

WEIGHTED SAMPLE BI-PREDICTION IN 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 benefit of International Patent Application No. PCT/CN2019/105825, filed on September 13, 2019. For all purposes under the law, the entire disclosure of the aforementioned application is 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] Devices, systems and methods related to digital video processing, and specifically, to collocated motion vector in video processing are described. The described methods may be applied to both the existing video coding standards (e.g., High Efficiency Video Coding (HEVC)) and future video coding standards (e.g., Versatile Video Coding (VVC)) or codecs.

[005] In one exemplary aspect, a method of video processing is disclosed. The method includes making a first determination about whether a collocated video block of a video block of a video is coded using an inter coding mode or a non-inter coding mode; making a second determination, based on the first determination, about availability of motion vectors of the collocated video block or a mode of derivation of motion vectors for the collocated video block; and performing a conversion between the video block and a coded representation of the video based on the second determination.

[006] In another exemplary aspect, a method of video processing is disclosed. The method includes determining, for a conversion between a video block of a video and a coded representation of the video, one or more collocated motion vectors based on a reference index or a reference list of a collocated video block of the video block; and performing the conversion based on the determining.

[007] In yet another exemplary aspect, a method of video processing is disclosed. The method includes deriving, for a conversion between a current block of a video and a coded representation of the video, a prediction block for the current block that is determined according to a bi-directional coding unit level weighting (BCW) tool in which weights are used to determine a weighted sum of two initial prediction blocks used to determine the prediction block; and performing the conversion using the prediction block, wherein the deriving of the prediction block comprises: applying the weights to two intermediate prediction samples that respectively belong to the two initial prediction blocks to derive a final prediction sample; and performing a right shift operation for the final prediction sample to convert a bit depth of the final prediction sample, wherein the right shift operation is $\text{pbSample} \gg (\text{shift1} + 3)$, wherein pbSample represents the final prediction sample, shift1 is set equal to $\text{Max}(2, 14 - \text{bitDepth})$; and determining the prediction block based on the final prediction sample.

[008] In yet another exemplary aspect, a method of video processing is disclosed. The method includes deriving, for a conversion between a current block of a video and a coded representation of the video, a prediction block for the current block that is determined according to a bi-directional coding unit level weighting (BCW) tool in which weights are used to determine a weighted sum of two initial prediction blocks used to determine the prediction block; and performing the conversion using the prediction block; wherein the deriving of the prediction block comprises: applying the weights to two intermediate prediction sample that respectively belong to the two initial prediction blocks to derive a final prediction sample; and adding an offset to the final prediction sample, wherein the offset is $1 \ll (\text{shift1} + 2)$, wherein shift1 is set equal to $\text{Max}(2, 14 - \text{bitDepth})$; and determining the prediction block based on the final prediction sample.

[009] In yet another exemplary aspect, a method of video processing is disclosed. The method includes deriving, for a conversion between a current block of a video and a coded representation of the video, a motion vector difference for a merge mode motion vector predictor for the current

block according to a rule; and performing the conversion based on the deriving, wherein the rule specifies whether to clip the motion vector difference to a same range as a range used for clipping motion vectors.

[0010] In yet another exemplary aspect, a method of video processing is disclosed. The method includes deriving, for a conversion between a current block of a video and a coded representation of the video, a prediction block for the current block that is determined according to a bi-directional coding unit level weighting (BCW) tool in which weights are used to determine a weighted sum of two initial prediction blocks used to determine the prediction block, wherein at least one of the weights belongs to a weight table, wherein the weight table is organized according to a rule; and performing the conversion using the prediction block, wherein the rule specifies that entries of the weight table are non-monotonically increasing.

[0011] In yet another exemplary aspect, a method of video processing is disclosed. The method includes deriving, for a conversion between a current block of a video and a coded representation of the video, a prediction block for the current block that is determined according to a bi-directional coding unit level weighting (BCW) tool in which weights are used to determine a weighted sum of two initial prediction blocks used to determine the prediction block, wherein at least one of the weights belongs to a weight table, and the weight table is selected from multiple weight tables; and performing the conversion using the prediction block.

[0012] In yet another exemplary aspect, a method of video processing is disclosed. The method includes deriving, for a conversion between a current block of a video and a coded representation of the video, a prediction block for the current block that is determined according to a bi-directional coding unit level weighting (BCW) tool in which weights are used to determine a weighted sum of two initial prediction blocks used to determine the prediction block; and performing the conversion using the prediction block, wherein the deriving of the prediction block comprises: converting an intermediate prediction sample to a first bit depth; applying the weights to the intermediate prediction sample to derive a final prediction sample; and converting the final prediction sample to a second bit depth.

[0013] In yet another exemplary aspect, a method of video processing is disclosed. The method includes deriving, for a conversion between a current block of a video and a coded representation of the video, a prediction block for the current block that is determined according to a bi-directional coding unit level weighting (BCW) tool in which a first weight and a second weight

are used to determine a weighted sum of two initial prediction blocks used to determine the prediction block, wherein at least one of the first weight and the second weight is included in a weight table, wherein the weight table is organized according to a rule; and performing the conversion using the prediction block, wherein the rule specifies that the weight table is asymmetric.

[0014] In yet another example aspect, the above-described method may be implemented by a video encoder apparatus that comprises a processor.

[0015] 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.

[0016] These, and other, aspects are further described in the present document.

BRIEF DESCRIPTION OF THE DRAWINGS

[0017] FIG. 1 shows a block diagram of an example encoder.

[0018] FIG. 2 shows an example of 67 intra prediction modes.

[0019] FIG. 3 shows an example of horizontal and vertical traverse scans.

[0020] FIG. 4 shows an example of motion vector scaling for temporal merge candidates.

[0021] FIG. 5 shows candidate positions for a temporal merge candidate.

[0022] FIG. 6A shows an example of spatial neighboring blocks used by Alternative Temporal Motion Vector Prediction (ATMVP).

[0023] FIG. 6B shows an example of deriving sub-CU motion field.

[0024] FIG. 7 shows an example of a search point for the merge with motion vector differences (MMVD) mode.

[0025] FIG. 8 is a flowchart for an example of a video processing method.

[0026] FIG. 9 is a flowchart for another example of a video processing method.

[0027] FIG. 10 is a flowchart for yet another example of a video processing method.

[0028] FIG. 11 is a flowchart for yet another example of a video processing method.

[0029] FIG. 12 is a flowchart for yet another example of a video processing method.

[0030] FIG. 13 is a flowchart for yet another example of a video processing method.

[0031] FIGS. 14A and 14B are block diagrams of examples of a hardware platform for implementing a visual media decoding or a visual media encoding technique described in the present document.

[0032] FIGS. 15A to 15E are flowcharts of example methods of video processing based on some implementations of the disclosed technology.

DETAILED DESCRIPTION

[0033] 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.

[0034] 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.

[0035] 1. Summary

[0036] This document is related to video coding technologies. Specifically, it is related to collocated motion vectors and other coding tools. 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.

[0037] 2. Initial Discussion

[0038] 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.

[0039] 2.1 Color space and chroma subsampling

[0040] 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.

[0041] For video compression, the most frequently used color spaces are YCbCr and RGB.

[0042] 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.

[0043] 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.

[0044] 2.1.1 4:4:4

[0045] 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.

[0046] 2.1.2 4:2:2

[0047] The two chroma components are sampled at half the sample rate of luma: the horizontal chroma resolution is halved. This reduces the bandwidth of an uncompressed video signal by one-third with little to no visual difference.

[0048] 2.1.3 4:2:0

[0049] 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.

[0050] • In MPEG-2, Cb and Cr are cosited horizontally. Cb and Cr are sited between pixels in the vertical direction (sited interstitially).

[0051] • In JPEG/JFIF, H.261, and MPEG-1, Cb and Cr are sited interstitially, halfway between alternate luma samples.

[0052] • 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.

[0053] 2.2 Coding flow of a typical video codec

[0054] FIG. 1 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.

[0055] 2.3 Intra mode coding with 67 intra prediction modes

[0056] 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.

[0057] Conventional angular intra prediction directions are defined from 45 degrees to -135 degrees in clockwise direction as shown in FIG. 2. In VTM6, several conventional angular intra prediction modes are adaptively replaced with wide-angle intra prediction modes for the non-square blocks. The replaced modes are signaled 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.

