

DERIVATION OF LINEAR PARAMETER IN CROSS-COMPONENT VIDEO CODING

CROSS REFERENCE TO RELATED APPLICATIONS

[001] Under the applicable patent law and/or rules pursuant to the Paris Convention, this application is made to timely claim the priority to and benefits of International Patent Application No. PCT/CN2019/115034, filed on November 1, 2019. For all purposes under the law, the entire disclosures of the aforementioned applications are incorporated by reference as part of the disclosure of this application.

TECHNICAL FIELD

[002] This document is related to video and image coding and decoding technologies.

BACKGROUND

[003] Digital video 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

[004] The disclosed techniques may be used by video or image decoder or encoder embodiments for performing encoding or decoding using cross-component linear model prediction.

[005] In one example aspect, a method of processing video is disclosed. The method includes deriving, for a conversion between a chroma block of a video and a bitstream representation of the video, parameters of a cross-component linear model by using downsampled luma samples that are generated from N above neighboring lines of a collocated luma block of the chroma block using a downsampling filter, where N is a positive integer; and performing the conversion using a predicted chroma block generated using the cross-component linear model.

[006] In another example aspect, a method of processing video is disclosed. The method includes determining, for a conversion between a video region of a component of a video and a bitstream representation of the video, a maximum allowed block size for a video block coded using a transform skip mode; and performing the conversion based on the determining.

[007] In another example aspect, a method of processing video is disclosed. The method includes performing a conversion between a video comprising video blocks and a bitstream representation of the video according to a first rule and a second rule, wherein a transform skip coding tool is used for coding a first portion of the video blocks, wherein a transform coding tool is used for coding a second portion of the video blocks, wherein the first rule specifies a maximum allowed block size for the first portion of the video blocks and the second rule specifies a maximum allowed block size for the second portion of the video blocks, and wherein the maximum allowed block size for the first portion of the video blocks is different from the maximum allowed block size for the second portion of the video block.

[008] In another example aspect, a method of processing video is disclosed. The method includes performing a conversion between a video comprising one or more chroma blocks and a bitstream representation of the video, wherein the bitstream representation conforms to a format rule that specifies that whether a syntax element to indicate usage of a transform skip tool is included in the bitstream representation depends on a maximum allowed size for a chroma block that is coded using the transform skip tool.

[009] In another example aspect, a method of processing video is disclosed. The method includes performing a conversion between a video comprising one or more first video blocks of a first chroma component and one or more second video blocks of a second chroma component and a bitstream representation of the video, wherein the bitstream representation conforms to a format rule that specifies to use a syntax element that jointly indicates availability of a transform skip tool for coding the one or more first chroma blocks and the one or more second chroma blocks.

[0010] In another example aspect, the above-described method may be implemented by a video encoder apparatus that comprises a processor.

[0011] In yet another example aspect, these methods may be embodied in the form of processor-executable instructions and stored on a computer-readable program medium.

[0012] These, and other, aspects are further described in the present document.

BRIEF DESCRIPTION OF THE DRAWINGS

- [0013] FIG. 1A shows Nominal vertical and horizontal locations of 4:2:2 luma and chroma samples in a picture.
- [0014] FIG. 1B shows an example of a video encoder.
- [0015] FIG. 2 shows examples of 67 intra prediction modes.
- [0016] FIG. 3 shows examples of horizontal and vertical traverse scans.
- [0017] FIG. 4 shows examples of locations of the samples used for the derivation of α and β .
- [0018] FIG. 5 shows example of dividing a block of 4x8 samples into two independently decodable areas.
- [0019] FIG. 6 shows an example order of processing of the rows of pixels to maximize throughput for 4xN blocks with vertical predictor.
- [0020] FIG. 7 shows an example of a low-Frequency Non-Separable Transform (LFNST) process.
- [0021] FIG. 8 shows an example of neighbouring chroma samples and downsampled collocated neighbouring luma samples used in the derivation of CCLM parameters for 4:2:2 videos.
- [0022] FIG. 9 shows an example of a video processing apparatus.
- [0023] FIG. 10 shows a block diagram of a video encoder.
- [0024] FIG. 11 is a flowchart for an example of a video processing method based on some implementations of the disclosed technology.
- [0025] FIG. 12 is a block diagram for an example of a video processing system.
- [0026] FIG. 13 is a block diagram that illustrates an example video coding system.
- [0027] FIG. 14 is a block diagram that illustrates an encoder in accordance with some embodiments of the disclosed technology.
- [0028] FIG. 15 is a block diagram that illustrates a decoder in accordance with some embodiments of the disclosed technology.
- [0029] FIGS. 16A and 16B are flowcharts for examples of video processing methods based on some implementations of the disclosed technology.

DETAILED DESCRIPTION

[0030] The present document provides various techniques that can be used by a decoder of image or video bitstreams to improve the quality of decompressed or decoded digital video or images. For brevity, the term “video” is used herein to include both a sequence of pictures (traditionally called video) and individual images. Furthermore, a video encoder may also implement these techniques during the process of encoding in order to reconstruct decoded frames used for further encoding.

[0031] Section headings are used in the present document for ease of understanding and do not limit the embodiments and techniques to the corresponding sections. As such, embodiments from one section can be combined with embodiments from other sections.

1. Brief summary

[0032] This invention is related to video coding technologies. Specifically, it is related cross-component linear model prediction and other coding tools in image/video coding. It may be applied to the existing video coding standard like HEVC, or the standard (Versatile Video Coding) to be finalized. It may be also applicable to future video coding standards or video codec.

2. Video coding introduction

[0033] Video coding standards have evolved primarily through the development of the well-known ITU-T and ISO/IEC standards. The ITU-T produced H.261 and H.263, ISO/IEC produced MPEG-1 and MPEG-4 Visual, and the two organizations jointly produced the H.262/MPEG-2 Video and H.264/MPEG-4 Advanced Video Coding (AVC) and H.265/HEVC standards. Since H.262, the video coding standards are based on the hybrid video coding structure wherein temporal prediction plus transform coding are utilized. To explore the future video coding technologies beyond HEVC, Joint Video Exploration Team (JVET) was founded by VCEG and MPEG jointly in 2015. Since then, many new methods have been adopted by JVET and put into the reference software named Joint Exploration Model (JEM). In April 2018, the Joint Video Expert Team (JVET) between VCEG (Q6/16) and ISO/IEC JTC1 SC29/WG11 (MPEG) was created to work on the VVC standard targeting at 50% bitrate reduction compared to HEVC.

2.1. Color space and chroma subsampling

[0034] Color space, also known as the color model (or color system), is an abstract mathematical model which simply describes the range of colors as tuples of numbers, typically as 3 or 4 values or color components (e.g. RGB). Basically speaking, color space is an elaboration of the coordinate system and sub-space.

[0035] For video compression, the most frequently used color spaces are YCbCr and RGB.

[0036] YCbCr, Y'CbCr, or Y Pb/Cb Pr/Cr, also written as YCBCR or Y'CBCR, is a family of color spaces used as a part of the color image pipeline in video and digital photography systems. Y' is the luma component and CB and CR are the blue-difference and red-difference chroma components. Y' (with prime) is distinguished from Y, which is luminance, meaning that light intensity is nonlinearly encoded based on gamma corrected RGB primaries.

[0037] Chroma subsampling is the practice of encoding images by implementing less resolution for chroma information than for luma information, taking advantage of the human visual system's lower acuity for color differences than for luminance.

2.1.1. 4:4:4

[0038] Each of the three Y'CbCr components have the same sample rate, thus there is no chroma subsampling. This scheme is sometimes used in high-end film scanners and cinematic post production.

2.1.2. 4:2:2

[0039] The two chroma components are sampled at half the sample rate of luma: the horizontal chroma resolution is halved while the vertical chroma resolution is unchanged. This reduces the bandwidth of an uncompressed video signal by one-third with little to no visual difference. An example of nominal vertical and horizontal locations of 4:2:2 color format is depicted in FIG. 1A in VVC working draft.

2.1.3. 4:2:0

[0040] In 4:2:0, the horizontal sampling is doubled compared to 4:1:1, but as the Cb and Cr channels are only sampled on each alternate line in this scheme, the vertical resolution is halved. The data rate is thus the same. Cb and Cr are each subsampled at a factor of 2 both horizontally and vertically. There are three variants of 4:2:0 schemes, having different horizontal and vertical siting.

- In MPEG-2, Cb and Cr are cosited horizontally. Cb and Cr are sited between pixels in the vertical direction (sited interstitially).
- In JPEG/JFIF, H.261, and MPEG-1, Cb and Cr are sited interstitially, halfway between alternate luma samples.
- In 4:2:0 DV, Cb and Cr are co-sited in the horizontal direction. In the vertical direction, they are co-sited on alternating lines.

Table 2-1. SubWidthC and SubHeightC values derived from chroma_format_idc and separate_colour_plane_flag

chroma_format_idc	separate_colour_plane_flag	Chroma format	SubWidthC	SubHeightC
0	0	Monochrome	1	1
1	0	4:2:0	2	2
2	0	4:2:2	2	1
3	0	4:4:4	1	1
3	1	4:4:4	1	1

2.2. Coding flow of a typical video codec

[0041] FIG. 1B shows an example of encoder block diagram of VVC, which contains three in-loop filtering blocks: deblocking filter (DF), sample adaptive offset (SAO) and ALF. Unlike DF, which uses predefined filters, SAO and ALF utilize the original samples of the current picture to reduce the mean square errors between the original samples and the reconstructed samples by adding an offset and by applying a finite impulse response (FIR) filter, respectively, with coded side information signalling the offsets and filter coefficients. ALF is located at the last processing stage of each picture and can be regarded as a tool trying to catch and fix artifacts created by the previous stages.

2.3. Intra mode coding with 67 intra prediction modes

[0042] To capture the arbitrary edge directions presented in natural video, the number of directional intra modes is extended from 33, as used in HEVC, to 65. The additional directional modes 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.

[0043] Conventional angular intra prediction directions are defined from 45 degrees to -135 degrees in clockwise direction as shown in FIG. 2. In VTM, several conventional angular intra prediction modes are adaptively replaced with wide-angle intra prediction modes for the non-square blocks. The replaced modes are signalled using the original method and remapped to the indexes of wide angular modes after parsing. The total number of intra prediction modes is unchanged, i.e., 67, and the intra mode coding is unchanged.

[0044] In the HEVC, every intra-coded block has a square shape and the length of each of its side is a power of 2. Thus, no division operations are required to generate an intra-predictor using DC mode. In VVC, blocks can have a rectangular shape that necessitates the use of a division operation per block in the general case. To avoid division operations for DC prediction, only the longer side is used to compute the average for non-square blocks.

FIG. 2 shows examples of 67 intra prediction modes.

2.4. Inter prediction

[0045] For each inter-predicted CU, motion parameters consisting of motion vectors, reference picture indices and reference picture list usage index, and additional information needed for the new coding feature of VVC to be used for inter-predicted sample generation. The motion parameter can be signalled in an explicit or implicit manner. When a CU is coded with skip mode, the CU is associated with one PU and has no significant residual coefficients, no coded motion vector delta or reference picture index. A merge mode is specified whereby the motion parameters for the current CU are obtained from neighbouring CUs, including spatial and temporal candidates, and additional schedules introduced in VVC. The merge mode can be applied to any inter-predicted CU, not only for skip mode. The alternative to merge mode is the explicit transmission of motion parameters, where motion vector, corresponding reference picture index for each reference picture list and reference picture list usage flag and other needed information are signalled explicitly per each CU.

