the bitstream representation.

[0167] FIG. 17 shows a flowchart of an example method for visual media processing. Steps of this flowchart are discussed in connection with example 17g in Section 4 of this document. At step 1702, during a conversion between a current video block of visual media data and a bitstream representation of the visual media data, the process computes a chroma residual factor of the current video block. At step 1704, the process stores, in a buffer, the chroma residual scaling factor for use with a second video block of the visual media data. At step 1706, the process removes the chroma residual scaling factor from the buffer, subsequent to the use.

[0168] Some embodiments discussed in this document are now presented in clause-based format.

[0169] A1. A method for video processing, comprising:

[0170] computing, during a conversion between a current video block of visual media data and a bitstream representation of the current video block, a cross-component linear model (CCLM) and/or a chroma residual scaling (CRS) factor for the current video block based, at least in part, on neighboring samples of a corresponding luma block which covers a top-left sample of a collocated luma block associated with the current video block, wherein one or more characteristics of the current video block are used for identifying the corresponding luma block.

[0171] A2. The method of clause A1, wherein the one or more characteristics of the current video block include: a size, a partition type, a location, or a coordination.

[0172] A3. The method of any one or more of clauses A1-A2, wherein the one or more characteristics of the current video block are associated with a color component of the current video block.

[0173] A4. The method of clause A3, wherein the one or more characteristics of the current video block are stored in a buffer for subsequent use.

[0174] A5. The method of any one or more of clauses A1-A3, wherein the current video block and the corresponding luma block are located inside a same coding tree unit (CTU) or same coding tree unit (CTU) row.

[0175] A6. The method of clause A5, wherein, if the current video block and the corresponding luma block are located inside the same coding tree unit (CTU) or same coding tree unit (CTU) row, then the one or more characteristics of the current video block are not stored.

[0176] A7. The method of clause A1, wherein the neighboring samples of the corresponding luma block are available when one or more conditions are satisfied, otherwise the neighboring samples are unavailable.

[0177] A8. The method of clause A7, wherein the one or more conditions include: a use of a coding mode of the current video block, a use of a coding mode of the neighboring samples of the corresponding luma block, a use of a type of a filter associated with the neighboring samples of the corresponding luma block, a location of the neighboring samples of the corresponding luma block relative to the current video blocks or sub-blocks thereof, a width of a current picture/subpicture/tile/tile group/VPDU/slice, and/or a height of a current picture/subpicture/tile/tile group/VPDU/slice/coding tree unit (CTU) row.

[0178] A9. The method of any one or more of clauses A7-A8, wherein, if a neighbouring sample is unavailable, then the neighbouring sample is substituted by a first-available adjacent sample.

[0179] A10. The method of clause A9, wherein the first-available adjacent sample is identified in accordance with a checking order.

[0180] A11. The method of clause A10, wherein the checking order is pre-defined.

[0181] A12. The method of clause A10, wherein the checking order is signalled in the bitstream representation.

[0182] A13. The method of any one or more of clauses A8-A9, wherein, if a neighbouring sample is unavailable, then the neighbouring sample is filled by a pre-determined or mapped value.

[0183] A14. The method of clause A13, wherein the pre-determined or mapped value is expressed as 1 << (bitDepth − 1), where bitDepth denoted a bit depth of samples in the collocated luma block.

[0184] A15. The method of clause A13, wherein the pre-determined or mapped value is based on a look up table (LUT).

[0185] B1. A method for visual media processing, comprising:

[0186] using a rule to make a determination of selectively enabling or disabling a chroma residual scaling (CRS) on color components of a current video block of visual media data, wherein the rule is based on coding mode information of the current video block and/or coding mode information of one or more neighbouring video blocks; and

[0187] performing a conversion between the current video block and a bitstream representation, based on the determination.

[0188] B2. The method of clause B1, wherein the current video block is a collocated video block.

[0189] B3. The method of clause B1, wherein the current video block is a current chroma block.

[0190] B4. The method of clause B1, wherein the current video block is a corresponding luma block which covers at least one sample of a collocated chroma block.