[0058] 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.

[0059] 2.4 Inter prediction

[0060] 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.

[0061] **2.5 Intra block copy (IBC)**

[0062] 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.

[0063] 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.

[0064] 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 4×4 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.

[0065] In block matching search, the search range is set to cover both the previous and current CTUs.

[0066] At CU level, IBC mode is signalled with a flag and it can be signaled as IBC AMVP mode or IBC skip/merge mode as follows:

[0067] – IBC skip/merge mode: a merge candidate index is used to indicate which of the block vectors in the list from neighboring candidate IBC coded blocks is used to predict the current block. The merge list consists of spatial, HMVP, and pairwise candidates.

[0068] – 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 neighbor and one from above neighbor (if IBC coded). When either neighbor is not available, a default block vector will be used as a predictor. A flag is signaled to indicate the block vector predictor index.

[0069] 2.6 Palette Mode

[0070] For palette mode signaling, 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 signaled directly.

[0071] 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 signaled 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.

[0072] 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).

[0073] 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`.

[0074] 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.

[0075] 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.

[0076] In VTM5.0, 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.

[0077] 2.7 Temporal motion vector prediction (TMVP) in VVC

[0078] In the derivation of this temporal merge candidate, a scaled motion vector is derived based on co-located CU belonging to the collocated reference picture. The reference picture list to be used for derivation of the co-located CU is explicitly signalled in the slice header. The scaled motion vector for temporal merge candidate is obtained as illustrated by the dotted line in FIG. 4, which is scaled from the motion vector of the co-located CU using the POC distances, t_b and t_d , where t_b is defined to be the POC difference between the reference picture of the current picture and the current picture and t_d is defined to be the POC difference between the reference picture of the co-located picture and the co-located picture. The reference picture index of temporal merge candidate is set equal to zero.

[0079] The position for the temporal candidate is selected between candidates C_0 and C_1 , as depicted in FIG. 5. If CU at position C_0 is not available, is intra coded, or is outside of the current row of CTUs, position C_1 is used. Otherwise, position C_0 is used in the derivation of the temporal merge candidate.

[0080] 2.8 Subblock-based temporal motion vector prediction (SbTMVP) in VVC

[0081] VTM supports the subblock-based temporal motion vector prediction (SbTMVP) method. Similar to the temporal motion vector prediction (TMVP) in HEVC, SbTMVP uses the motion field in the collocated picture to improve motion vector prediction and merge mode for CUs in the current picture. The same collocated picture used by TMVP is used for SbTMVP. SbTMVP differs from TMVP in the following two main aspects:

[0082] 1. TMVP predicts motion at CU level but SbTMVP predicts motion at sub-CU level;

[0083] 2. Whereas TMVP fetches the temporal motion vectors from the collocated block in the collocated picture (the collocated block is the bottom-right or center block relative to the current CU), SbTMVP applies a motion shift before fetching the temporal motion information from the collocated picture, where the motion shift is obtained from the motion vector from one of the spatial neighboring blocks of the current CU.

[0084] The SbTMVP process is illustrated in FIG. 6. SbTMVP predicts the motion vectors of the sub-CUs within the current CU in two steps. In the first step, the spatial neighbor A1 in FIG. 6A is examined. If A1 has a motion vector that uses the collocated picture as its reference picture,

this motion vector is selected to be the motion shift to be applied. If no such motion is identified, then the motion shift is set to (0, 0).

[0085] In the second step, the motion shift identified in Step 1 is applied (i.e. added to the current block's coordinates) to obtain sub-CU-level motion information (motion vectors and reference indices) from the collocated picture as shown in FIG. 6B. The example in FIG. 6B assumes the motion shift is set to block A1's motion. Then, for each sub-CU, the motion information of its corresponding block (the smallest motion grid that covers the center sample) in the collocated picture is used to derive the motion information for the sub-CU. After the motion information of the collocated sub-CU is identified, it is converted to the motion vectors and reference indices of the current sub-CU in a similar way as the TMVP process of HEVC, where temporal motion scaling is applied to align the reference pictures of the temporal motion vectors to those of the current CU.

[0086] In VTM6, a combined sub-block based merge list which contains both SbTMVP candidate and affine merge candidates is used for the signalling of sub-block based merge mode. The SbTMVP mode is enabled/disabled by a sequence parameter set (SPS) flag. If the SbTMVP mode is enabled, the SbTMVP predictor is added as the first entry of the list of sub-block based merge candidates, and followed by the affine merge candidates. The size of sub-block based merge list is signalled in SPS and the maximum allowed size of the sub-block based merge list is 5 in VTM6.

[0087] The sub-CU size used in SbTMVP is fixed to be 8x8, and as done for affine merge mode, SbTMVP mode is only applicable to the CU with both width and height are larger than or equal to 8.

[0088] The encoding logic of the additional SbTMVP merge candidate is the same as for the other merge candidates, that is, for each CU in P or B slice, an additional RD check is performed to decide whether to use the SbTMVP candidate.

[0089] Description of TMVP and SbTMVP in working Draft

8.5.2.11 Derivation process for temporal luma motion vector prediction

Inputs to this process are:

- a luma location (xCb, yCb) of the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
- a variable cbWidth specifying the width of the current coding block in luma samples,

- a variable cbHeight specifying the height of the current coding block in luma samples,
- a reference index refIdxLX, with X being 0 or 1.

Outputs of this process are:

- the motion vector prediction mvLXCol in 1/16 fractional-sample accuracy,
- the availability flag availableFlagLXCol.

The variable currCb specifies the current luma coding block at luma location (xCb, yCb).

The variables mvLXCol and availableFlagLXCol are derived as follows:

- If slice_temporal_mvp_enabled_flag is equal to 0 or (cbWidth * cbHeight) is less than or equal to 32, both components of mvLXCol are set equal to 0 and availableFlagLXCol is set equal to 0.
- Otherwise (slice_temporal_mvp_enabled_flag is equal to 1), the following ordered steps apply:
 1. The bottom right collocated motion vector and the bottom and right boundary sample locations are derived as follows:

$$xColBr = xCb + cbWidth \quad (8-421)$$

$$yColBr = yCb + cbHeight \quad (8-422)$$

$$rightBoundaryPos = subpic_treated_as_pic_flag[SubPicIdx] ?$$

$$SubPicRightBoundaryPos : pic_width_in_luma_samples - 1 \quad (8-423)$$

$$botBoundaryPos = subpic_treated_as_pic_flag[SubPicIdx] ?$$

$$SubPicBotBoundaryPos : pic_height_in_luma_samples - 1 \quad (8-424)$$

- If yCb >> CtbLog2SizeY is equal to yColBr >> CtbLog2SizeY, yColBr is less than or equal to botBoundaryPos and xColBr is less than or equal to rightBoundaryPos, the following applies:
 - The variable colCb specifies the luma coding block covering the modified location given by ((xColBr >> 3) << 3, (yColBr >> 3) << 3) inside the collocated picture specified by ColPic.
 - The luma location (xColCb, yColCb) is set equal to the top-left sample of the collocated luma coding block specified by colCb relative to the top-left luma sample of the collocated picture specified by ColPic.
 - The derivation process for collocated motion vectors as specified in clause 8.5.2.12 is invoked with currCb, colCb, (xColCb, yColCb), refIdxLX and sbFlag set equal to 0 as inputs, and the output is assigned to mvLXCol and availableFlagLXCol.
 - Otherwise, both components of mvLXCol are set equal to 0 and availableFlagLXCol is set equal to 0.
- 2. When availableFlagLXCol is equal to 0, the central collocated motion vector is derived as follows:

$$xColCtr = xCb + (cbWidth >> 1) \quad (8-425)$$

$$yColCtr = yCb + (cbHeight >> 1) \quad (8-426)$$

- The variable colCb specifies the luma coding block covering the modified location given by $((xColCtr >> 3) << 3, (yColCtr >> 3) << 3)$ inside the collocated picture specified by ColPic.
- The luma location $(xColCb, yColCb)$ is set equal to the top-left sample of the collocated luma coding block specified by colCb relative to the top-left luma sample of the collocated picture specified by ColPic.
- The derivation process for collocated motion vectors as specified in clause 8.5.2.12 is invoked with currCb, colCb, $(xColCb, yColCb)$, refldxLX and sbFlag set equal to 0 as inputs, and the output is assigned to mvLXCol and availableFlagLXCol.