2.5. Intra block copy (IBC)

[0046] Intra block copy (IBC) is a tool adopted in HEVC extensions on SCC. It is well known that it significantly improves the coding efficiency of screen content materials. Since IBC mode is implemented as a block level coding mode, block matching (BM) is performed at the encoder to find the optimal block vector (or motion vector) for each CU. Here, a block vector is used to

indicate the displacement from the current block to a reference block, which is already reconstructed inside the current picture. The luma block vector of an IBC-coded CU is in integer precision. The chroma block vector rounds to integer precision as well. When combined with AMVR, the IBC mode can switch between 1-pel and 4-pel motion vector precisions. An IBC-coded CU is treated as the third prediction mode other than intra or inter prediction modes. The IBC mode is applicable to the CUs with both width and height smaller than or equal to 64 luma samples.

[0047] At the encoder side, hash-based motion estimation is performed for IBC. The encoder performs RD check for blocks with either width or height no larger than 16 luma samples. For non-merge mode, the block vector search is performed using hash-based search first. If hash search does not return valid candidate, block matching based local search will be performed.

[0048] In the hash-based search, hash key matching (32-bit CRC) between the current block and a reference block is extended to all allowed block sizes. The hash key calculation for every position in the current picture is based on 4x4 sub-blocks. For the current block of a larger size, a hash key is determined to match that of the reference block when all the hash keys of all 4x4 sub-blocks match the hash keys in the corresponding reference locations. If hash keys of multiple reference blocks are found to match that of the current block, the block vector costs of each matched reference are calculated and the one with the minimum cost is selected.

In block matching search, the search range is set to cover both the previous and current CTUs.

At CU level, IBC mode is signalled with a flag and it can be signalled as IBC AMVP mode or IBC skip/merge mode as follows:

[0049] – IBC skip/merge mode: a merge candidate index is used to indicate which of the block vectors in the list from neighbouring candidate IBC coded blocks is used to predict the current block. The merge list consists of spatial, HMVP, and pairwise candidates.

[0050] – IBC AMVP mode: block vector difference is coded in the same way as a motion vector difference. The block vector prediction method uses two candidates as predictors, one from left neighbour and one from above neighbour (if IBC coded). When either neighbour is not available, a default block vector will be used as a predictor. A flag is signalled to indicate the block vector predictor index.

2.6. Palette mode

[0051] For palette mode signalling, the palette mode is coded as a prediction mode for a coding unit, i.e., the prediction modes for a coding unit can be MODE_INTRA, MODE_INTER, MODE_IBC and MODE_PLT. If the palette mode is utilized, the pixels values in the CU are represented by a small set of representative colour values. The set is referred to as the palette. For pixels with values close to the palette colors, the palette indices are signalled. For pixels with values outside the palette, the pixel is denoted with an escape symbol and the quantized pixel values are signalled directly.

[0052] To decode a palette encoded block, the decoder needs to decode palette colors and indices. Palette colors are described by a palette table and encoded by palette table coding tools. An escape flag is signalled for each CU to indicate if escape symbols are present in the current CU. If escape symbols are present, the palette table is augmented by one and the last index is assigned to the escape mode. Palette indices of all pixels in a CU form a palette index map and are encoded by palette index map coding tools.

[0053] For coding of the palette table, a palette predictor is maintained. The predictor is initialized at the beginning of each slice where predictor is reset to 0. For each entry in the palette predictor, a reuse flag is signalled to indicate whether it is part of the current palette. The reuse flags are sent using run-length coding of zeros. After this, the number of new palette entries are signalled using exponential Golomb code of order 0. Finally, the component values for the new palette entries are signalled. After encoding the current CU, the palette predictor will be updated using the current palette, and entries from the previous palette predictor which are not reused in the current palette will be added at the end of new palette predictor until the maximum size allowed is reached (palette stuffing).

[0054] For coding the palette index map, the indices are coded using horizontal and vertical traverse scans as shown in FIG. 3. The scan order is explicitly signalled in the bitstream using the palette_transpose_flag.

[0055] FIG. 3 shows examples of horizontal and vertical traverse scans.

[0056] The palette indices are coded using two main palette sample modes: 'INDEX' and 'COPY_ABOVE'. The mode is signalled using a flag except for the top row when horizontal scan is used, the first column when the vertical scan is used, or when the previous mode was 'COPY_ABOVE'. In the 'COPY_ABOVE' mode, the palette index of the sample in the row

above is copied. In the 'INDEX' mode, the palette index is explicitly signalled. For both 'INDEX' and 'COPY_ABOVE' modes, a run value is signalled which specifies the number pixels that are coded using the same mode.

[0057] The encoding order for index map is as follows: First, the number of index values for the CU is signalled. This is followed by signalling of the actual index values for the entire CU using truncated binary coding. Both the number of indices as well as the index values are coded in bypass mode. This groups the index-related bypass bins together. Then the palette mode (INDEX or COPY_ABOVE) and run are signalled in an interleaved manner. Finally, the component escape values corresponding to the escape samples for the entire CU are grouped together and coded in bypass mode. An additional syntax element, *last_run_type_flag*, is signalled after signalling the index values. This syntax element, in conjunction with the number of indices, eliminates the need to signal the run value corresponding to the last run in the block.

[0058] In VTM, dual tree is enabled for I slice which separate the coding unit partitioning for Luma and Chroma. Hence, in this proposal, palette is applied on Luma (Y component) and Chroma (Cb and Cr components) separately. If dual tree is disabled, palette will be applied on Y, Cb, Cr components jointly, same as in HEVC palette.

2.7. Cross-component linear model prediction

[0059] A cross-component linear model (CCLM) prediction mode is used in the VVC, for which the chroma samples are predicted based on the reconstructed luma samples of the same CU by using a linear model as follows:

$$\text{pred}_C(i, j) = \alpha \cdot \text{rec}_L'(i, j) + \beta \quad (2-1)$$

[0060] where $\text{pred}_C(i, j)$ represents the predicted chroma samples in a CU and $\text{rec}_L(i, j)$ represents the downsampled reconstructed luma samples of the same CU.

[0061] FIG. 4 shows an example of the location of the left and above samples and the sample of the current block involved in the LM mode.

[0062] FIG. 4 shows examples of locations of the samples used for the derivation of α and β .

[0063] Besides the above template and left template can be used to calculate the linear model coefficients together in LM mode, they also can be used alternatively in the other 2 LM modes, called LM_A, and LM_L modes. In LM_A mode, only the above template is used to calculate the linear model coefficients. To get more samples, the above template is extended to (W+H). In LM_L mode, only left template is used to calculate the linear model coefficients. To get more

samples, the left template is extended to (H+W). For a non-square block, the above template is extended to W+W, the left template is extended to H+H.

[0064] The CCLM parameters (α and β) are derived with at most four neighbouring chroma samples and their corresponding down-sampled luma samples. Suppose the current chroma block dimensions are $W \times H$, then W' and H' are set as

- $W'=W$, $H'=H$ when LM mode is applied;
- $W'=W+H$ when LM-A mode is applied;
- $H'=H+W$ when LM-L mode is applied;

[0065] The above neighbouring positions are denoted as $S[0, -1] \dots S[W'-1, -1]$ and the left neighbouring positions are denoted as $S[-1, 0] \dots S[-1, H'-1]$. Then the four samples are selected as

- $S[W'/4, -1]$, $S[3W'/4, -1]$, $S[-1, H'/4]$, $S[-1, 3H'/4]$ when LM mode is applied and both above and left neighbouring samples are available;
- $S[W'/8, -1]$, $S[3W'/8, -1]$, $S[5W'/8, -1]$, $S[7W'/8, -1]$ when LM-A mode is applied or only the above neighbouring samples are available;
- $S[-1, H'/8]$, $S[-1, 3H'/8]$, $S[-1, 5H'/8]$, $S[-1, 7H'/8]$ when LM-L mode is applied or only the left neighbouring samples are available;

[0066] The four neighbouring luma samples at the selected positions are down-sampled and compared four times to find two smaller values: x^0_A and x^1_A , and two larger values: x^0_B and x^1_B . Their corresponding chroma sample values are denoted as y^0_A, y^1_A, y^0_B and y^1_B . Then x_A, x_B, y_A and y_B are derived as:

$$X_a = (x^0_A + x^1_A + 1) \gg 1; X_b = (x^0_B + x^1_B + 1) \gg 1; Y_a = (y^0_A + y^1_A + 1) \gg 1; Y_b = (y^0_B + y^1_B + 1) \gg 1 \quad (2-2)$$

[0067] Finally, the linear model parameters α and β are obtained according to the following equations.

$$\alpha = \frac{Y_a - Y_b}{X_a - X_b} \quad (2-3)$$

$$\beta = Y_b - \alpha \cdot X_b \quad (2-4)$$

[0068] 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:

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

[0069] 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.

[0070] 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.

$$\text{rec}_L'(i, j) = \left[\frac{\text{rec}_L(2i - 1, 2j - 1) + 2 \cdot \text{rec}_L(2i - 1, 2j) + \text{rec}_L(2i + 1, 2j - 1) + \text{rec}_L(2i + 1, 2j) + 4}{\text{rec}_L(2i - 1, 2j) + 2 \cdot \text{rec}_L(2i, 2j) + \text{rec}_L(2i + 1, 2j) + 4} \right] \gg 3 \quad (2-6)$$

$$\text{rec}_L'(i, j) = \left[\frac{\text{rec}_L(2i, 2j - 1) + \text{rec}_L(2i - 1, 2j) + 4 \cdot \text{rec}_L(2i, 2j) + \text{rec}_L(2i + 1, 2j) + \text{rec}_L(2i, 2j + 1) + 4}{\text{rec}_L(2i + 1, 2j) + \text{rec}_L(2i, 2j + 1) + 4} \right] \gg 3 \quad (2-7)$$

[0071] 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.

[0072] This parameter computation is performed as part of the decoding process, and not just as an encoder search operation. As a result, no syntax is used to convey the α and β values to the decoder.

[0073] 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 (LM, LM_A, and LM_L). Chroma mode signalling and derivation process are shown in Table 2-2. 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.

Table 2-2. Derivation of chroma prediction mode from luma mode when cclm is enabled

Chroma prediction mode	Corresponding luma intra prediction mode				
	0	50	18	1	X (0 <= X <= 66)
0	66	0	0	0	0
1	50	66	50	50	50
2	18	18	66	18	18
3	1	1	1	66	1
4	81	81	81	81	81
5	82	82	82	82	82
6	83	83	83	83	83
7	0	50	18	1	X

2.8. Block differential pulse-code modulation coding (BDPCM)

[0074] BDPCM is proposed in JVET-M0057. Due to the shape of the horizontal (resp. vertical) predictors, which use the left (A) (resp. top (B)) pixel for prediction of the current pixel, the most throughput-efficient way of processing the block is to process all the pixels of one column (resp. line) in parallel, and to process these columns (resp. lines) sequentially. In order to increase throughput, we introduce the following process: a block of width 4 is divided into two halves with a horizontal frontier when the predictor chosen on this block is vertical, and a block of height 4 is divided into two halves with a vertical frontier when the predictor chosen on this block is horizontal.