[0191] B5. The method of any one or more of clauses B1-B4, wherein the one or more neighbouring video blocks are adjacent video blocks.

[0192] B6. The method of any one or more of clauses B1-B4, wherein the one or more neighbouring video blocks are non-adjacent video blocks.

[0193] B7. The method of any one or more of clauses B1-B6, wherein the one or more neighbouring video blocks are multiple neighbouring blocks that cover multiple samples relative to the current video block.

[0194] B8. The method of any one or more of clauses B1-B6, wherein the rule specifies disabling the CRS, if reconstruction of a neighbouring video block makes use of samples in a slice/tile group associated with the current video block.

[0195] B9. The method of clause B8, wherein the rule specifies disabling the CRS, if coding mode information of the neighbouring video block is one of: intra mode, combined inter-intra prediction (CIIP) mode, or intra block copy (IBC) mode.

[0196] B10. The method of any one or more of clauses B8-B9, wherein a default value of chroma residual factor is used for applying the CRS.

[0197] B11. The method of clause B10, wherein the default value is expressed as $1 << (bitDepth - 1)$, where bitDepth denoted a bit depth of luma or chroma samples in the current video block.

[0198] B12. The method of clause B10, wherein the default value is based on a look up table (LUT).

[0199] B13. The method of clause B10, wherein the default value is pre-defined.

[0200] C1. A method for visual media processing:

[0201] using a single chroma residual scaling factor for at least one chroma block associated with video blocks in a slice or a tile group associated with a current video block of visual media data; and

[0202] performing a conversion between the current video block and a bitstream representation of the current video block.

[0203] C2. The method of clause C1, wherein the single chroma residual scaling factor for the at least one chroma block is a fixed value.

[0204] C3. The method of any one or more of clauses C1-C2, wherein the single chroma residual factor is based on an index of a linear model used in deriving the chroma residual scaling factor.

[0205] C4. The method of clause C3, wherein the linear model is piecewise linear.

[0206] C5. The method of clause C1, wherein the single chroma residual scaling factor for the at least one chroma block is pre-defined.

[0207] C6. The method of clause C1, wherein the single chroma residual scaling factor for the at least one chroma block is based on a bit depth of luma or chroma samples in the current video block.

[0208] C7. The method of clause C3, wherein the index of the linear model is derived based on a bit depth of luma or chroma samples in the current video block.

[0209] D1. A method for visual media processing:

[0210] deriving a chroma residual scaling factor during a conversion between a current video block of visual media data and a bitstream representation of the current video block;

[0211] storing the chroma residual scaling factor for use with other video blocks of the visual media data; and

[0212] applying the chroma residual factor for the conversion of the current video block and the other video blocks into the bitstream representation.

[0213] D2. The method of clause D1, wherein the chroma residual scaling factor is stored in a line buffer.

[0214] D3. A method for visual media processing:

[0215] during a conversion between a current video block of visual media data and a bitstream representation of the visual media data:

[0216] computing a chroma residual factor of the current video block;

[0217] storing, in a buffer, the chroma residual scaling factor for use with a second video block of the visual media data; and

[0218] subsequent to the use, removing the chroma residual scaling factor from the buffer.

[0219] D4. The method of clause D3, wherein, if the current video block and the second video block in the visual media data belong to different video regions, then the chroma residual scaling factor of the second video block is determined to be unavailable for use on the current video block.

[0220] D5. The method of clause D4, wherein the video regions include one of: a slice, a tile, a tile group, a virtual pipeline data unit (VPDU), a coding tree unit (CTU), or a CTU row.

[0221] D6. The method of clause D4, wherein the second video block is a neighbouring video block of the current video block.

[0222] D7. The method of clause D6, wherein the neighbouring video block is adjacent to the current video block.

[0223] D8. The method of clause D6, wherein the neighbouring video block is non-adjacent to the current video block.

[0224] D9. The method of any one or more of clauses D1-D8, wherein the chroma residual scaling factor is dynamically updated during the conversion.