8.5.2.12 Derivation process for collocated motion vectors

Inputs to this process are:

- a variable currCb specifying the current coding block,
- a variable colCb specifying the collocated coding block inside the collocated picture specified by ColPic,
- a luma location $(xColCb, yColCb)$ specifying the top-left sample of the collocated luma coding block specified by colCb relative to the top-left luma sample of the collocated picture specified by ColPic,
- a reference index refldxLX, with X being 0 or 1,
- a flag indicating a subblock temporal merging candidate sbFlag.

Outputs of this process are:

- the motion vector prediction mvLXCol in 1/16 fractional-sample accuracy,
- the availability flag availableFlagLXCol.

The variable currPic specifies the current picture.

The arrays predFlagL0Col[x][y], mvL0Col[x][y] and refldxL0Col[x][y] are set equal to PredFlagL0[x][y], MvDmvrL0[x][y] and RefldxL0[x][y], respectively, of the collocated picture specified by ColPic, and the arrays predFlagL1Col[x][y], mvL1Col[x][y] and refldxL1Col[x][y] are set equal to PredFlagL1[x][y], MvDmvrL1[x][y] and RefldxL1[x][y], respectively, of the collocated picture specified by ColPic.

The variables mvLXCol and availableFlagLXCol are derived as follows:

- *If colCb is coded in an intra or IBC prediction mode, both components of mvLXCol are set equal to 0 and availableFlagLXCol is set equal to 0.*

- Otherwise, the motion vector *mvCol*, the reference index *refIdxCol* and the reference list identifier *listCol* are derived as follows:

2.8.1 Derivation for collocated motion vectors when colCb is coded in an inter block

- If *sbFlag* is equal to 0, *availableFlagLXCol* is set to 1 and the following applies:
 - If *predFlagL0Col*[*xColCb*][*yColCb*] is equal to 0, *mvCol*, *refIdxCol* and *listCol* are set equal to *mvL1Col*[*xColCb*][*yColCb*], *refIdxL1Col*[*xColCb*][*yColCb*] and *L1*, respectively.
 - Otherwise, if *predFlagL0Col*[*xColCb*][*yColCb*] is equal to 1 and *predFlagL1Col*[*xColCb*][*yColCb*] is equal to 0, *mvCol*, *refIdxCol* and *listCol* are set equal to *mvL0Col*[*xColCb*][*yColCb*], *refIdxL0Col*[*xColCb*][*yColCb*] and *L0*, respectively.
 - Otherwise (*predFlagL0Col*[*xColCb*][*yColCb*] is equal to 1 and *predFlagL1Col*[*xColCb*][*yColCb*] is equal to 1), the following assignments are made:
 - If *NoBackwardPredFlag* is equal to 1, *mvCol*, *refIdxCol* and *listCol* are set equal to *mvLXCol*[*xColCb*][*yColCb*], *refIdxLXCol*[*xColCb*][*yColCb*] and *LX*, respectively.
 - Otherwise, *mvCol*, *refIdxCol* and *listCol* are set equal to *mvLNCol*[*xColCb*][*yColCb*], *refIdxLNCol*[*xColCb*][*yColCb*] and *LN*, respectively, with *N* being the value of *collocated_from_l0_flag*.
- Otherwise (*sbFlag* is equal to 1), the following applies:
 - If *PredFlagLXCol*[*xColCb*][*yColCb*] is equal to 1, *mvCol*, *refIdxCol*, and *listCol* are set equal to *mvLXCol*[*xColCb*][*yColCb*], *refIdxLXCol*[*xColCb*][*yColCb*], and *LX*, respectively, *availableFlagLXCol* is set to 1.
 - Otherwise (*PredFlagLXCol*[*xColCb*][*yColCb*] is equal to 0), the following applies:
 - If *NoBackwardPredFlag* is equal to 1 and *PredFlagLYCol*[*xColCb*][*yColCb*] is equal to 1, *mvCol*, *refIdxCol*, and *listCol* are set to *mvLYCol*[*xColCb*][*yColCb*], *refIdxLYCol*[*xColCb*][*yColCb*] and *LY*, respectively, with *Y* being equal to !*X* where *X* being the value of *X* this process is invoked for. *availableFlagLXCol* is set equal to 1.
 - Otherwise, both components of *mvLXCol* are set equal to 0 and *availableFlagLXCol* is set equal to 0.
- When *availableFlagLXCol* is equal to TRUE, *mvLXCol* and *availableFlagLXCol* are derived as follows:

- If LongTermRefPic(currPic, currCb, refIdxLX, LX) is not equal to LongTermRefPic(ColPic, colCb, refIdxCol, listCol), both components of mvLXCol are set equal to 0 and availableFlagLXCol is set equal to 0.
- Otherwise, the variable availableFlagLXCol is set equal to 1, refPicList[listCol][refIdxCol] is set to be the picture with reference index refIdxCol in the reference picture list listCol of the slice containing coding block colCb in the collocated picture specified by ColPic, and the following applies:

$$\text{colPocDiff} = \text{DiffPicOrderCnt}(\text{ColPic}, \text{refPicList}[\text{listCol}][\text{refIdxCol}]) \quad (8-427)$$

$$\text{currPocDiff} = \text{DiffPicOrderCnt}(\text{currPic}, \text{RefPicList}[X][\text{refIdxLX}]) \quad (8-428)$$

- The temporal motion buffer compression process for collocated motion vectors as specified in clause 8.5.2.15 is invoked with mvCol as input, and the modified mvCol as output.
- If RefPicList[X][refIdxLX] is a long-term reference picture, or colPocDiff is equal to currPocDiff, mvLXCol is derived as follows:

$$\text{mvLXCol} = \text{mvCol} \quad (8-429)$$

- Otherwise, mvLXCol is derived as a scaled version of the motion vector mvCol as follows:

$$\text{tx} = (16384 + (\text{Abs}(\text{td}) \gg 1)) / \text{td} \quad (8-430)$$

$$\text{distScaleFactor} = \text{Clip3}(-4096, 4095, (\text{tb} * \text{tx} + 32) \gg 6) \quad (8-431)$$

$$\text{mvLXCol} = \text{Clip3}(-131072, 131071, (\text{distScaleFactor} * \text{mvCol} + 128 - (\text{distScaleFactor} * \text{mvCol} \geq 0) \gg 8)) \quad (8-432)$$

where td and tb are derived as follows:

$$\text{td} = \text{Clip3}(-128, 127, \text{colPocDiff}) \quad (8-433)$$

$$\text{tb} = \text{Clip3}(-128, 127, \text{currPocDiff}) \quad (8-434)$$

[0090] 2.9 Bi-prediction with CU-level weight (BCW)

[0091] In HEVC, the bi-prediction signal is generated by averaging two prediction signals obtained from two different reference pictures and/or using two different motion vectors. In VTM6, the bi-prediction mode is extended beyond simple averaging to allow weighted averaging of the two prediction signals.

$$P_{\text{bi-pred}} = ((8 - w) * P_0 + w * P_1 + 4) \gg 3 \quad (3-19)$$

[0092] Five weights are allowed in the weighted averaging bi-prediction, $w \in \{-2, 3, 4, 5, 10\}$. For each bi-predicted CU, the weight w is determined in one of two ways: 1) for a non-merge CU,

the weight index is signalled after the motion vector difference; 2) for a merge CU, the weight index is inferred from neighbouring blocks based on the merge candidate index. Weighted averaging bi-prediction is only applied to CUs with 256 or more luma samples (i.e., CU width times CU height is greater than or equal to 256). For low-delay pictures, all 5 weights are used. For non-low-delay pictures, only 3 weights ($w \in \{3, 4, 5\}$) are used.

[0093] – At the encoder, fast search algorithms are applied to find the weight index without significantly increasing the encoder complexity. These algorithms are summarized as follows. For further details readers are referred to the VTM software and document JVET-L0646. When combined with AMVR, unequal weights are only conditionally checked for 1-pel and 4-pel motion vector precisions if the current picture is a low-delay picture.