[0075] When a block is divided, samples from one area are not allowed to use pixels from another area to compute the prediction: if this situation occurs, the prediction pixel is replaced by the reference pixel in the prediction direction. This is shown in FIG. 5 for different positions of current pixel X in a 4x8 block predicted vertically.

[0076] FIG. 5 shows example of dividing a block of 4x8 samples into two independently decodable areas.

[0077] Thanks to this property, it becomes now possible to process a 4x4 block in 2 cycles, and a 4x8 or 8x4 block in 4 cycles, and so on, as shown on FIG. 6.

[0078] FIG. 6 shows an example order of processing of the rows of pixels to maximize throughput for 4xN blocks with vertical predictor.

[0079] Table 2-3 summarizes the number of cycles required to process the block, depending on the block size. It is trivial to show that any block which has both dimensions larger or equal to 8 can be processed in 8 pixels per cycle or more.

Table 2-3. Worst case throughput for blocks of size $4 \times N$, $N \times 4$

Block size	4x4	4x8, 8x4	4x16, 16x4	4x32, 32x4
Cycles	2	4	8	16
Pixels	16	32	64	128
Throughput (pixels/cycle)	8	8	8	8

2.9. Quantized residual domain BDPCM

[0080] In JVET-N0413, quantized residual domain BDPCM (denote as RBDPCM hereinafter) is proposed. The intra prediction is done on the entire block by sample copying in prediction direction (horizontal or vertical prediction) similar to intra prediction. The residual is quantized and the delta between the quantized residual and its predictor (horizontal or vertical) quantized value is coded.

[0081] For a block of size M (rows) $\times$ N (cols), let $r_{i,j}$, $0 \leq i \leq M-1$, $0 \leq j \leq N-1$ be the prediction residual after performing intra prediction horizontally (copying left neighbour pixel value across the the predicted block line by line) or vertically (copying top neighbour line to each line in the predicted block) using unfiltered samples from above or left block boundary samples. Let $Q(r_{i,j})$, $0 \leq i \leq M-1$, $0 \leq j \leq N-1$ denote the quantized version of the residual $r_{i,j}$, where residual is difference between original block and the predicted block values. Then the block DPCM is applied to the quantized residual samples, resulting in modified $M \times N$ array $\tilde{R}$ with elements $\tilde{r}_{i,j}$. When vertical BDPCM is signalled:

$$\tilde{r}_{i,j} = \begin{cases} Q(r_{i,j}), & i = 0, 0 \leq j \leq (N-1) \\ Q(r_{i,j}) - Q(r_{(i-1),j}), & 1 \leq i \leq (M-1), 0 \leq j \leq (N-1) \end{cases} \quad (2-8)$$

[0082] For horizontal prediction, similar rules apply, and the residual quantized samples are obtained by

$$\tilde{r}_{i,j} = \begin{cases} Q(r_{i,j}), & 0 \leq i \leq (M-1), j = 0 \\ Q(r_{i,j}) - Q(r_{i,(j-1)}), & 0 \leq i \leq (M-1), 1 \leq j \leq (N-1) \end{cases} \quad (2-9)$$

[0083] The residual quantized samples $\tilde{r}_{i,j}$ are sent to the decoder.

[0084] On the decoder side, the above calculations are reversed to produce $Q(r_{i,j})$, $0 \leq i \leq M-1$, $0 \leq j \leq N-1$. For vertical prediction case,

$$Q(r_{i,j}) = \sum_{k=0}^i \tilde{r}_{k,j}, \quad 0 \leq i \leq (M-1), \quad 0 \leq j \leq (N-1). \quad (2-10)$$

[0085] For horizontal case,

$$Q(r_{i,j}) = \sum_{k=0}^j \tilde{r}_{i,k}, \quad 0 \leq i \leq (M-1), \quad 0 \leq j \leq (N-1). \quad (2-11)$$

[0086] The inverse quantized residuals, $Q^{-1}(Q(r_{i,j}))$, are added to the intra block prediction values to produce the reconstructed sample values.

[0087] The main benefit of this scheme is that the invert DPCM can be done on the fly during coefficient parsing simply adding the predictor as the coefficients are parsed or it can be performed after parsing.

[0088] Transform skip is always used in quantized residual domain BDPCM.

2.10. Multiple Transform Set (MTS) in VVC

[0089] In VTM, large block-size transforms, up to 64×64 in size, are enabled, which is primarily useful for higher resolution video, e.g., 1080p and 4K sequences. High frequency transform coefficients are zeroed out for the transform blocks with size (width or height, or both width and height) equal to 64, so that only the lower-frequency coefficients are retained. For example, for an M×N transform block, with M as the block width and N as the block height, when M is equal to 64, only the left 32 columns of transform coefficients are kept. Similarly, when N is equal to 64, only the top 32 rows of transform coefficients are kept. When transform skip mode is used for a large block, the entire block is used without zeroing out any values. The VTM also supports configurable max transform size in SPS, such that encoder has the flexibility to choose up to 16-length, 32-length or 64-length transform size depending on the need of specific implementation.

[0090] In addition to DCT-II which has been employed in HEVC, a Multiple Transform Selection (MTS) scheme is used for residual coding both inter and intra coded blocks. It uses multiple selected transforms from the DCT8/DST7. The newly introduced transform matrices are DST-VII and DCT-VIII. Table 2-4 shows the basis functions of the selected DST/DCT.

Table 2-4. Transform basis functions of DCT-II/VIII and DSTVII for N-point input

Transform Type	Basis function $T_i(j), i, j = 0, 1, \dots, N-1$
DCT-II	$T_i(j) = \omega_0 \cdot \sqrt{\frac{2}{N}} \cdot \cos\left(\frac{\pi \cdot i \cdot (2j + 1)}{2N}\right)$ <p>where, $\omega_0 = \begin{cases} \sqrt{\frac{2}{N}} & i = 0 \\ 1 & i \neq 0 \end{cases}$</p>
DCT-VIII	$T_i(j) = \sqrt{\frac{4}{2N + 1}} \cdot \cos\left(\frac{\pi \cdot (2i + 1) \cdot (2j + 1)}{4N + 2}\right)$
DST-VII	$T_i(j) = \sqrt{\frac{4}{2N + 1}} \cdot \sin\left(\frac{\pi \cdot (2i + 1) \cdot (j + 1)}{2N + 1}\right)$

[0091] In order to keep the orthogonality of the transform matrix, the transform matrices are quantized more accurately than the transform matrices in HEVC. To keep the intermediate values of the transformed coefficients within the 16-bit range, after horizontal and after vertical transform, all the coefficients are to have 10-bit.

[0092] In order to control MTS scheme, separate enabling flags are specified at SPS level for intra and inter, respectively. When MTS is enabled at SPS, a CU level flag is signalled to indicate whether MTS is applied or not. Here, MTS is applied only for luma. The MTS CU level flag is signalled when the following conditions are satisfied.

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

[0093] If MTS CU flag is equal to zero, then DCT2 is applied in both directions. However, if MTS CU flag is equal to one, then two other flags are additionally signalled to indicate the transform type for the horizontal and vertical directions, respectively. Transform and signalling mapping table as shown in Table 2-5. Unified the transform selection for ISP and implicit MTS is used by removing the intra-mode and block-shape dependencies. If current block is ISP mode or if the current block is intra block and both intra and inter explicit MTS is on, then only DST7 is used for both horizontal and vertical transform cores. When it comes to transform matrix precision, 8-bit primary transform cores are used. Therefore, all the transform cores used in HEVC are kept as the same, including 4-point DCT-2 and DST-7, 8-point, 16-point and 32-point

DCT-2. Also, other transform cores including 64-point DCT-2, 4-point DCT-8, 8-point, 16-point, 32-point DST-7 and DCT-8, use 8-bit primary transform cores.

Table 2-5. Transform and signalling mapping table

MTS_CU_flag	MTS_Hor_flag	MTS_Ver_flag	Intra/inter	
			Horizontal	Vertical
0			DCT2	
1	0	0	DST7	DST7
	0	1	DCT8	DST7
	1	0	DST7	DCT8
	1	1	DCT8	DCT8

[0094] To reduce the complexity of large size DST-7 and DCT-8, High frequency transform coefficients are zeroed out for the DST-7 and DCT-8 blocks with size (width or height, or both width and height) equal to 32. Only the coefficients within the 16x16 lower-frequency region are retained.

[0095] As in HEVC, the residual of a block can be coded with transform skip mode. To avoid the redundancy of syntax coding, the transform skip flag is not signalled when the CU level MTS_CU_flag is not equal to zero. The block size limitation for transform skip is the same to that for MTS in JEM4, which indicate that transform skip is applicable for a CU when both block width and height are equal to or less than 32. Note that implicit MTS transform is set to DCT2 when LFNST or MIP is activated for the current CU. Also, the implicit MTS can be still enabled when MTS is enabled for inter coded blocks.

2.11. Low-Frequency Non-Separable Transform (LFNST)

[0096] In VVC, LFNST (low-frequency non-separable transform), which is known as reduced secondary transform, is applied between forward primary transform and quantization (at encoder) and between de-quantization and inverse primary transform (at decoder side) as shown in FIG. 7. In LFNST, 4x4 non-separable transform or 8x8 non-separable transform is applied

according to block size. For example, 4x4 LFNST is applied for small blocks (i.e., min (width, height) < 8) and 8x8 LFNST is applied for larger blocks (i.e., min (width, height) > 4).

[0097] FIG. 7 shows an example of a low-Frequency Non-Separable Transform (LFNST) process.

[0098] Application of a non-separable transform, which is being used in LFNST, is described as follows using input as an example. To apply 4x4 LFNST, the 4x4 input block X

$$X = \begin{bmatrix} X_{00} & X_{01} & X_{02} & X_{03} \\ X_{10} & X_{11} & X_{12} & X_{13} \\ X_{20} & X_{21} & X_{22} & X_{23} \\ X_{30} & X_{31} & X_{32} & X_{33} \end{bmatrix} \quad (2-12)$$

[0099] is first represented as a vector $\vec{X}$:

$$\vec{X} = [X_{00} \ X_{01} \ X_{02} \ X_{03} \ X_{10} \ X_{11} \ X_{12} \ X_{13} \ X_{20} \ X_{21} \ X_{22} \ X_{23} \ X_{30} \ X_{31} \ X_{32} \ X_{33}]^T \quad (2-13)$$

[00100] The non-separable transform is calculated as $\vec{F} = T \cdot \vec{X}$, where $\vec{F}$ indicates the transform coefficient vector, and T is a 16x16 transform matrix. The 16x1 coefficient vector $\vec{F}$ is subsequently re-organized as 4x4 block using the scanning order for that block (horizontal, vertical or diagonal). The coefficients with smaller index will be placed with the smaller scanning index in the 4x4 coefficient block.