[0225] D10. The method of clause D9, wherein the chroma residual scaling factor is stored in a table, and dynamically updating the chroma residual scaling factor stored in the table is in accordance with a first-in-first-out (FIFO) order.

[0226] D11. The method of clause D10, wherein the chroma residual scaling factor is stored in the table subsequent to decoding/encoding a chroma block.

[0227] D12. The method of clause D10, wherein, at an instant of time, the table stores at most entry of the chroma residual scaling factor.

[0228] D13. The method of clause D10, wherein the chroma residual scaling factor is stored in the table prior to decoding/encoding a picture, a slice, a tile, a tile group, , a virtual pipeline data unit (VPDU), a CTU, or a CTU row.

[0229] D14. The method of clause D13, wherein storing a default chroma residual scaling factor in the table results in refreshing the table.

[0230] D15. The method of clause D14, wherein the default chroma residual scaling factor is a null value when the table is refreshed.

[0231] D16. The method of any one or more of clauses A1-D15, wherein the conversion includes generating the bitstream representation from the current video block.

[0232] D17. The method of any one or more of clauses A1-D15, wherein the conversion includes generating pixel values of the current video block from the bitstream representation.

[0233] D18. A video encoder apparatus comprising a processor configured to implement a method recited in any one or more of clauses A1-D15.

[0234] D19. A video decoder apparatus comprising a processor configured to implement a method recited in any one or more of clauses A1-D15.

[0235] D20. A computer readable medium having code stored thereon, the code embodying processor-executable instructions for implementing a method recited in any one or more of clauses A1-D15.

[0236] In the present document, the term "video processing" or "visual media processing" may refer to video encoding, video decoding, video compression or video decompression. For example, video compression algorithms may be applied during conversion from pixel representation of a video to a corresponding bitstream representation or vice versa. The bitstream representation of a current video block may, for example, correspond to bits that are either co-located or spread in different places within the bitstream, as is defined by the syntax. For example, a macroblock may be encoded in terms of transformed and coded error residual values and also using bits in headers and other fields in the bitstream. Furthermore, during conversion, a decoder may parse a bitstream with the knowledge that some fields may be present, or absent, based on the determination, as is described in the above solutions. Similarly, an encoder may determine that certain syntax fields are or are not to be included and generate the coded representation accordingly by including or excluding the syntax fields from the coded representation.

[0237] From the foregoing, it will be appreciated that specific embodiments of the presently disclosed technology have been described herein for purposes of illustration, but that various modifications may be made without deviating from the scope of the invention. Accordingly, the presently disclosed technology is not limited except as by the appended claims.

[0238] Implementations of the subject matter and the functional operations described in this patent document can be implemented in various systems, digital electronic circuitry, or in computer software, firmware, or hardware, including the structures disclosed in this specification and their structural equivalents, or in combinations of one or more of them. Implementations of the subject matter described in this specification can be implemented as one or more computer program products, i.e., one or more modules of computer program instructions encoded on a tangible and non-transitory computer readable medium for execution by, or to control the operation of, data processing apparatus. The computer readable medium can be a machine-readable storage device, a machine-readable storage substrate, a memory device, a composition of matter effecting a machine-readable propagated signal, or a combination of one or more of them. The term "data processing unit" or "data processing apparatus" encompasses all apparatus, devices, and machines for processing data, including by way of example a programmable processor, a computer, or multiple processors or computers. The apparatus can include, in addition to hardware, code that creates an execution environment for the computer program in question,

e.g., code that constitutes processor firmware, a protocol stack, a database management system, an operating system, or a combination of one or more of them.

[0239] A computer program (also known as a program, software, software application, script, or code) can be written in any form of programming language, including compiled or interpreted languages, and it can be deployed in any form, including as a stand-alone program or as a module, component, subroutine, or other unit suitable for use in a computing environment. A computer program does not necessarily correspond to a file in a file system. A program can be stored in a portion of a file that holds other programs or data (e.g., one or more scripts stored in a markup language document), in a single file dedicated to the program in question, or in multiple coordinated files (e.g., files that store one or more modules, sub programs, or portions of code). A computer program can be deployed to be executed on one computer or on multiple computers that are located at one site or distributed across multiple sites and interconnected by a communication network.