[0094] – When combined with affine, affine ME will be performed for unequal weights if and only if the affine mode is selected as the current best mode.

[0095] – When the two reference pictures in bi-prediction are the same, unequal weights are only conditionally checked.

[0096] – Unequal weights are not searched when certain conditions are met, depending on the POC distance between current picture and its reference pictures, the coding QP, and the temporal level.

[0097] The BCW weight index is coded using one context coded bin followed by bypass coded bins. The first context coded bin indicates if equal weight is used; and if unequal weight is used, additional bins are signalled using bypass coding to indicate which unequal weight is used.

[0098] Weighted prediction (WP) is a coding tool supported by the H.264/AVC and HEVC standards to efficiently code video content with fading. Support for WP was also added into the VVC standard. WP allows weighting parameters (weight and offset) to be signalled for each reference picture in each of the reference picture lists L0 and L1. Then, during motion compensation, the weight(s) and offset(s) of the corresponding reference picture(s) are applied. WP and BCW are designed for different types of video content. In order to avoid interactions between WP and BCW, which will complicate VVC decoder design, if a CU uses WP, then the BCW weight index is not signalled, and w is inferred to be 4 (i.e. equal weight is applied). For a merge CU, the weight index is inferred from neighbouring blocks based on the merge candidate index. This can be applied to both normal merge mode and inherited affine merge mode. For

constructed affine merge mode, the affine motion information is constructed based on the motion information of up to 3 blocks. The following process is used to derive the BCW index for a CU using the constructed affine merge mode.

[0099] 1. Divide the range of BCW index $\{0,1,2,3,4\}$ into three groups $\{0\}$, $\{1,2,3\}$ and $\{4\}$. If all of the control point's BCW indices are from the same group, the BCW index is derived as in the step 2; otherwise, the BCW index is set to 2.

[00100] If at least two control points have the same BCW index, then this BCW index value is assigned to the candidate; else, the BCW index of the current constructed candidate is set to 2.

[00101] 2.10 Merge mode with MVD (MMVD)

[00102] In addition to merge mode, where the implicitly derived motion information is directly used for prediction samples generation of the current CU, the merge mode with motion vector differences (MMVD) is introduced in VVC. A MMVD flag is signaled right after sending a skip flag and merge flag to specify whether MMVD mode is used for a CU.

[00103] In MMVD, after a merge candidate is selected, it is further refined by the signalled MVDs information. The further information includes a merge candidate flag, an index to specify motion magnitude, and an index for indication of motion direction. In MMVD mode, one for the first two candidates in the merge list is selected to be used as MV basis. The merge candidate flag is signalled to specify which one is used.

[00104] Distance index specifies motion magnitude information and indicate the pre-defined offset from the starting point. As shown in FIG. 7, an offset is added to either horizontal component or vertical component of starting MV. The relation of distance index and pre-defined offset is specified in Table 1.

Table 1: The relation of distance index and pre-defined offset

Distance IDX	0	1	2	3	4	5	6	7
Offset (in unit of luma sample)	1/4	1/2	1	2	4	8	16	32

[00105] Direction index represents the direction of the MVD relative to the starting point. The direction index can represent of the four directions as shown in Table2. It's noted that the meaning of MVD sign could be variant according to the information of starting MVs. When the starting MVs is an un-prediction MV or bi-prediction MVs with both lists point to the same side of the current picture (i.e. POCs of two references are both larger than the POC of the current

picture, or are both smaller than the POC of the current picture), the sign in Table 2 specifies the sign of MV offset added to the starting MV. When the starting MVs is bi-prediction MVs with the two MVs point to the different sides of the current picture (i.e. the POC of one reference is larger than the POC of the current picture, and the POC of the other reference is smaller than the POC of the current picture), the sign in Table 2 specifies the sign of MV offset added to the list0 MV component of starting MV and the sign for the list1 MV has opposite value.

Table 2: Sign of MV offset specified by direction index

Direction IDX	00	01	10	11
x-axis	+	–	N/A	N/A
y-axis	N/A	N/A	+	–

[00106] 2.11 Alternative luma half-pel interpolation filters

[00107] In JVET-N0309, alternative half-pel interpolation filters are proposed.

[00108] The switching of the half-pel luma interpolation filter is done depending on the motion vector accuracy. In addition to the existing quarter-pel, full-pel, and 4-pel AMVR modes, a new half-pel accuracy AMVR mode is introduced. Only in case of half-pel motion vector accuracy, an alternative half-pel luma interpolation filter can be selected.

[00109] 2.11.1 Half-pel AMVR mode

[00110] An additional AMVR mode for non-affine non-merge inter-coded CUs is proposed which allows signaling of motion vector differences at half-pel accuracy. The existing AMVR scheme of the current VVC draft is extended straightforward in the following way: Directly following the syntax element `amvr_flag`, if `amvr_flag == 1`, there is a new context-modeled binary syntax element `hpel_amvr_flag` which indicates usage of the new half-pel AMVR mode if `hpel_amvr_flag == 1`. Otherwise, i.e. if `hpel_amvr_flag == 0`, the selection between full-pel and 4-pel AMVR mode is indicated by the syntax element `amvr_precision_flag` as in the current VVC draft.

[00111] 2.11.2 Alternative luma half-pel interpolation filters

[00112] For a non-affine non-merge inter-coded CU which uses half-pel motion vector accuracy (i.e., the half-pel AMVR mode), a switching between the HEVC/VVC half-pel luma interpolation filter and one or more alternative half-pel interpolation is made based on the value of a new syntax element `if_idx`. The syntax element `if_idx` is only signaled in case of half-pel

AMVR mode. In case of skip/merge mode using a spatial merging candidate, the value of the syntax element `if_idx` is inherited from the neighbouring block.

[00113] 2.11.2.1 Test 1: One alternative half-pel interpolation filter

[00114] In this test case, there is one 6-tap interpolation filter as an alternative to the ordinary HEVC/VVC half-pel interpolation filter. The following table shows the mapping between the value of the syntax element `if_idx` and the selected half-pel luma interpolation filter:

<code>if_idx</code>	Binarization	Filter	Interpolation filter coefficients
0	0	Gauss (6-tap)	[0, 3, 9, 20, 20, 9, 3, 0]
1	1	HEVC/VVC (8-tap)	[-1, 4, -11, 40, 40, -11, 4, -1]

[00115] 2.11.2.1 Test 2: Two alternative half-pel interpolation filters

[00116] In this test case, there are two 8-tap interpolation filters as an alternative to the ordinary HEVC/VVC half-pel interpolation filter. The following table shows the mapping between the value of the syntax element `if_idx` and the selected half-pel luma interpolation filter:

<code>if_idx</code>	Binarization	Filter	Interpolation filter coefficients
0	0	Filter 1 (8-tap)	[3, 6, 10, 13, 13, 10, 6, 3]
1	10	Filter 2 (8-tap)	[-1, -1, 9, 25, 25, 9, -1, -1]
2	11	HEVC/VVC (8-tap)	[-1, 4, -11, 40, 40, -11, 4, -1]

[00117] `amvr_precision_idx` is signalled to indicate whether the current CU employs 1/2-pel, 1-pel or 4-pel MV precision. There are 2 bins to be coded.

[00118] `hpel_if_idx` is signalled to indicate whether the default half-pel interpolation filter or alternative half-pel interpolation filters are used. When 2 alternative half-pel interpolation filters are used, there are 2 bins to be coded.

[00119] 3. Drawbacks of existing implementations

[00120] The current design of derivation of collocated motion vectors in inter prediction has the following problems:

[00121] 1. The derivation of the collocated motion vectors in TMVP and SbTMVP depends on prediction mode of the collocated coding block. In current VVC, If the collocated coding block is coded in an intra or IBC prediction mode, the collocated motion vectors are set equal to zero motion vector. Therefore, even the collocated block is palette coded, it may still return a collocated motion vector which is undefined since there is no motion vector associated with a palette prediction mode.