2.11.1. Reduced Non-separable transform

[00101] LFNST (low-frequency non-separable transform) is based on direct matrix multiplication approach to apply non-separable transform so that it is implemented in a single pass without multiple iterations. However, the non-separable transform matrix dimension needs to be reduced to minimize computational complexity and memory space to store the transform coefficients. Hence, reduced non-separable transform (or RST) method is used in LFNST. The main idea of the reduced non-separable transform is to map an $\mathbf{N}$ ($\mathbf{N}$ is commonly equal to 64 for 8x8 NSST) dimensional vector to an $\mathbf{R}$ dimensional vector in a different space, where $\mathbf{N}/\mathbf{R}$ ($\mathbf{R} < \mathbf{N}$) is the reduction factor. Hence, instead of $\mathbf{N} \times \mathbf{N}$ matrix, RST matrix becomes an $\mathbf{R} \times \mathbf{N}$ matrix as follows:

$$T_{R \times N} = \begin{bmatrix} t_{11} & t_{12} & t_{13} & \dots & t_{1N} \\ t_{21} & t_{22} & t_{23} & & t_{2N} \\ & \vdots & & \ddots & \vdots \\ t_{R1} & t_{R2} & t_{R3} & \dots & t_{RN} \end{bmatrix} \quad (2-14)$$

[00102] where the $\mathbf{R}$ rows of the transform are $\mathbf{R}$ bases of the $\mathbf{N}$ dimensional space. The inverse transform matrix for RT is the transpose of its forward transform. For 8x8 LFNST, a reduction factor of 4 is applied, and 64x64 direct matrix, which is conventional 8x8 non-separable transform matrix size, is reduced to 16x48 direct matrix. Hence, the 48x16 inverse RST matrix is used at the decoder side to generate core (primary) transform coefficients in 8x8 top-left regions. When 16x48 matrices are applied instead of 16x64 with the same transform set configuration, each of which takes 48 input data from three 4x4 blocks in a top-left 8x8 block excluding right-bottom 4x4 block. With the help of the reduced dimension, memory usage for storing all LFNST matrices is reduced from 10KB to 8KB with reasonable performance drop. In order to reduce complexity LFNST is restricted to be applicable only if all coefficients outside the first coefficient sub-group are non-significant. Hence, all primary-only transform coefficients have to be zero when LFNST is applied. This allows a conditioning of the LFNST index signalling on the last-significant position, and hence avoids the extra coefficient scanning in the current LFNST design, which is needed for checking for significant coefficients at specific positions only. The worst-case handling of LFNST (in terms of multiplications per pixel) restricts the non-separable transforms for 4x4 and 8x8 blocks to 8x16 and 8x48 transforms, respectively. In those cases, the last-significant scan position has to be less than 8 when LFNST is applied, for other sizes less than 16. For blocks with a shape of 4xN and Nx4 and $N > 8$, the proposed restriction implies that the LFNST is now applied only once, and that to the top-left 4x4 region only. As all primary-only coefficients are zero when LFNST is applied, the number of operations needed for the primary transforms is reduced in such cases. From encoder perspective, the quantization of coefficients is remarkably simplified when LFNST transforms are tested. A rate-distortion optimized quantization has to be done at maximum for the first 16 coefficients (in scan order), the remaining coefficients are enforced to be zero.

2.11.2. LFNST transform selection

[00103] There are totally 4 transform sets and 2 non-separable transform matrices (kernels) per transform set are used in LFNST. The mapping from the intra prediction mode to the transform

set is pre-defined as shown in Table 2-6. If one of three CCLM modes (INTRA_LT_CCLM, INTRA_T_CCLM or INTRA_L_CCLM) is used for the current block ($81 \leq \text{predModeIntra} \leq 83$), transform set 0 is selected for the current chroma block. For each transform set, the selected non-separable secondary transform candidate is further specified by the explicitly signalled LFNST index. The index is signalled in a bit-stream once per Intra CU after transform coefficients.

Table 2-6. Transform selection table

IntraPredMode	Tr. set index
$\text{IntraPredMode} < 0$	1
$0 \leq \text{IntraPredMode} \leq 1$	0
$2 \leq \text{IntraPredMode} \leq 12$	1
$13 \leq \text{IntraPredMode} \leq 23$	2
$24 \leq \text{IntraPredMode} \leq 44$	3
$45 \leq \text{IntraPredMode} \leq 55$	2
$56 \leq \text{IntraPredMode} \leq 80$	1
$81 \leq \text{IntraPredMode} \leq 83$	0

2.11.3. LFNST index Signalling and interaction with other tools

[00104] Since LFNST is restricted to be applicable only if all coefficients outside the first coefficient sub-group are non-significant, LFNST index coding depends on the position of the last significant coefficient. In addition, the LFNST index is context coded but does not depend on intra prediction mode, and only the first bin is context coded. Furthermore, LFNST is applied for intra CU in both intra and inter slices, and for both Luma and Chroma. If a dual tree is enabled, LFNST indices for Luma and Chroma are signalled separately. For inter slice (the dual tree is disabled), a single LFNST index is signalled and used for both Luma and Chroma.

[00105] When ISP mode is selected, LFNST is disabled and RST index is not signalled, because performance improvement was marginal even if RST is applied to every feasible partition block. Furthermore, disabling RST for ISP-predicted residual could reduce encoding complexity.

LFNST is also disabled and the index is not signalled when MIP mode is selected.

[00106] Considering that a large CU greater than 64x64 is implicitly split (TU tiling) due to the existing maximum transform size restriction (64x64), an LFNST index search could increase data buffering by four times for a certain number of decode pipeline stages. Therefore, the maximum size that LFNST is allowed is restricted to 64x64. Note that LFNST is enabled with DCT2 only.

2.12. Transform skip for chroma

[00107] Chroma transform skip (TS) is introduced in VVC. The motivation is to unify TS and MTS signalling between luma and chroma by relocating transform_skip_flag and mts_idx into residual_coding part. One context model is added for chroma TS. No context model and no binarization are changed for the mts_idx. In addition, TS residual coding is also applied when chroma TS is used.

Semantics

[00108] transform_skip_flag[x0][y0][cIdx] specifies whether a transform is applied to the associated ~~luma~~ transform block or not. The array indices x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered transform block relative to the top-left luma sample of the picture. transform_skip_flag[x0][y0][cIdx] equal to 1 specifies that no transform is applied to the current ~~luma~~ transform block. The array index cIdx specifies an indicator for the colour component; it is equal to 0 for luma, equal to 1 for Cb and equal to 2 for Cr. transform_skip_flag[x0][y0][cIdx] equal to 0 specifies that the decision whether transform is applied to the current transform block or not depends on other syntax elements. When transform_skip_flag[x0][y0][cIdx] is not present, it is inferred to be equal to 0.

2.13. BDPCM for chroma

[00109] In addition to chroma TS support, BDPCM is added to chroma components. If sps_bdpcm_enable_flag is 1, a further syntax element **sps_bdpcm_chroma_enable_flag** is added to the SPS. The flags have the following behaviour, as indicated in Table 2-7.

Table 2-7. sps flags for luma and chroma BDPCM

sps_bdpcm_enable_flag	sps_bdpcm_chroma_enable_flag	behaviour
0	not written	BDPCM is not used in the sequence
1	0	BDPCM is available for luma only
1	1	BDPCM is available for luma and chroma

[00110] When BDPCM is available for luma only, the current behaviour is unchanged. When BDPCM is also available for chroma, a **bdpcm_chroma_flag** is sent for each chroma block. This indicates whether BDPCM is used on the chroma blocks. When it is on, BDPCM is used for

both chroma components, and an additional **bdpcm_dir_chroma** flag is coded, indicating the prediction direction used for both chroma components.

[00111] The deblocking filter is de-activated on a border between two Block-DPCM blocks, since neither of the blocks uses the transform stage usually responsible for blocking artifacts. This deactivation happens independently for luma and chroma components.

3. Examples of technical problems solved by the disclosed solutions

[00112] The current design of derivation of linear parameters in CCLM and TS has the following problems:

1. For the non-4:4:4 color format, the derivation of linear parameters in CCLM involves neighbouring chroma samples and down-sampled collocated neighbouring luma samples. As shown in FIG. 8, in current VVC, when the nearest line is not at the CTU boundary, the downsampled collocated neighbouring top luma samples are derived using the second line above current block for 4:2:2 videos. However, for the 4:2:2 videos, the vertical resolution is unchanged. Therefore, there is phase shift between the downsampled collocated neighbouring top luma samples and the neighbouring chroma samples.

[00113] FIG. 8 shows an example of neighbouring chroma samples and downsampled collocated neighbouring luma samples used in the derivation of CCLM parameters for 4:2:2 videos.

2. In current VVC, the same maximum block size is used in the condition check for signalling of luma transform skip flag and signalling of chroma transform skip flag. Such a design doesn't take the color format into consideration which is not desirable.
 - a. similar problem also exists for signalling of luma BDPCM flag and signalling of chroma BDPCM flag wherein the same maximum block size is used in the condition check.

4. A listing of embodiments and techniques

[00114] The listing below should be considered as examples to explain general concepts. These items should not be interpreted in a narrow way. Furthermore, these items can be combined in any manner.

[00115] In this document, the term 'CCLM' represents a coding tool that utilizes cross-color component information to predict samples/residuals for current color component or to derive

reconstruction of samples in current color component. It is not limited to the CCLM technologies described in VVC.

Derivation of linear parameters in CCLM

1. When deriving the CCLM parameters for a chroma block, one or multiple above neighboring lines of its collocated luma block may be used to derive its downsampled collocated neighbouring top luma samples.
 - a. In one example, when the current chroma block is not at the top CTU boundary, the nearest above line of the collocated luma block, instead of the second line above, may be used for derivation of the downsampled collocated neighbouring top luma samples.
 - i. In one example, one same downsampling filter may be used for deriving the downsampled collocated neighbouring top luma samples and the downsampled collocated neighbouring left luma samples.
 - 1) For example, $[1 \ 2 \ 1]$ filter may be used. More specifically, $pDsY[x] = (pY[2*x - 1][-1] + 2 * pY[2 * x][-1] + pY[2 * x + 1][-1] + 2) >> 2$, wherein $pY[2*x][-1]$, $pY[2*x - 1][-1]$, $pY[2*x + 1][-1]$ are luma samples from the nearest above neighboring line and $pDstY[x]$ represents the downsampled collocated neighbouring top luma samples.
 - ii. In one example, different downsampling filters (e.g., different filter taps/different filter coefficients) may be used for deriving the downsampled collocated neighbouring top luma samples and the downsampled collocated neighbouring left luma samples.
 - iii. In one example, one same downsampling filter may be used for deriving the downsampled collocated neighbouring top luma samples regardless of the position of the chroma block (e.g., the chroma block may be or may not be at a top CTU boundary).
 - iv. In one example, the above methods may be only applied to images/videos in 4:2:2 format.
 - b. In one example, when the current chroma block is not at the top CTU boundary, above neighbouring luma samples, including the nearest above line of the

collocated luma block, but excluding the second line above, may be used for derivation of the downsampled collocated neighbouring top luma samples.

c. In one example, the derivation of the downsampled collocated neighbouring top luma samples may depend on samples located at multiple lines.

i. In one example, it may depend on both the second nearest line and the nearest line above the collocated luma block.

ii. In one example, the downsampled collocated neighbouring top luma samples may be derived using one same downsampling filter for different colour formats (e.g. 4:2:0 and 4:2:2).