[0240] The processes and logic flows described in this specification can be performed by one or more programmable processors executing one or more computer programs to perform functions by operating on input data and generating output. The processes and logic flows can also be performed by, and apparatus can also be implemented as, special purpose logic circuitry, e.g., an FPGA (field programmable gate array) or an ASIC (application specific integrated circuit).

[0241] Processors suitable for the execution of a computer program include, by way of example, both general and special purpose microprocessors, and any one or more processors of any kind of digital computer. Generally, a processor will receive instructions and data from a read only memory or a random access memory or both. The essential elements of a computer are a processor for performing instructions and one or more memory devices for storing instructions and data. Generally, a computer will also include, or be operatively coupled to receive data from or transfer data to, or both, one or more mass storage devices for storing data, e.g., magnetic, magneto optical disks, or optical disks. However, a computer need not have such devices. Computer readable media suitable for storing computer program instructions and data include all forms of nonvolatile memory, media and memory devices, including by way of example semiconductor memory devices, e.g., EPROM, EEPROM, and flash memory devices. The processor and the memory can be supplemented by, or incorporated in, special purpose logic circuitry.

[0242] It is intended that the specification, together with the drawings, be considered exemplary only, where exemplary means an example. As used herein, the use of "or" is intended to include "and/or", unless the context clearly indicates otherwise.

[0243] While this patent document contains many specifics, these should not be construed as limitations on the scope of any invention or of what may be claimed, but rather as descriptions of features that may be specific to particular embodiments of particular inventions. Certain features that are described in this

patent document in the context of separate embodiments can also be implemented in combination in a single embodiment. Conversely, various features that are described in the context of a single embodiment can also be implemented in multiple embodiments separately or in any suitable subcombination. Moreover, although features may be described above as acting in certain combinations and even initially claimed as such, one or more features from a claimed combination can in some cases be excised from the combination, and the claimed combination may be directed to a subcombination or variation of a subcombination.

[0244] Similarly, while operations are depicted in the drawings in a particular order, this should not be understood as requiring that such operations be performed in the particular order shown or in sequential order, or that all illustrated operations be performed, to achieve desirable results. Moreover, the separation of various system components in the embodiments described in this patent document should not be understood as requiring such separation in all embodiments.

[0245] Only a few implementations and examples are described and other implementations, enhancements and variations can be made based on what is described and illustrated in this patent document.

CLAIMS

1. A method of processing video data, comprising:

determining, during a conversion between a current chroma video block of a video and a bitstream of the video, that a scaling process is applied on chroma residual samples of the current chroma video block; and

performing the conversion by applying the scaling process on the chroma residual samples,

wherein in the scaling process, the chroma residual samples are scaled based on at least one scaling factor before being used to reconstruct the current chroma video block, and

wherein the at least one scaling factor is derived based on an averaged luma variable computed based on at least one neighboring luma block of a video unit of the video which is determined based on a luma sample corresponding to a top-left sample of the current chroma video block, and

wherein in response to a first neighboring luma block of the at least one neighboring luma block and the video unit being located in different video regions, the first neighboring luma block is treated as unavailable, and wherein the different video regions include at least one of a different slices or different tiles.

2. The method of claim 1, wherein the scaling process is performed based on a piecewise linear model, and wherein an index identifying a piece to which the averaged luma variable belongs, and the at least one scaling factor is derived based on the index.

3. The method of claim 1, wherein in response to the at least one neighboring luma block being unavailable, the averaged luma variable is set to a default fixed value.

4. The method of claim 3, wherein the default fixed value is used to find an index identifying the piece to which the default fixed value belongs, and the at least one scaling factor is calculated from the found index.