- [00122] 2. The derivation of weighted sample prediction may not be efficient in BCW.
- [00123] 3. In current VVC, MV is clipped to 18 bits. However, the merge motion vector difference is clipped to 16 bits, which may result accuracy loss.
- [00124] 4. The signaled cu_skip_flag may cause overhead bits. In current VVC, the maximum width and height of coding unit for IBC is 64. There is no need to signal cu_skip_flag for blocks of either width or height is larger than 64 in I slice.
- [00125] 5. Alternative luma half-pel interpolation filters flag may be set equal to true even when the CU/PU/block has no half-pel or coarser MV component.

[00126] 4. Example methods for collocated motion vectors in video coding

[00127] The detailed inventions below should be considered as examples to explain general concepts. These inventions should not be interpreted in a narrow way. Furthermore, these inventions can be combined in any manner.

Derivation of the collocated motion vectors

1. Instead of check whether the collocated coding block is intra or IBC coded, how to derive the collocated motion vectors and/or the availability of collocated motion vectors may depend on whether the prediction mode of the collocated coding block is inter or not. In this case, there is no need to store the four prediction modes for each block, only one 1 bit is required to tell whether the block is inter or non-inter.
 - a. In one example, how to derive the collocated motion vectors and/or the availability of collocated motion vectors for the case that the collocated coding block is coded in palette prediction mode may be the same as that for the case that the collocated coding block is coded in intra/IBC prediction mode.
 - b. In one example, when the collocated coding block is coded in a non-inter prediction mode (e.g., intra, palette, or IBC), the collocated motion vector may be set to be unavailable.
 - c. Alternatively, collocated motion vectors may be marked as available, and a default motion vector may be assigned as the collocated motion vector when the collocated coding block is coded in a non-inter prediction mode (e.g., intra, palette, or IBC).

- d. Alternatively, other blocks (e.g., one neighboring inter coded block of the collocated coding block) may be checked instead when the collocated coding block is coded in a non-inter prediction mode.
 - i. In one example, the neighboring block may be the nearest inter coded block left/right/above/bottom the collocated coding block.
- 2. The determination of the collocated motion vectors may depend on the reference lists and/or reference indices of the collocated coding block.
 - a. In one example, the collocated motion vectors may be derived (e.g., using prior art as described in 2.8.1) when the reference index of reference list X (e.g. L1) of the collocated coding block is not equal to a specific value (e.g., 0).
 - b. In one example, the derivation of the collocated motion vectors may be invoked when the reference index of reference list X (e.g. L1) of the collocated coding block is equal to a specific value (e.g., 0).
 - c. Alternatively, furthermore, when the collocated coding block is coded in a non-inter prediction mode (including or excluding IBC), or its reference picture doesn't satisfy a given condition, the collocated MV is set to be unavailable.
 - i. Alternatively, the collocated MV is set to a default value.

Sample prediction of BCW

- 3. The weight table applied in the BCW coding mode may be asymmetry.
 - a. In one example, for a weight W as an entry of the table, $(1-W)$ may not be a table entry, wherein W is in the range of $[a, b]$ wherein $(a + b)$ is equal to 1.
 - b. In one example, for a weight W as an entry of the table, $(2^N - W)$ may not be a table entry, assuming that the final prediction block is generated by $(W_0 * P_0 + W_1 * P_1) \gg N$ wherein W_0 and W_1 are two weights applied to two prediction blocks P_0 and P_1 , respectively; and $(W_0 + W_1)$ is equal to $(1 \ll N)$.
- 4. The weights in the weight table applied in the BCW coding mode may not be in a monotonically increasing order.
 - a. In one example, the $(i+1)$ -th entry of the table may have a smaller value than the i -th entry of the table.
- 5. In one example, the weighted sample prediction process of BCW may depend on a different weight look up table.

- a. In one example, $\{4, 5, 3, 10, 2\}/\{4, 3, 5, 10, 2\}/\{4, 5, 3, 10, 1\}/\{4, 3, 5, 10, 1\}/\{4, 5, 3, 10, -1\}$ may be used as the weight look up table of BCW.
6. In one example, in the weighted sample prediction process of BCW, the intermediate prediction sample in each prediction direction may be converted (if necessary, e.g., when the first bit depth is not equal to the bit depth of the intermediate prediction sample) to a first bit depth, then weighted prediction may be applied and the final prediction sample may be converted to a second bit depth.
 - a. In one example, the second bit depth is same with the input bit depth of the current color component.
 - b. In one example, in conversion between different bit depth, right shift (e.g., convert a sample from a higher bit depth to a lower bit depth) or left shift (e.g., convert a sample from a lower bit depth to a higher bit depth) may be applied.
 - i. Alternatively, furthermore, an offset may be added before the right shift or left shift.
 - c. In one example, the first bit depth is same with the bit depth of the intermediate prediction sample. The weighted sample prediction of BCW may be derived as:
$$pbSamples[x][y] = Clip3(0, (1 \ll bitDepth) - 1, (w0 * predSamplesL0[x][y] + w1 * predSamplesL1[x][y] + offset3) \gg (shift1 + 3))$$
, where shift1 is set equal to $Max(2, 14 - bitDepth)$, the variable offset3 is set equal to $1 \ll (shift1 + 2)$, and the bitDepth is a bit depth of a current color component.
 - d. In one example, the first bit depth is same to the input bit depth of the current color component. The weighted sample prediction of BCW may be derived as:
$$pbSamples[x][y] = Clip3(0, (1 \ll bitDepth) - 1, (w0 * ((predSamplesL0[x][y] + offset1) \gg shift1) + w1 * ((predSamplesL1[x][y] + offset1) \gg shift1) + 4) \gg 3)$$
, where shift1 is set equal to $Max(2, 14 - bitDepth)$, the variable offset1 is set equal to $1 \ll (shift1 - 1)$.

MMVD range

7. In one example, the merge motion vector difference may be clipped to the same range as the motion vector.

- a. In one example, the merge motion vector difference may be clipped to 18 bits, e.g., $[-2^{17}, 2^{17} - 1]$, which is the same as motion vector range in VVC.
- b. In one example, the merge motion vector difference may be clipped to $[-2^{17}+1, 2^{17} - 1]$.
- c. In one example, the merge motion vector difference may not be clipped.
 - i. For example, after the merge motion vector difference is added to the motion vector prediction candidate, the result of motion vector is clipped to 18 bits.

Signal of the cu_skip_flag

- 8. It is proposed that the indication of whether a CU/PU/block is coded in skip mode (e.g., **cu_skip_flag**) may be conditionally signaled, depending on the dimensions of the coding block.
 - a. In one example, the indication (e.g., **cu_skip_flag**) may be signaled when the current slice type is I-slice, and the `sps_ibc_enabled_flag` is equal to true, and both the block width and block height are less than or equal to N (N is an integer). For example, N = 64.
 - b. In one example, when the current block is coded in IBC mode, the indication (e.g., **cu_skip_flag**) may be signaled when both the block width and block height are less than or equal to N (N is an integer). For example, N = 64.

Coding tools on chroma components

- 9. Whether a coding tool X (e.g., X being TMVP/ATMVP/BCW/MMVD/PDPC) could be applied on one block of a first color component may depend on whether it is applied on one or multiple corresponding blocks in a second color component.
 - a. In one example, the usage of the coding tool X for a first color component (e.g., chroma) block may be disabled when the coding tool is applied to the corresponding second color component (e.g., luma) block.
 - b. In one example, the usage of the coding tool X for a first color component (e.g., chroma) block may be disabled when a coding tool Y (wherein Y is different from X) is applied to the corresponding second color component (e.g., luma) block.