1) In one example, 6-tap filter (e.g., [1 2 1; 1 2 1]) may be utilized.

a) In one example, the downsampled collocated neighbouring top luma samples may be derived as:

$$pDsY[x] = (pY[2*x - 1][-2] + 2 * pY[2 * x][-2] + pY[2 * x + 1][-2] + pY[2*x - 1][-1] + 2 * pY[2 * x][-1] + pY[2 * x + 1][-1] + 4) >> 3$$
 wherein pY are corresponding luma samples and pDstY[x] represents the downsampled collocated neighbouring top luma samples.

b) Alternatively, furthermore, the above method may be applied when sps_cclm_collocated_chroma_flag is equal to 0.

2) In one example, 5-tap filter (e.g., [0 1 0; 1 4 1; 0 1 0]) may be utilized.

a) In one example, the downsampled collocated neighbouring top luma samples may be derived as:

$$pDsY[x] = (pY[2 * x][-2] + pY[2 * x - 1][-1] + 4 * pY[2 * x][-1] + pY[2 * x + 1][-1] + pY[2 * x][0] + 4) >> 3$$
 wherein pY are corresponding luma samples and pDstY[x] represents the downsampled collocated neighbouring top luma samples.

- b) Alternatively, furthermore, the above method may be applied when `sps_cclm_colocated_chroma_flag` is equal to 1.
- iii. In one example, the above methods may be only applied to images/videos in 4:2:2 format.

Maximum block sizes of transform skip coded blocks (e.g., with **`transform_skip_flag`** equal to 1, or BDPCM or other modes that bypass transform process/use identity transform)

2. Maximum block size of transform skip coded blocks may be dependent on the colour component. Denote the maximum block size of transform skip coded blocks for luma and chroma by `MaxTsSizeY` and `MaxTsSizeC`, respectively.
 - a. In one example, maximum block sizes for luma and chroma components may be different.
 - b. In one example, maximum block sizes for two chroma components may be different.
 - c. In one example, maximum block sizes for luma and chroma components or for each colour component may be signalled separately.
 - i. In one example, `MaxTsSizeC/MaxTsSizeY` may be signalled at sequence level/picture level/slice level/tile group level, such as in sequence header/picture header/SPS/VPS/DPS/PPS/APS/slice header/tile group header.
 - ii. In one example, the `MaxTsSizeY` may be conditionally signalled, such as according to transform skip is enabled or not/BDPCM is enabled or not.
 - iii. In one example, the `MaxTsSizeC` may be conditionally signalled, such as according to colour format/transform skip is enabled or not/BDPCM is enabled or not.
 - iv. Alternatively, predictive coding between maximum block sizes for luma and chroma components may be utilized.
 - d. In one example, `MaxTsSizeC` may depend on `MaxTsSizeY`.
 - i. In one example, `MaxTsSizeC` may be set equal to `MaxTsSizeY`.
 - ii. In one example, `MaxTsSizeC` may be set equal to `MaxTsSizeY/N` (N is an integer). For example, $N = 2$.

- e. In one example, MaxTsSizeC may be set according to the chroma subsampling ratios.
 - i. In one example, MaxTsSizeC is set equal to $\text{MaxTsSizeY} \gg \text{SubWidthC}$, wherein SubWidthC is defined in Table 2-1.
 - ii. In one example, MaxTsSizeC is set equal to $\text{MaxTsSizeY} \gg \text{SubHeightC}$, wherein SubHeightC is defined in Table 2-1.
 - iii. In one example, MaxTsSizeC is set equal to $\text{MaxTsSizeY} \gg \max(\text{SubWidthC}, \text{SubHeightC})$.
 - iv. In one example, MaxTsSizeC is set equal to $\text{MaxTsSizeY} \gg \min(\text{SubWidthC}, \text{SubHeightC})$.
- 3. Maximum allowed block size width and height for a transform coded block may be defined differently.
 - a. In one example, the maximum allowed block size width and height may be signalled separately.
 - b. In one example, the maximum allowed block size width and height for a chroma transform coded block may be denoted as MaxTsSizeWC and MaxTsSizeHC , respectively. MaxTsSizeWC may be set equal to $\text{MaxTsSizeY} \gg \text{SubWidthC}$ and MaxTsSizeHC may be set equal to $\text{MaxTsSizeY} \gg \text{SubHeightC}$.
 - i. In one example, the MaxTsSizeY is the one defined in bullet 2.
- 4. Whether to signal a transform skip flag for a chroma block (e.g., **transform_skip_flag**[x0][y0][1] and/or **transform_skip_flag**[x0][y0][2]) may depend on the maximum allowed size for chroma transform skip coded blocks.
 - a. In one example, the chroma transform skip flag may be conditionally signalled according to the following conditions:
 - i. In one example, the conditions are: tbW is less than or equal to MaxTsSizeC and tbH is less than or equal to MaxTsSizeC , wherein tbW and tbH are the width and height of the current chroma block.
 - 1) In one example, MaxTsSizeC may be defined as that in bullets 2-3.
 - ii. In one example, the conditions are: tbW is less than or equal to MaxTsSizeWC and tbH is less than or equal to MaxTsSizeHC , wherein

tbW and tbH are the width and height of the current chroma block, MaxTsSizeWC and MaxTsSizeHC represent the maximum allowed block size width and height, respectively, for chroma transform skip coded blocks.

- 1) In one example, MaxTsSizeWC and/or MaxTsSizeHC may be defined as that in bullet 3.
 - b. In one example, the above methods may be applicable to the coding of chroma BDPCM flags (e.g., **intra_bdpcm_chroma_flag**) by replacing 'transform skip' by 'BDPCM'.
5. Instead of coding two TS flags for two chroma color component, it is proposed to use one syntax to indicate the usage of TS for the two chroma color components.
- a. In one example, instead of coding **transform_skip_flag**[x0][y0][1] and/or **transform_skip_flag**[x0][y0][2]), a single syntax element (e.g., TS_chroma_flag) may be coded.
 - i. In one example, the value of the single syntax element is a binary value.
 - 1) Alternatively, furthermore, the two chroma component blocks share the same on/off control of TS mode according to the single syntax element.
 - a) In one example, the value of the single syntax element equal to 0 indicates TS is disabled for both.
 - b) In one example, the value of the single syntax element equal to 0 indicates TS is enabled for both.
 - 2) Alternatively, furthermore, a second syntax element may be further singled based on whether the value of the single syntax element is equal to K (e.g., K = 1).
 - a) In one example, the value of the single syntax element equal to 0 indicates TS is disabled for both; the value of the single syntax element equal to 0 indicates TS is enabled for at least one of the two chroma component.

- b) The second syntax element may be used to indicate TS is applied to which one of the two chroma components and/or TS is applied to both of them.
 - ii. In one example, the value of the single syntax element is a non-binary value.
 - 1) In one example, the value of the single syntax element equal to K0 indicates TS is disabled for both
 - 2) In one example, the value of the single syntax element equal to K1 indicates TS is enabled for the first chroma color component and disabled for the second color component.
 - 3) In one example, the value of the single syntax element equal to K2 indicates TS is disabled for the first chroma color component and enabled for the second color component.
 - 4) In one example, the value of the single syntax element equal to K3 indicates TS is enabled for both.
 - 5) In one example, the single syntax element may be coded with fixed length, unary, truncated unary, k-th order EG binarization methods.
 - iii. In one example, the single syntax element and/or second syntax element may be context coded or bypass coded.

General claims

- 6. Whether to and/or how to apply the disclosed methods above may be signalled at sequence level/picture level/slice level/tile group level, such as in sequence header/picture header/SPS/VPS/DPS/PPS/APS/slice header/tile group header.
- 7. Whether to and/or how to apply the disclosed methods above may be dependent on coded information, such as color format, single/dual tree partitioning.

5. Embodiments

[00116] This section shows example embodiments and ways to modify the current VVC standard to describe these embodiments. The changes to the VVC specification are highlighted in

bold and Italic. Deleted texts are marked with double brackets (e.g., ~~[[a]]~~ denotes the deletion of the character “a”).

5.1. Embodiment 1

[00117] The working draft specified in JVET-P2001-v9 may be changed as below.

8.4.5.2.13 Specification of INTRA_LT_CCLM, INTRA_L_CCLM and INTRA_T_CCLM intra prediction mode

...

3. The down-sampled collocated luma samples $pDsY[x][y]$ with $x = 0..nTbW - 1$, $y = 0..nTbH - 1$ are derived as follows:

- If both SubWidthC and SubHeightC are equal to 1, the following applies:
 - $pDsY[x][y]$ with $x = 1..nTbW - 1$, $y = 1..nTbH - 1$ is derived as follows:

$$pDstY[x][y] = pY[x][y] \quad (8-159)$$
 - Otherwise, the following applies:
 - The one-dimensional filter coefficients array F1 and F2, and the 2-dimensional filter coefficients arrays F3 and F4 are specified as follows.

$$F1[i] = 1, \text{ with } i = 0..1 \quad (8-160)$$

$$F2[0] = 1, F2[1] = 2, F2[2] = 1 \quad (8-161)$$

$$F3[i][j] = F4[i][j] = 0, \text{ with } i = 0..2, j = 0..2 \quad (8-162)$$
 - If both SubWidthC and SubHeightC are equal to 2, the following applies:

$$F1[0] = 1, F1[1] = 1 \quad (8-163)$$

$$F3[0][1] = 1, F3[1][1] = 4, F3[2][1] = 1, F3[1][0] = 1, F3[1][2] = 1 \quad (8-164)$$

$$F4[0][1] = 1, F4[1][1] = 2, F4[2][1] = 1 \quad (8-165)$$

$$F4[0][2] = 1, F4[1][2] = 2, F4[2][2] = 1 \quad (8-166)$$
 - Otherwise, the following applies:

$$F1[0] = 2, F1[1] = 0 \quad (8-167)$$

$$F3[1][1] = 8 \quad (8-168)$$

$$F4[0][1] = 2, F4[1][1] = 4, F4[2][1] = 2, \quad (8-169)$$

...

5. When numSampT is greater than 0, the selected 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 $idx = 0..cntL + cntT - 1$ are specified as follows:

...