5. The method of claim 3, wherein the default fixed value depends on a bit depth of the video.

6. The method of claim 5, wherein the default fixed value is expressed as 1 << (bitDepth − 1), where bitDepth denotes the bit depth of the video.

7. The method of claim 1, wherein the at least one neighboring luma block is adjacent to the video unit.

8. The method of claim 1, wherein a location of the video unit is derived based on one or more characteristics of the current chroma video block or the video unit.

9. The method of claim 8, wherein the one or more characteristics include at least one of a block size and a location information.

10. The method of claim 8, wherein the one or more characteristics are refrained to be stored in a line buffer.

11. The method of claim 1, wherein the current chroma video block and the video unit are located inside a same coding tree unit.

12. The method of claim 1, wherein the current chroma video block and the video unit are located inside a same coding tree unit row.

13. The method of claim 1, wherein the conversion includes encoding the current chroma video block into the bitstream.

14. The method of claim 1, wherein the conversion includes decoding the current chroma video block from the bitstream.

15. An apparatus for processing video data comprising a processor and a non-transitory memory with instructions thereon, wherein the instructions upon execution by the processor, cause the processor to:

determine, during a conversion between a current chroma video block of a video and a bitstream of the video, that a scaling process is applied on chroma residual samples of the current chroma video block; and

perform the conversion by applying the scaling process on the chroma residual samples,

wherein in the scaling process, the chroma residual samples are scaled based on at least one scaling factor before being used to reconstruct the current chroma video block, and

wherein the at least one scaling factor is derived based on an averaged luma variable computed based on at least one neighboring luma block of a video unit of the video which is determined based on a luma sample corresponding to a top-left sample of the current chroma video block, and

wherein in response to a first neighboring luma block of the at least one neighboring luma block and the video unit being located in different video regions, the first neighboring luma block is treated as unavailable, and wherein the different video regions include at least one of different slices or different tiles.

16. The apparatus of claim 15, wherein the scaling process is performed based on a piecewise linear model, and wherein an index identifying a piece to which the averaged luma variable belongs, and the at least one scaling factor is derived based on the index.

17. The apparatus of claim 15, wherein in response to the at least one neighboring luma block being unavailable, the averaged luma variable is set to a default fixed value.

18. The apparatus of claim 15, wherein the default fixed value is used to find an index identifying the piece to which the default fixed value belongs, and the at least one scaling factor is calculated from the found index.

19. A non-transitory computer-readable storage medium storing instructions that cause a processor to:

determine, during a conversion between a current chroma video block of a video and a bitstream of the video, that a scaling process is applied on chroma residual samples of the current chroma video block; and

perform the conversion by applying the scaling process on the chroma residual samples,

wherein in the scaling process, the chroma residual samples are scaled based on at least one scaling factor before being used to reconstruct the current chroma video block, and

wherein the at least one scaling factor is derived based on an averaged luma variable computed based on at least one neighboring luma block of a video unit of the video which is determined based on a luma sample corresponding to a top-left sample of the current chroma video block, and

wherein in response to a first neighboring luma block of the at least one neighboring luma block and the video unit being located in different video regions, the first neighboring luma block is treated as unavailable, and wherein the different video regions include at least one of different slices or different tiles.

20. A non-transitory computer-readable recording medium storing a bitstream of a video which is generated by a method performed by a video processing apparatus, wherein the method comprises:

determining that a scaling process is applied on chroma residual samples of the current chroma video block; and

generating the bitstream by applying the scaling process on the chroma residual samples,

wherein in the scaling process, the chroma residual samples are scaled based on at least one scaling factor before being used to reconstruct the current chroma video block, and

wherein the at least one scaling factor is derived based on an averaged luma variable computed based on at least one neighboring luma block of a video unit of the video which is determined based on a luma sample corresponding to a top-left sample of the current chroma video block, and

wherein in response to a first neighboring luma block of the at least one neighboring luma block and the video unit being located in different video regions, the first neighboring luma block is treated as unavailable, and wherein the different video regions include at least one of different slices or different tiles.