- c. In one example, a message (such as a flag or an index) may be conditionally signaled to indicate whether coding tool X is applied on first color components of a block. The condition may be defined as whether it is applied on the corresponding second color component blocks. Alternatively, furthermore, it is not applied on first components of a block without signaling if it is not applied on the corresponding second color component blocks.
 - i. In one example, coding tool X may be applied in a different manner on different color components.
 - 1) It may be signaled how to apply coding tool X on the luma component and the chroma components separately.
- d. In one example, the first color component is chroma component and the second color component is the luma component.
- e. In one example, the first color component is one chroma color component and the second color component is another chroma color component.
- f. In one example, the first color component is luma color component and the second color component is a chroma color component.
- g. In the above discussion, a “corresponding second color component block” may refer to a second color component block which covers at least one “corresponding sample” of the first color component block.
 - i. In one example, the first color component is chroma component and the second color component is the luma component.
 - ii. The sample positions may be scaled according to the color format such as 4:4:4 or 4:2:0. Suppose the top-left position of the chroma block is (x0, y0), and the width and height of the chroma block are W and H, all of which are scaled to the luma sample unit.
 - iii. In one example, the corresponding sample may be at (x0, y0);
 - iv. In one example, the corresponding sample may be at (x0+W-1, y0+H-1);
 - v. In one example, the corresponding sample may be at (x0+W/2-1, y0+H/2-1);
 - vi. In one example, the corresponding sample may be at (x0+W/2, y0+H/2);

- vii. In one example, the corresponding sample may be at $(x_0+W/2, y_0+H/2-1)$;
 - viii. In one example, the corresponding sample may be at $(x_0+W/2-1, y_0+H/2)$;
 - h. In the above discussion, “chroma components” may refer to “one or multiple chroma components”.
- 10. It is proposed that position-dependent intra prediction sample filtering process (a.k.a. PDPC) may be disabled for some chroma components when the co-located luma block of the chroma block is coded in some modes.
 - a. In one example, such process may be disabled when the co-located luma block of the chroma block is coded in MIP (Matrix-based Intra Prediction) mode.
 - b. In one example, such process may be disabled when the co-located luma block of the chroma block is coded in MRL (Multiple Reference Line) mode.
- 11. Indication of whether an alternative half-pel interpolation filter is used may be set equal to false when a CU/PU/block only has MV component with finer precision (e.g., 1/4-pel, 1/8-pel etc.) than half-pel. That is to say, the default half-pel interpolation filter instead of the alternative half-pel interpolation filter may be used.
 - a. In one example, such indication may be set equal to false in MMVD mode if the reconstructed MV only has MV component with finer precision than half-pel.
 - b. In one example, such indication may be set equal to false for a pair-wise merge candidate if it only has MV component with finer precision than half-pel.
 - c. In one example, such indication may be set equal to false when a reconstructed bi-predicted motion information is converted to a uni-predicted motion information (e.g., applied on small blocks like $4*8$ or/and $8*4$ blocks) which only has MV component with finer precision than half-pel.
- 12. Indication of whether an alternative half-pel interpolation filter is used may be always set equal to false in some cases. That is to say, the default half-pel interpolation filter may be always used in these cases.
 - a. In one example, such indication may be always set equal to false in MMVD mode.
 - b. In one example, such indication may be always set equal to false in MMVD mode when some specific MVDs are selected.

- i. For example, if an MVD with 1/4-pel precision is selected, such indication may be set equal to false.
- c. In one example, such indication may be always set equal to false for pair-wise merge candidates.
- d. In one example, such indication may be always set equal to false when a reconstructed bi-predicted motion information is converted to a uni-predicted motion information (e.g., applied on small blocks like 4*8 or/and 8*4 blocks).

General solutions

13. Whether to and/or how to apply the disclosed methods above may be signaled 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.

[00128] The examples described above may be incorporated in the context of the methods described below, e.g., methods 800, 900, 1000, 1100, 1200 and 1300, which may be implemented at a video decoder or a video encoder.

[00129] FIG. 8 shows a flowchart of an exemplary method for video processing. The method 800 includes, at step 810, determining an availability of one or more collocated motion vectors based on a prediction mode of a collocated video block relative to a current video block.

[00130] The method 800 includes, at step 820, performing, based on the one or more collocated motion vectors, a conversion between the current block and a bitstream representation of the current block, an indication of the prediction mode comprising one bit that indicates whether the current video block is coded with an inter mode or a non-inter mode.

[00131] FIG. 9 shows a flowchart of an exemplary method for video processing. The method 800 includes, at step 910, determining, for a conversion between a coded representation of a video block and the video block, a type of a collocated video block of the video block, the type taking one of only two possible values.

[00132] The method 800 includes, at step 920, performing the conversion based on the determining.

[00133] FIG. 10 shows a flowchart of an exemplary method for video processing. The method 800 includes, at step 1010, deriving, based on a first weight table associated with a bi-prediction mode with a coding unit (CU)-level weight (BCW) process, a prediction sample for a current block, the first weight table being asymmetric.

[00134] The method 800 includes, at step 1020, performing, based on the prediction sample, a conversion between the current block and a bitstream representation of the current block.

[00135] FIG. 11 shows a flowchart of an exemplary method for video processing. The method 800 includes, at step 1110, making a decision, based on a dimension of the current video block, regarding a selective signaling of an indication of a skip mode coding of the current video block in a bitstream representation of the current video block.

[00136] The method 800 includes, at step 1120, performing, based on the decision, a conversion between the current block and the bitstream representation of the current block.

[00137] FIG. 12 shows a flowchart of an exemplary method for video processing. The method 800 includes, at step 1210, making a decision, based on an application of a first coding tool to one or more blocks in a first color component of a current video block, regarding a selective application of a second coding tool to at least one block of a second color component of the current video block.

[00138] The method 800 includes, at step 1220, performing, based on the decision, a conversion between the current block and the bitstream representation of the current block.

[00139] FIG. 13 shows a flowchart of an exemplary method for video processing. The method 800 includes, at step 1310, making a decision, based on a precision of a motion vector in a current video block, regarding a selective signaling of an indication for using an alternative half-pel interpolation filter instead of a default half-pel interpolation filter.

[00140] The method 800 includes, at step 1320, performing, based on the decision, a conversion between the current block and the bitstream representation of the current block.

[00141] 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.

[00142] 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 disabled based on the decision or determination.

[00143] 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.

[00144] **5. Exemplary embodiments of the disclosed technology**

[00145] The changes are highlighted in bold and Italic. Deleted texts are marked with double brackets (e.g., [[a]] denotes the deletion of the character “a”).

[00146] **5.1 Embodiment #1**

[00147] The working draft specified in JVET-O2001-vE may be changed as below.

8.5.2.12 Derivation process for collocated motion vectors

...

The variables mvLXCol and availableFlagLXCol are derived as follows:

- If colCb is coded in an intra or *palette or* IBC prediction mode, both components of mvLXCol are set equal to 0 and availableFlagLXCol is set equal to 0.
- Otherwise, the motion vector mvCol, the reference index refIdxCol and the reference list identifier listCol are derived as follows:

...

Alternatively, the following may apply:

The variables mvLXCol and availableFlagLXCol are derived as follows:

- If colCb is *not* coded in an [[intra or IBC]] *inter* prediction mode, both components of mvLXCol are set equal to 0 and availableFlagLXCol is set equal to 0.
- Otherwise, the motion vector mvCol, the reference index refIdxCol and the reference list identifier listCol are derived as follows:

...

[00148] 5.2 Embodiment #2

[00149] The working draft specified in JVET-O2001-vE may be changed as below.

8.5.1 General decoding process for coding units coded in inter prediction mode

...

3. The arrays of luma and chroma motion vectors after decoder side motion vector refinement, refMvLX[xSbIdx][ySbIdx] and refMvCLX[xSbIdx][ySbIdx], with X being 0 and 1, are derived as follows for xSbIdx = 0..numSbX - 1, ySbIdx = 0..numSbY - 1:

- If dmvrFlag is equal to 1, the derivation process for chroma motion vectors in clause 8.5.2.13 is invoked with refMvLX[xSbIdx][ySbIdx] and refIdxLX as inputs, and refMvCLX[xSbIdx][ySbIdx] as output and the input refMvLX[xSbIdx][ySbIdx] is derived as follows;

$$\text{refMvLX}[xSbIdx][ySbIdx] = \text{mvLX}[xSbIdx][ySbIdx] + \text{dMvLX}[xSbIdx][ySbIdx] \quad (8-287)$$

$$\text{refMvLX}[xSbIdx][ySbIdx][0] = \text{Clip3}(-2^{17}, 2^{17} - 1, \text{refMvLX}[xSbIdx][ySbIdx][0]) \quad (8-288)$$

$$\text{refMvLX}[xSbIdx][ySbIdx][1] = \text{Clip3}(-2^{17}, 2^{17} - 1, \text{refMvLX}[xSbIdx][ySbIdx][1]) \quad (8-289)$$

- Otherwise (dmvrFlag is equal to 0), the following applies:

$$\text{refMvLX}[xSbIdx][ySbIdx] = \text{mvLX}[xSbIdx][ySbIdx] \quad (8-290)$$

$$\text{refMvCLX}[xSbIdx][ySbIdx] = \text{mvCLX}[xSbIdx][ySbIdx] \quad (8-291)$$

NOTE – The array refMvLX is stored in MvDmvrLX and used in the derivation process for collocated motion vectors in clause 8.5.2.12. After decoding a slice,

MvDmvrLX[xSbIdx][ySbIdx] and the corresponding reference indices are all set equal to -1 when the coding block Cb[xSbIdx][ySbIdx] is coded in an IBC prediction mode. The array of

non-refine luma motion vectors $MvLX$ is used in the spatial motion vector prediction and deblocking boundary strength derivation processes.