- Otherwise (sps_cclm_collocated_chroma_flag is equal to 0), the following applies:
 - If x is greater than 0, the following applies:
 - If bCTUboundary is equal to FALSE, the following applies:

$$\begin{aligned}
pSelDsY[idx] = & (F4[0][1] * pY[SubWidthC * x - 1][[-2]] - 1] + \\
& F4[0][2] * pY[SubWidthC * x - 1][[-1]] - 2] + \\
& F4[1][1] * pY[SubWidthC * x][[-2]] - 1] + \quad (8-193) \\
& F4[1][2] * pY[SubWidthC * x][[-1]] - 2] + \\
& F4[2][1] * pY[SubWidthC * x + 1][[-2]] - 1] + \\
& F4[2][2] * pY[SubWidthC * x + 1][[-1]] - 2] + 4) >> 3
\end{aligned}$$

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

$$\begin{aligned}
pSelDsY[idx] = & (F2[0] * pY[SubWidthC * x - 1][-1] + \\
& F2[1] * pY[SubWidthC * x][-1] + \quad (8-194) \\
& F2[2] * pY[SubWidthC * x + 1][-1] + 2) >> 2
\end{aligned}$$

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

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

$$\begin{aligned}
pSelDsY[idx] = & (F4[0][1] * pY[-1][[-2]] - 1] + F4[0][2] * pY[-1][[-1]] - 2] + \\
& F4[1][1] * pY[0][[-2]] - 1] + F4[1][2] * pY[0][[-1]] - 2] + \quad (8-195) \\
& F4[2][1] * pY[1][[-2]] - 1] + F4[2][2] * pY[1][[-1]] - 2] + 4) >> 3
\end{aligned}$$

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

$$\begin{aligned}
pSelDsY[idx] = & (F2[0] * pY[-1][-1] + F2[1] * \quad pY[0][-1] + \\
& F2[2] * pY[1][-1] + 2) >> 2 \quad (8-196)
\end{aligned}$$

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

$$pSelDsY[idx] = (F1[1] * pY[0][-2] + F1[0] * pY[0][-1] + 1) >> 1 \quad (8-197)$$

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

$$pSelDsY[idx] = pY[0][-1] \quad (8-198)$$

...

5.2. Embodiment 2

[00118] This embodiment shows an example on chroma transform skip flag coding according to maximum allowed transform skip coded block sizes. The working draft specified in JVET-P2001-v9 may be changed as below.

7.3.9.10 Transform unit syntax

...

if(tu_cbf_luma[x0][y0] && treeType != DUAL_TREE_CHROMA) {	
if(sps_transform_skip_enabled_flag && !BdpcmFlag[x0][y0][0] &&.. tbWidth <= MaxTsSize && tbHeight <= MaxTsSize && (IntraSubPartitionsSplit[x0][y0] == ISP_NO_SPLIT) && !cu_sbt_flag)	
transform_skip_flag[x0][y0][0]	ae(v)
if(!transform_skip_flag[x0][y0][0])	
residual_coding(x0, y0, Log2(tbWidth), Log2(tbHeight), 0)	
else	
residual_ts_coding(x0, y0, Log2(tbWidth), Log2(tbHeight), 0)	
}	
if(tu_cbf_cb[x0][y0] && treeType != DUAL_TREE_LUMA)	
if(sps_transform_skip_enabled_flag && !BdpcmFlag[x0][y0][1] &&.. wC <= (MaxTsSize >> <i>subWidthC</i>) && hC <= (MaxTsSize >> <i>subHeightC</i>) && !cu_sbt_flag)	
transform_skip_flag[x0][y0][1]	ae(v)
if(!transform_skip_flag[x0][y0][1])	
residual_coding(xC, yC, Log2(wC), Log2(hC), 1)	
else	
residual_ts_coding(xC, yC, Log2(wC), Log2(hC), 1)	
if(tu_cbf_cr[x0][y0] && treeType != DUAL_TREE_LUMA &&.. !(tu_cbf_cb[x0][y0] && tu_joint_cbr_residual_flag[x0][y0])) {	
if(sps_transform_skip_enabled_flag && !BdpcmFlag[x0][y0][2] &&.. wC <= (MaxTsSize >> <i>subWidthC</i>) && hC <= (MaxTsSize >> <i>subHeightC</i>) && !cu_sbt_flag)	
transform_skip_flag[x0][y0][2]	ae(v)
if(!transform_skip_flag[x0][y0][2])	
residual_coding(xC, yC, Log2(wC), Log2(hC), 2)	
else	
residual_ts_coding(xC, yC, Log2(wC), Log2(hC), 2)	
}	

...

5.3. Embodiment 3

[00119] This embodiment shows an example on chroma BDPCM flag coding according to maximum allowed chroma transform skip coded block sizes. The working draft specified in JVET-P2001-v9 may be changed as below.

7.3.9.5 Coding unit syntax

...

if((treeType == SINGLE_TREE treeType == DUAL_TREE_CHROMA) && ChromaArrayType != 0) {	
if(pred_mode_plt_flag && treeType == DUAL_TREE_CHROMA)	
palette_coding(x0, y0, cbWidth/ SubWidthC, cbHeight/ SubHeightC, 1, 2)	
else {	

if(!cu_act_enabled_flag) {	
if(cbWidth <= (MaxTsSize>>subWidthC) && cbHeight <= (MaxTsSize>>subHeightC) && sps_bdpcm_chroma_enabled_flag) {	
intra_bdpcm_chroma_flag	ae(v)
if(intra_bdpcm_chroma_flag)	
intra_bdpcm_chroma_dir_flag	ae(v)
} else {	
if(CclmEnabled)	
cclm_mode_flag	ae(v)
if(cclm_mode_flag)	
cclm_mode_idx	ae(v)
else	
intra_chroma_pred_mode	ae(v)
}	
}	
}	

...

[00120] FIG. 9 is a block diagram of a video processing apparatus 900. The apparatus 900 may be used to implement one or more of the methods described herein. The apparatus 900 may be embodied in a smartphone, tablet, computer, Internet of Things (IoT) receiver, and so on. The apparatus 900 may include one or more processors 902, one or more memories 904 and video processing hardware 906. The processor(s) 902 may be configured to implement one or more methods described in the present document. The memory (memories) 904 may be used for storing data and code used for implementing the methods and techniques described herein. The video processing hardware 906 may be used to implement, in hardware circuitry, some techniques described in the present document (e.g., listed in the previous section). In some embodiments, the hardware 906 may be partly or entirely within the processors 902, such as a graphics processor.

[00121] FIG. 10 shows block diagram of a video encoder.

[00122] FIG. 11 is a flowchart for a method 1100 of processing a video. The method 1100 includes deriving (1102), for a conversion between a chroma block of a video and a coded representation of the video, parameters of a cross-component linear model by using downsampled collocated neighboring top luma samples that are generated from N above neighboring lines of a collocated luma block using a downsampling filter, where N is a positive

integer, and performing (1104) the conversion using a predicted chroma block generated using the cross-component linear model.

[00123] FIG. 12 is a block diagram of an example video processing system in which disclosed techniques may be implemented.

[00124] 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.

[00125] 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.

[00126] 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.

[00127] Some embodiments of the disclosed technology include making a decision or determination to enable a video processing tool or mode. In an example, when the video processing tool or mode is enabled, the encoder will use or implement the tool or mode in the processing of a block of video, but may not necessarily modify the resulting bitstream based on the usage of the tool or mode. That is, a conversion from the block of video to the bitstream representation of the video will use the video processing tool or mode when it is enabled based on the decision or determination. In another example, when the video processing tool or mode is enabled, the decoder will process the bitstream with the knowledge that the bitstream has been modified based on the video processing tool or mode. That is, a conversion from the bitstream representation of the video to the block of video will be performed using the video processing tool or mode that was enabled based on the decision or determination.

[00128] Some embodiments of the disclosed technology include making a decision or determination to disable a video processing tool or mode. In an example, when the video processing tool or mode is disabled, the encoder will not use the tool or mode in the conversion of the block of video to the bitstream representation of the video. In another example, when the video processing tool or mode is disabled, the decoder will process the bitstream with the knowledge that the bitstream has not been modified using the video processing tool or mode that was enabled based on the decision or determination.

[00129] The disclosed and other solutions, examples, embodiments, modules and the functional operations described in this document can be implemented in digital electronic circuitry, or in computer software, firmware, or hardware, including the structures disclosed in this document and their structural equivalents, or in combinations of one or more of them. The disclosed and other embodiments can be implemented as one or more computer program products, i.e., one or more modules of computer program instructions encoded on a 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 them. The term “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. A propagated signal is an artificially generated signal, e.g., a machine-generated electrical, optical, or electromagnetic signal, that is generated to encode information for transmission to suitable receiver apparatus.

[00130] 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.

[00131] The processes and logic flows described in this document 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).

[00132] 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 non-volatile memory, media and memory devices, including by way of example semiconductor memory devices, e.g., EPROM, EEPROM, and flash memory devices; magnetic disks, e.g., internal hard disks or removable disks; magneto optical disks; and CD ROM and DVD-ROM disks. The processor and the memory can be supplemented by, or incorporated in, special purpose logic circuitry.

[00133] FIG. 13 is a block diagram that illustrates an example video coding system 100 that may utilize the techniques of this disclosure.

[00134] As shown in FIG. 13, video coding system 100 may include a source device 110 and a destination device 120. Source device 110 generates encoded video data which may be referred to as a video encoding device. Destination device 120 may decode the encoded video data generated by source device 110 which may be referred to as a video decoding device.

[00135] Source device 110 may include a video source 112, a video encoder 114, and an input/output (I/O) interface 116.

[00136] Video source 112 may include a source such as a video capture device, an interface to receive video data from a video content provider, and/or a computer graphics system for generating video data, or a combination of such sources. The video data may comprise one or more pictures. Video encoder 114 encodes the video data from video source 112 to generate a bitstream. The bitstream may include a sequence of bits that form a coded representation of the video data. The bitstream may include coded pictures and associated data. The coded picture is a coded representation of a picture. The associated data may include sequence parameter sets, picture parameter sets, and other syntax structures. I/O interface 116 may include a modulator/demodulator (modem) and/or a transmitter. The encoded video data may be transmitted directly to destination device 120 via I/O interface 116 through network 130a. The encoded video data may also be stored onto a storage medium/server 130b for access by destination device 120.

[00137] Destination device 120 may include an I/O interface 126, a video decoder 124, and a display device 122.

[00138] I/O interface 126 may include a receiver and/or a modem. I/O interface 126 may acquire encoded video data from the source device 110 or the storage medium/ server 130b. Video decoder 124 may decode the encoded video data. Display device 122 may display the decoded video data

to a user. Display device 122 may be integrated with the destination device 120, or may be external to destination device 120 which be configured to interface with an external display device.

[00139] Video encoder 114 and video decoder 124 may operate according to a video compression standard, such as the High Efficiency Video Coding (HEVC) standard, Versatile Video Coding (VVC) standard and other current and/or further standards.

[00140] FIG. 14 is a block diagram illustrating an example of video encoder 200, which may be video encoder 114 in the system 100 illustrated in FIG. 13.

[00141] Video encoder 200 may be configured to perform any or all of the techniques of this disclosure. In the example of FIG. 14, video encoder 200 includes a plurality of functional components. The techniques described in this disclosure may be shared among the various components of video encoder 200. In some examples, a processor may be configured to perform any or all of the techniques described in this disclosure.

[00142] The functional components of video encoder 200 may include a partition unit 201, a predication unit 202 which may include a mode select unit 203, a motion estimation unit 204, a motion compensation unit 205 and an intra prediction unit 206, a residual generation unit 207, a transform unit 208, a quantization unit 209, an inverse quantization unit 210, an inverse transform unit 211, a reconstruction unit 212, a buffer 213, and an entropy encoding unit 214.

[00143] In other examples, video encoder 200 may include more, fewer, or different functional components. In an example, predication unit 202 may include an intra block copy (IBC) unit. The IBC unit may perform predication in an IBC mode in which at least one reference picture is a picture where the current video block is located.

[00144] Furthermore, some components, such as motion estimation unit 204 and motion compensation unit 205 may be highly integrated, but are represented in the example of FIG. 14 separately for purposes of explanation.