...

8.5.2.12 Derivation process for collocated motion vectors

...

The variables $mvLXCol$ and $availableFlagLXCol$ are derived as follows:

- If $colCb$ is coded in an intra or IBC prediction mode, both components of $mvLXCol$ are set equal to 0 and $availableFlagLXCol$ is set equal to 0.

- Otherwise, the motion vector $mvCol$, the reference index $refIdxCol$ and the reference list identifier $listCol$ are derived as follows:

If $predFlagL0Col[xColCb][yColCb]$ is equal to 0 and

$predFlagL1Col[xColCb][yColCb]$ is equal to 0, both components of $mvLXCol$ are set equal to 0 and $availableFlagLXCol$ is set equal to 0.

Otherwise, the following applies:

- If $sbFlag$ is equal to 0, $availableFlagLXCol$ is set to 1 and the following applies:
- If $predFlagL0Col[xColCb][yColCb]$ is equal to 0, $mvCol$, $refIdxCol$ and $listCol$ are set equal to $mvL1Col[xColCb][yColCb]$, $refIdxL1Col[xColCb][yColCb]$ and $L1$, respectively.
- Otherwise, if $predFlagL0Col[xColCb][yColCb]$ is equal to 1 and $predFlagL1Col[xColCb][yColCb]$ is equal to 0, $mvCol$, $refIdxCol$ and $listCol$ are set equal to $mvL0Col[xColCb][yColCb]$, $refIdxL0Col[xColCb][yColCb]$ and $L0$, respectively.
- Otherwise ($predFlagL0Col[xColCb][yColCb]$ is equal to 1 and $predFlagL1Col[xColCb][yColCb]$ is equal to 1), the following assignments are made:
 - If $NoBackwardPredFlag$ is equal to 1, $mvCol$, $refIdxCol$ and $listCol$ are set equal to $mvLXCol[xColCb][yColCb]$, $refIdxLXCol[xColCb][yColCb]$ and LX , respectively.
 - Otherwise, $mvCol$, $refIdxCol$ and $listCol$ are set equal to $mvLNCol[xColCb][yColCb]$, $refIdxLNCol[xColCb][yColCb]$ and LN , respectively, with N being the value of $collocated_from_l0_flag$.
- Otherwise ($sbFlag$ is equal to 1), the following applies:

- If $\text{PredFlagLXCol}[x\text{ColCb}][y\text{ColCb}]$ is equal to 1, mvCol , refIdxCol , and listCol are set equal to $\text{mvLXCol}[x\text{ColCb}][y\text{ColCb}]$, $\text{refIdxLXCol}[x\text{ColCb}][y\text{ColCb}]$, and LX , respectively, $\text{availableFlagLXCol}$ is set to 1.
- Otherwise ($\text{PredFlagLXCol}[x\text{ColCb}][y\text{ColCb}]$ is equal to 0), the following applies:
 - If $\text{NoBackwardPredFlag}$ is equal to 1 and $\text{PredFlagLYCol}[x\text{ColCb}][y\text{ColCb}]$ is equal to 1, mvCol , refIdxCol , and listCol are set to $\text{mvLYCol}[x\text{ColCb}][y\text{ColCb}]$, $\text{refIdxLYCol}[x\text{ColCb}][y\text{ColCb}]$ and LY , respectively, with Y being equal to $!X$ where X being the value of X this process is invoked for. $\text{availableFlagLXCol}$ is set equal to 1.
 - Otherwise, both components of mvLXCol are set equal to 0 and $\text{availableFlagLXCol}$ is set equal to 0.

...

[00150] 5.3 Embodiment #3

[00151] The working draft specified in JVET-O2001-vE may be changed as below.

8.5.6.6.2 Default weighted sample prediction process

...

Variables shift1 , shift2 , offset1 , offset2 , and offset3 are derived as follows:

- The variable shift1 is set equal to $\text{Max}(2, 14 - \text{bitDepth})$ and the variable shift2 is set equal to $\text{Max}(3, 15 - \text{bitDepth})$.
- The variable offset1 is set equal to $1 \ll (\text{shift1} - 1)$.
- The variable offset2 is set equal to $1 \ll (\text{shift2} - 1)$.

The variable offset3 is set equal to $1 \ll (\text{shift2} + I[[2]])$.

...

- Otherwise (predFlagL0 is equal to 1 and predFlagL1 is equal to 1), the following applies:
 - If bcwIdx is equal to 0 or $\text{ciip_flag}[xCb][yCb]$ is equal to 1, the prediction sample values are derived as follows:

$$\text{pbSamples}[x][y] = \text{Clip3}(0, (1 \ll \text{bitDepth}) - 1, \quad (8-823)$$

$$(\text{predSamplesL0}[x][y] + \text{predSamplesL1}[x][y] + \text{offset2}) \gg \text{shift2}$$

- Otherwise (bcwIdx is not equal to 0 and $\text{ciip_flag}[xCb][yCb]$ is equal to 0), the following applies:
 - The variable $w1$ is set equal to $\text{bcwWLut}[\text{bcwIdx}]$ with $\text{bcwWLut}[k] = \{4, 5, 3, 10, -2\}$.
 - The variable $w0$ is set equal to $(8 - w1)$.

- The prediction sample values are derived as follows.

$$\begin{aligned} \text{pbSamples}[x][y] &= \text{Clip3}(0, (1 \ll \text{bitDepth}) - 1, \\ & (w0 * \text{predSamplesL0}[x][y] + w1 * \text{predSamplesL1}[x][y] + \text{offset3}) \gg (\text{shift2} + [[3][2])) \end{aligned} \quad (8-824)$$

...

Alternatively, the following may apply:

...

Variables shift1, shift2, offset1, offset2, and offset3 are derived as follows:

- The variable shift1 is set equal to $\text{Max}(2, 14 - \text{bitDepth})$ [[and the variable shift2 is set equal to $\text{Max}(3, 15 - \text{bitDepth})$]].
- The variable offset1 is set equal to $1 \ll (\text{shift1} - 1)$.
- The variable offset2 is set equal to $1 \ll (\text{shift1} + [[2] - 1])$.

The variable offset3 is set equal to $1 \ll (\text{shift}[[2]] + 2)$.

...

- Otherwise (predFlagL0 is equal to 1 and predFlagL1 is equal to 1), the following applies:
 - If bcwIdx is equal to 0 or ciip_flag[xCb][yCb] is equal to 1, the prediction sample values are derived as follows:

$$\text{pbSamples}[x][y] = \text{Clip3}(0, (1 \ll \text{bitDepth}) - 1, \quad (8-823)$$

$$(\text{predSamplesL0}[x][y] + \text{predSamplesL1}[x][y] + \text{offset2}) \gg (\text{shift1} + 1)[[2]])$$

- Otherwise (bcwIdx is not equal to 0 and ciip_flag[xCb][yCb] is equal to 0), the following applies:
 - The variable w1 is set equal to bcwWLut[bcwIdx] with $\text{bcwWLut}[k] = \{4, 5, 3, 10, -2\}$.
 - The variable w0 is set equal to $(8 - w1)$.
 - The prediction sample values are derived as follows.

$$\begin{aligned} \text{pbSamples}[x][y] &= \text{Clip3}(0, (1 \ll \text{bitDepth}) - 1, \\ & (w0 * \text{predSamplesL0}[x][y] + w1 * \text{predSamplesL1}[x][y] + \text{offset3}) \gg (\text{shift1} + 3[[2+3]])) \end{aligned} \quad (8-824)$$

...