[00145] Partition unit 201 may partition a picture into one or more video blocks. Video encoder 200 and video decoder 300 may support various video block sizes.

[00146] Mode select unit 203 may select one of the coding modes, intra or inter, e.g., based on error results, and provide the resulting intra- or inter-coded block to a residual generation unit 207 to generate residual block data and to a reconstruction unit 212 to reconstruct the encoded block for use as a reference picture. In some example, Mode select unit 203 may select a combination of intra and inter predication (CIIP) mode in which the predication is based on an inter predication

signal and an intra predication signal. Mode select unit 203 may also select a resolution for a motion vector (e.g., a sub-pixel or integer pixel precision) for the block in the case of inter-predication.

[00147] To perform inter prediction on a current video block, motion estimation unit 204 may generate motion information for the current video block by comparing one or more reference frames from buffer 213 to the current video block. Motion compensation unit 205 may determine a predicted video block for the current video block based on the motion information and decoded samples of pictures from buffer 213 other than the picture associated with the current video block.

[00148] Motion estimation unit 204 and motion compensation unit 205 may perform different operations for a current video block, for example, depending on whether the current video block is in an I slice, a P slice, or a B slice.

[00149] In some examples, motion estimation unit 204 may perform uni-directional prediction for the current video block, and motion estimation unit 204 may search reference pictures of list 0 or list 1 for a reference video block for the current video block. Motion estimation unit 204 may then generate a reference index that indicates the reference picture in list 0 or list 1 that contains the reference video block and a motion vector that indicates a spatial displacement between the current video block and the reference video block. Motion estimation unit 204 may output the reference index, a prediction direction indicator, and the motion vector as the motion information of the current video block. Motion compensation unit 205 may generate the predicted video block of the current block based on the reference video block indicated by the motion information of the current video block.

[00150] In other examples, motion estimation unit 204 may perform bi-directional prediction for the current video block, motion estimation unit 204 may search the reference pictures in list 0 for a reference video block for the current video block and may also search the reference pictures in list 1 for another reference video block for the current video block. Motion estimation unit 204 may then generate reference indexes that indicate the reference pictures in list 0 and list 1 containing the reference video blocks and motion vectors that indicate spatial displacements between the reference video blocks and the current video block. Motion estimation unit 204 may output the reference indexes and the motion vectors of the current video block as the motion information of the current video block. Motion compensation unit 205 may generate the predicted

video block of the current video block based on the reference video blocks indicated by the motion information of the current video block.

[00151] In some examples, motion estimation unit 204 may output a full set of motion information for decoding processing of a decoder.

[00152] In some examples, motion estimation unit 204 may do not output a full set of motion information for the current video. Rather, motion estimation unit 204 may signal the motion information of the current video block with reference to the motion information of another video block. For example, motion estimation unit 204 may determine that the motion information of the current video block is sufficiently similar to the motion information of a neighboring video block.

[00153] In one example, motion estimation unit 204 may indicate, in a syntax structure associated with the current video block, a value that indicates to the video decoder 300 that the current video block has the same motion information as the another video block.

[00154] In another example, motion estimation unit 204 may identify, in a syntax structure associated with the current video block, another video block and a motion vector difference (MVD). The motion vector difference indicates a difference between the motion vector of the current video block and the motion vector of the indicated video block. The video decoder 300 may use the motion vector of the indicated video block and the motion vector difference to determine the motion vector of the current video block.

[00155] As discussed above, video encoder 200 may predictively signal the motion vector. Two examples of predictive signaling techniques that may be implemented by video encoder 200 include advanced motion vector predication (AMVP) and merge mode signaling.

[00156] Intra prediction unit 206 may perform intra prediction on the current video block. When intra prediction unit 206 performs intra prediction on the current video block, intra prediction unit 206 may generate prediction data for the current video block based on decoded samples of other video blocks in the same picture. The prediction data for the current video block may include a predicted video block and various syntax elements.

[00157] Residual generation unit 207 may generate residual data for the current video block by subtracting (e.g., indicated by the minus sign) the predicted video block(s) of the current video block from the current video block. The residual data of the current video block may include residual video blocks that correspond to different sample components of the samples in the current video block.

[00158] In other examples, there may be no residual data for the current video block for the current video block, for example in a skip mode, and residual generation unit 207 may not perform the subtracting operation.

[00159] Transform processing unit 208 may generate one or more transform coefficient video blocks for the current video block by applying one or more transforms to a residual video block associated with the current video block.

[00160] After transform processing unit 208 generates a transform coefficient video block associated with the current video block, quantization unit 209 may quantize the transform coefficient video block associated with the current video block based on one or more quantization parameter (QP) values associated with the current video block.

[00161] Inverse quantization unit 210 and inverse transform unit 211 may apply inverse quantization and inverse transforms to the transform coefficient video block, respectively, to reconstruct a residual video block from the transform coefficient video block. Reconstruction unit 212 may add the reconstructed residual video block to corresponding samples from one or more predicted video blocks generated by the predication unit 202 to produce a reconstructed video block associated with the current block for storage in the buffer 213.

[00162] After reconstruction unit 212 reconstructs the video block, loop filtering operation may be performed reduce video blocking artifacts in the video block.

[00163] Entropy encoding unit 214 may receive data from other functional components of the video encoder 200. When entropy encoding unit 214 receives the data, entropy encoding unit 214 may perform one or more entropy encoding operations to generate entropy encoded data and output a bitstream that includes the entropy encoded data.

[00164] FIG. 15 is a block diagram illustrating an example of video decoder 300 which may be video decoder 114 in the system 100 illustrated in FIG. 13.

[00165] The video decoder 300 may be configured to perform any or all of the techniques of this disclosure. In the example of FIG. 15, the video decoder 300 includes a plurality of functional components. The techniques described in this disclosure may be shared among the various components of the video decoder 300. In some examples, a processor may be configured to perform any or all of the techniques described in this disclosure.

[00166] In the example of FIG. 15, video decoder 300 includes an entropy decoding unit 301, a motion compensation unit 302, an intra prediction unit 303, an inverse quantization unit 304, an

inverse transformation unit 305, and a reconstruction unit 306 and a buffer 307. Video decoder 300 may, in some examples, perform a decoding pass generally reciprocal to the encoding pass described with respect to video encoder 200 (e.g., FIG. 14).

[00167] Entropy decoding unit 301 may retrieve an encoded bitstream. The encoded bitstream may include entropy coded video data (e.g., encoded blocks of video data). Entropy decoding unit 301 may decode the entropy coded video data, and from the entropy decoded video data, motion compensation unit 302 may determine motion information including motion vectors, motion vector precision, reference picture list indexes, and other motion information. Motion compensation unit 302 may, for example, determine such information by performing the AMVP and merge mode.

[00168] Motion compensation unit 302 may produce motion compensated blocks, possibly performing interpolation based on interpolation filters. Identifiers for interpolation filters to be used with sub-pixel precision may be included in the syntax elements.

[00169] Motion compensation unit 302 may use interpolation filters as used by video encoder 200 during encoding of the video block to calculate interpolated values for sub-integer pixels of a reference block. Motion compensation unit 302 may determine the interpolation filters used by video encoder 200 according to received syntax information and use the interpolation filters to produce predictive blocks.

[00170] Motion compensation unit 302 may use some of the syntax information to determine sizes of blocks used to encode frame(s) and/or slice(s) of the encoded video sequence, partition information that describes how each macroblock of a picture of the encoded video sequence is partitioned, modes indicating how each partition is encoded, one or more reference frames (and reference frame lists) for each inter-encoded block, and other information to decode the encoded video sequence.

[00171] Intra prediction unit 303 may use intra prediction modes for example received in the bitstream to form a prediction block from spatially adjacent blocks. Inverse quantization unit 303 inverse quantizes, i.e., de-quantizes, the quantized video block coefficients provided in the bitstream and decoded by entropy decoding unit 301. Inverse transform unit 303 applies an inverse transform.

[00172] Reconstruction unit 306 may sum the residual blocks with the corresponding prediction blocks generated by motion compensation unit 202 or intra-prediction unit 303 to form decoded

blocks. If desired, a deblocking filter may also be applied to filter the decoded blocks in order to remove blockiness artifacts. The decoded video blocks are then stored in buffer 307, which provides reference blocks for subsequent motion compensation.

[00173] Some embodiments of the disclosed technology include making a decision or determination to enable a video processing tool or mode. In an example, when the video processing tool or mode is enabled, the encoder will use or implement the tool or mode in the processing of a block of video, but may not necessarily modify the resulting bitstream based on the usage of the tool or mode. That is, a conversion from the block of video to the bitstream representation of the video will use the video processing tool or mode when it is enabled based on the decision or determination. In another example, when the video processing tool or mode is enabled, the decoder will process the bitstream with the knowledge that the bitstream has been modified based on the video processing tool or mode. That is, a conversion from the bitstream representation of the video to the block of video will be performed using the video processing tool or mode that was enabled based on the decision or determination.

[00174] In the present document, the term “video 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.

[00175] A listing of clauses preferred by some embodiments is provided next.

[00176] The first set of clauses show example embodiments of techniques discussed in the previous sections.

[00177] 1. A method of video processing, comprising: deriving, for a conversion between a chroma block of a video and a coded representation of the video, parameters of a cross-component linear model by using downsampled collocated neighboring top luma samples that are generated from N above neighboring lines of a collocated luma block using a downsampling filter, where N is a positive integer; and performing the conversion using a predicted chroma block generated using the cross-component linear model.

- [00178] 2. The method of clause 1, wherein, due to the chroma block not being at a top coding tree unit boundary, the N above neighboring lines correspond to a nearest above line of the collocated luma block.
- [00179] 3. The method of any of clauses 1-2, wherein the downsampling filter is also applied for generating downsampled collocated neighbouring left luma samples.
- [00180] 4. The method of any of clauses 1-2, wherein the downsampling filter is different from another downsampling filter used for generating downsampled collocated neighbouring left luma samples.
- [00181] 5. The method of any of clauses 1, wherein the downsampling filter is independent of a position of the chroma block with respect to a top boundary of the coding tree unit.
- [00182] 6. The method of any of clauses 1, wherein the method is selectively applied due to the video having a 4:2:2 format.
- [00183] 7. The method of clause 1, wherein N is greater than one.
- [00184] 8. The method of clause 7, wherein the N above neighboring lines include the nearest above line and a second nearest above line.
- [00185] 9. The method of clause 1, wherein the downsampling filter is dependent on a color format of the video.
- [00186] 10. The method of any of clauses 1-9, wherein the downsampling filter is a 6-tap filter.
- [00187] 11. The method of any of clauses 1-9, wherein the downsampling filter is a 5-tap filter.
- [00188] 12. The method of any of clauses 1 to 11, wherein the conversion comprises encoding the video into the coded representation.
- [00189] 13. The method of any of clauses 1 to 11, wherein the conversion comprises decoding the coded representation to generate pixel values of the video.
- [00190] 14. A video decoding apparatus comprising a processor configured to implement a method recited in one or more of clauses 1 to 13.
- [00191] 15. A video encoding apparatus comprising a processor configured to implement a method recited in one or more of clauses 1 to 13.
- [00192] 16. A computer program product having computer code stored thereon, the code, when executed by a processor, causes the processor to implement a method recited in any of clauses 1 to 13.
- [00193] 17. A method, apparatus or system described in the present document.

[00194] The second set of clauses describe certain features and aspects of the disclosed techniques in the previous section (e.g., item 1).

[00195] 1. A method of video processing, comprising: deriving 1602, for a conversion between a chroma block of a video and a bitstream representation of the video, parameters of a cross-component linear model by using downsampled luma samples that are generated from N above neighboring lines of a collocated luma block of the chroma block using a downsampling filter, where N is a positive integer; and performing the conversion using a predicted chroma block generated using the cross-component linear model.

[00196] 2. The method of clause 1, wherein, due to the chroma block not being at a top coding tree unit boundary, the N above neighboring lines correspond to a nearest above line of the collocated luma block.

[00197] 3. The method of any of clauses 1-2, wherein the downsampling filter is also applied for generating other downsampled luma samples that are generated from left neighboring lines of the collocated luma block.

[00198] 4. The method any of clauses 1-2, wherein another downsampling filter is applied for generating other downsampled luma samples that are generated from left neighboring lines of the collocated luma block.

[00199] 5. The method of any of clauses 1-4, wherein the downsampling filter has filter coefficients of [1, 2, 1].

[00200] 6. The method of any of clauses 1-5, wherein a downsampled luma sample, $pDsY[x]$, satisfies an equation, $pDsY[x] = (pY[2*x - 1][-1] + 2 * pY[2 * x][-1] + pY[2 * x + 1][-1] + 2) >> 2$, wherein $pY[2*x][-1]$, $pY[2*x - 1][-1]$ and $pY[2*x + 1][-1]$ are luma samples from the nearest above neighboring line, x being an integer.

[00201] 7. The method of any of clauses 1-6, wherein the downsampling filter is independent of a position of the chroma block with respect to a top boundary of the coding tree unit.

[00202] 8. The method of any of clauses 1-6, wherein the method is selectively applied due to a 4:2:2 colour format of the video.

[00203] 9. The method of clause 1, wherein, due to the chroma block not being at a top coding tree unit boundary, the N above neighboring lines include a nearest above line of the collocated luma block but excludes a second nearest above line.

[00204] 10. The method of clause 1, wherein N is greater than one.

- [00205] 11. The method of clause 10, wherein the N above neighboring lines include the nearest above line and a second nearest above line.
- [00206] 12. The method of clause 1, wherein the downsampling filter is dependent on a color format of the video.
- [00207] 13. The method of any of clause 1-12, wherein the downsampling filter is a 6-tap filter.
- [00208] 14. The method of any of clauses 1-12, wherein the downsampling filter is a 5-tap filter.
- [00209] 15. The method of any of clauses 1 to 14, wherein the conversion includes encoding the video into the bitstream representation.
- [00210] 16. The method of any of clauses 1 to 14, wherein the conversion includes decoding the video from the bitstream representation.
- [00211] 17. A video processing apparatus comprising a processor configured to implement a method recited in any one or more of clauses 1 to 16.
- [00212] 18. A computer readable medium storing program code that, when executed, causes a processor to implement a method recited in any one or more of clauses 1 to 16.
- [00213] 19. A computer readable medium that stores a bitstream representation generated according to any of the above described methods.
- [00214] The third set of clauses describe certain features and aspects of the disclosed techniques in the previous sections (e.g., items 2 to 7).
- [00215] 1. A method of video processing (e.g., method 1610 as shown in FIG. 16A), comprising: determining 1612, for a conversion between a video region of a component of a video and a bitstream representation of the video, a maximum allowed block size for a video block coded using a transform skip mode; and performing 1614 the conversion based on the determining.
- [00216] 2. The method of clause 1, wherein the transform skip mode comprises, during encoding, coding residual of the video block without applying a non-identity transform, or during decoding, determining a decoded video block without applying a non-identity inverse transform to residuals coded in the bitstream representation.
- [00217] 3. The method of clause 1, wherein the transform skip mode comprises a BDPCM (block differential pulse-code modulation) that corresponds to an intra-coding tool that uses a differential pulse-code modulation (DPCM) at a block level.
- [00218] 4. The method of clause 1, wherein the maximum allowed block size is dependent on

whether the transform skipped block is chroma block or a luma block.

[00219] 5. The method of clause 1, wherein the maximum allowed block size is dependent on a chroma component of the transform skipped block.

[00220] 6. The method of clause 1, wherein a maximum allowed block size for a luma block (MaxTsSizeY) and a maximum allowed block size for a chroma block (MaxTsSizeC) are separately signaled in the bitstream representation.

[00221] 7. The method of clause 6, wherein the MaxTsSizeC and/or the MaxTsSizeY is signaled at a sequence level, a picture level, a slice level, or a tile group level.

[00222] 8. The method of clause 6, wherein the MaxTsSizeY is conditionally signaled based on an enablement status of the transform skip mode.

[00223] 9. The method of clause 6, wherein the MaxTsSizeY is conditionally signaled based on a color format and/or an enablement status of the transform skip mode.

[00224] 10. The method of clause 1, wherein the conversion is performed by utilizing a predictive coding between maximum block sizes for a luma component and a chroma component.

[00225] 11. The method of clause 1, wherein the video block is a chroma video block and wherein the maximum allowed block size (MaxTsSizeC) for the video block is dependent on a maximum allowed block size (MaxTsSizeY) for another video block of a luma component.

[00226] 12. The method of clause 11, wherein the MaxTsSizeC is set equal to the MaxTsSizeY.

[00227] 13. The method of clause 11, wherein the MaxTsSizeC is set equal to $\text{MaxTsSizeY}/N$, whereby N is an integer.

[00228] 14. The method of clause 1, wherein the video block is a chroma video block and wherein the maximum allowed block size (MaxTsSizeC) for the video block is set according to chroma subsampling ratios.

[00229] 15. The method of clause 14, wherein the MaxTsSizeC is set equal to i) $\text{MaxTsSizeY} \gg \text{SubWidthC}$, ii) $\text{MaxTsSizeY} \gg \text{SubHeightC}$, iii) $\text{MaxTsSizeY} \gg \max(\text{SubWidthC}, \text{SubHeightC})$, iv) $\text{MaxTsSizeY} \gg \min(\text{SubWidthC}, \text{SubHeightC})$, wherein MaxTsSizeY indicates a maximum block size of a luma video block and SubWidthC and SubHeightC are predefined.

[00230] 16. A method of video processing (e.g., method 1610 as shown in FIG. 16A), comprising: performing a conversion between a video comprising video blocks and a bitstream

representation of the video according to a first rule and a second rule. A transform skip coding tool is used for coding a first portion of the video blocks, a transform coding tool is used for coding a second portion of the video blocks, the first rule specifies a maximum allowed block size for the first portion of the video blocks and the second rule specifies a maximum allowed block size for the second portion of the video blocks, and the maximum allowed block size for the first portion of the video blocks is different from the maximum allowed block size for the second portion of the video block.

[00231] 17. The method of clause 16, wherein the maximum allowed block size corresponds to a width and a height of a corresponding block.

[00232] 18. The method of clause 17, wherein the width and the height of the maximum allowed block size are signaled separately.

[00233] 19. The method of clause 17, wherein, for the second portion of the video blocks that are chroma blocks, the width (MaxTsSizeWC) is set equal to $\text{MaxTsSizeY} \gg \text{SubWidthC}$ and the height (MaxTsSizeHC) is set equal to $\text{MaxTsSizeY} \gg \text{SubHeightC}$, wherein MaxTsSizeY indicates a maximum allowed block size for a luma block.

[00234] 20. A method of video processing (e.g., method 1610 as shown in FIG. 16A), comprising: performing a conversion between a video comprising one or more chroma blocks and a bitstream representation of the video. The bitstream representation conforms to a format rule that specifies that whether a syntax element to indicate usage of a transform skip tool is included in the bitstream representation depends on a maximum allowed size for a chroma block that is coded using the transform skip tool.

[00235] 21. The method of clause 20, wherein the transform skip tool includes bypassing a transform or applying an identity transform.

[00236] 22. The method of clause 20, wherein the syntax element is signaled in a case that tbW is less than or equal to MaxTsSizeC and tbH is less than or equal to MaxTsSizeC , wherein tbW and tbH are a width and a height of the chroma block, respectively, and MaxTsSizeC is the maximum allowed size for the chroma block, respectively.

[00237] 23. The method of clause 20, wherein the syntax element is signaled in a case that tbW is less than or equal to MaxTsSizeWC and tbH is less than or equal to MaxTsSizeHC , wherein tbW and tbH are a width and a height of the chroma block, respectively, and MaxTsSizeWC and MaxTsSizeHC represent a width and a height of the maximum allowed size for the chroma

block, respectively.

[00238] 24. The method of clause 20, wherein the transform skip tool includes a BDPCM (block differential pulse-code modulation) mode that corresponds to an intra-coding tool that uses a differential pulse-code modulation (DPCM) at a block level.

[00239] 25. A method of video processing (e.g., method 1610 as shown in FIG. 16A), comprising: performing a conversion between a video comprising one or more first video blocks of a first chroma component and one or more second video blocks of a second chroma component and a bitstream representation of the video. The bitstream representation conforms to a format rule that specifies to use a syntax element that jointly indicates availability of a transform skip tool for coding the one or more first chroma blocks and the one or more second chroma blocks.

[00240] 26. The method of clause 25, wherein the syntax element has a binary value.

[00241] 27. The method of clause 25, wherein the transform skip tool is enabled or disabled in the one or more first video blocks and the one or more second video blocks according to the syntax element.

[00242] 28. The method of clause 25, wherein the format rule further specifies that an additional syntax element is included in the bitstream representation based on whether a value of the syntax element is equal to K, whereby K is an integer.

[00243] 29. The method of clause 28, wherein the second syntax element is used to indicate to which one of the one or more first video blocks and the one or more second video blocks the transform skip tool is applied.

[00244] 30. The method of clause 25, wherein the syntax element has a non-binary value.

[00245] 31. The method of clause 30, wherein the syntax element is coded with a fixed length, unary, truncated unary, or k-th order Exp-Golomb (EG) binarization method.

[00246] 32. The method of clause 25, wherein the syntax element is context coded or bypass coded.

[00247] 33. The method of any one of previous clauses, wherein whether to and/or how to apply the method is signaled at a sequence level, a picture level, a slice level, or a tile group level.

[00248] 34. The method of any one of previous clauses, wherein the method is further based on coded information.

[00249] 35. The method of any of clauses 1 to 34, wherein the conversion includes encoding the video into the bitstream representation.

[00250] 36. The method of any of clauses 1 to 34, wherein the conversion includes decoding the video from the bitstream representation.

[00251] 37. A video processing apparatus comprising a processor configured to implement a method recited in any one or more of clauses 1 to 36.

[00252] 38. A computer readable medium storing program code that, when executed, causes a processor to implement a method recited in any one or more of clauses 1 to 36.

[00253] 39. A computer readable medium that stores a coded representation or a bitstream representation generated according to any of the above described methods.

[00254] While this patent document contains many specifics, these should not be construed as limitations on the scope of any subject matter or of what may be claimed, but rather as descriptions of features that may be specific to particular embodiments of particular techniques. 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.

[00255] 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.

[00256] 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.