Alternatively, the following may apply:

...

Variables shift1, shift2, offset1, offset2, and offset3 are derived as follows:

- The variable shift1 is set equal to $\text{Max}(2, 14 - \text{bitDepth})$ and the variable shift2 is set equal to $\text{Max}(3, 15 - \text{bitDepth})$.
- The variable offset1 is set equal to $1 \ll (\text{shift1} - 1)$.
- The variable offset2 is set equal to $1 \ll (\text{shift2} - 1)$.

- [[The variable offset3 is set equal to $1 \ll (\text{shift2} + 2)$.]]
- ...
- Otherwise (predFlagL0 is equal to 1 and predFlagL1 is equal to 1), the following applies:
 - If bcwIdx is equal to 0 or ciip_flag[xCb][yCb] is equal to 1, the prediction sample values are derived as follows:

$$\text{pbSamples}[x][y] = \text{Clip3}(0, (1 \ll \text{bitDepth}) - 1, \quad (8-823)$$

$$(\text{predSamplesL0}[x][y] + \text{predSamplesL1}[x][y] + \text{offset2}) \gg \text{shift2}$$

- Otherwise (bcwIdx is not equal to 0 and ciip_flag[xCb][yCb] is equal to 0), the following applies:
 - The variable w1 is set equal to bcwWLut[bcwIdx] with bcwWLut[k] = { 4, 5, 3, 10, -2 }.
 - The variable w0 is set equal to (8 - w1).
 - The prediction sample values are derived as follows.

$$\text{pbSamples}[x][y] = \text{Clip3}(0, (1 \ll \text{bitDepth}) - 1, \quad (8-824)$$

$$(w0 * ((\text{predSamplesL0}[x][y] + \text{offset1}) \gg \text{shift1}) + w1 * ((\text{predSamplesL1}[x][y] + \text{offset1}) \gg \text{shift1}) + 4) \gg (8[(\text{shift2} + 3)])$$

...

[00152] 5.4 Embodiment #4

[00153] The working draft specified in JVET-O2001-vE may be changed as below.

8.5.6.6.2 Default weighted sample prediction process

...

- Otherwise (bcwIdx is not equal to 0 and ciip_flag[xCb][yCb] is equal to 0), the following applies:

The variable w1 is set equal to bcwWLut[bcwIdx] with bcwWLut[k] =

{ 4, 5, 3, 10, 2[[-2]] } / { 4, 5, 3, 10, 1 } / { 4, 3, 5, 10, 2 } / { 4, 3, 5, 10, 1 } / { 4, 5, 3, 10, -1 }.

...

[00154] 5.5 Embodiment #5

[00155] The working draft specified in JVET-O2001-vE may be changed as below.

8.5.2.7 Derivation process for merge motion vector difference

...

$$\text{mMvdL1}[0] = \text{Clip3}(-2^{[(15)]/7}, 2^{[(15)]/7} - 1, (\text{distScaleFactor} * \text{mMvdL0}[0] + \quad (8-394)$$

$$128 - (\text{distScaleFactor} * \text{mMvdL0}[0] \geq 0)) \gg 8)$$

```

mMvdL1[ 1 ] = Clip3( -2[15], 2[15] - 1, (distScaleFactor * mMvdL0[ 1 ] + (8-395)
128 - ( distScaleFactor * mMvdL0[ 1 ] >= 0 ) ) >> 8 )
...
mMvdL0[ 0 ] = Clip3( -2[15], 2[15] - 1, (distScaleFactor * mMvdL1[ 0 ] + (8-404)
128 - (distScaleFactor * mMvdL1[ 0 ] >= 0 ) ) >> 8 )
mMvdL0[ 1 ] = Clip3( -2[15], 2[15] - 1, , (distScaleFactor * mMvdL1[ 1 ] + (8-405)
128 - (distScaleFactor * mMvdL1[ 1 ] >= 0 ) ) >> 8 )
...

```

[00156] 5.6 Embodiment #6

[00157] The working draft specified in JVET-O2001-vE may be changed as below.

7.3.8.5 Coding unit syntax

coding_unit(x0, y0, cbWidth, cbHeight, cqtDepth, treeType, modeType) {	Descriptor
chType = treeType = DUAL_TREE_CHROMA ? 1 : 0	
if(slice_type != I sps_ibc_enabled_flag sps_palette_enabled_flag) {	
if(treeType != DUAL_TREE_CHROMA && !((cbWidth == 4 && cbHeight == 4) modeType == MODE_TYPE_INTRA) && !(sps_ibc_enabled_flag && cbWidth <= 64 && cbHeight <= 64))	
cu_skip_flag[x0][y0]	ae(v)
if(cu_skip_flag[x0][y0] == 0 && slice_type != I && !(cbWidth == 4 && cbHeight == 4) && modeType == MODE_TYPE_ALL)	
pred_mode_flag	ae(v)
if(((slice_type == I && cu_skip_flag[x0][y0] == 0) (slice_type != I && (CuPredMode[chType][x0][y0] != MODE_INTRA (cbWidth == 4 && cbHeight == 4 && cu_skip_flag[x0][y0] == 0)))) && cbWidth <= 64 && cbHeight <= 64 && modeType != MODE_TYPE_INTER && sps_ibc_enabled_flag && treeType != DUAL_TREE_CHROMA)	
pred_mode_ibc_flag	ae(v)
}	
if((((slice_type == I (cbWidth == 4 && cbHeight == 4) sps_ibc_enabled_flag) && CuPredMode[x0][y0] == MODE_INTRA) (slice_type != I && !(cbWidth == 4 && cbHeight == 4) && !sps_ibc_enabled_flag && CuPredMode[x0][y0] != MODE_INTRA)) && sps_palette_enabled_flag && cbWidth <= 64 && cbHeight <= 64 && cu_skip_flag[x0][y0] == 0 && modeType != MODE_INTER)	
pred_mode_plt_flag	ae(v)
{ }	

[00158] FIG. 14A is a block diagram of a video processing apparatus 1400. The apparatus 1400 may be used to implement one or more of the methods described herein. The apparatus 1400

may be embodied in a smartphone, tablet, computer, Internet of Things (IoT) receiver, and so on. The apparatus 1400 may include one or more processors 1402, one or more memories 1404 and video processing hardware 1406. The processor(s) 1402 may be configured to implement one or more methods (including, but not limited to, methods 800, 900, 1000, 1100, 1200 and 1300) described in the present document. The memory (memories) 1404 may be used for storing data and code used for implementing the methods and techniques described herein. The video processing hardware 1406 may be used to implement, in hardware circuitry, some techniques described in the present document. In some embodiments, the hardware 1406 may be completely or partially implemented in the processor 1402, e.g., a graphics co-processor.

[00159] FIG. 14B is a block diagram showing an example video processing system 1410 in which various techniques disclosed herein may be implemented. Various implementations may include some or all of the components of the system 1410. The system 1410 may include input 1412 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 1412 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.

[00160] The system 1410 may include a coding component 1414 that may implement the various coding or encoding methods described in the present document. The coding component 1414 may reduce the average bitrate of video from the input 1412 to the output of the coding component 1414 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 1414 may be either stored, or transmitted via a communication connected, as represented by the component 1416. The stored or communicated bitstream (or coded) representation of the video received at the input 1412 may be used by the component 1418 for generating pixel values or displayable video that is sent to a display interface 1420. 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.

[00161] Examples of a peripheral bus interface or a display interface may include universal serial bus (USB) or high definition multimedia interface (HDMI) or display port, 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.

[00162] In some embodiments, the video coding methods may be implemented using an apparatus that is implemented on a hardware platform as described with respect to FIG. 14A or 14B.

[00163] The following listing provides embodiments that can address the technical problems described in the present document, among other problems. The first set of clauses describe certain features and aspects of the disclosed techniques in the previous section.

[00164] 1. A method for processing video, comprising: determining an availability of one or more collocated motion vectors based on a prediction mode of a collocated video block relative to a current video block; and performing, based on the one or more collocated motion vectors, a conversion between the current block and a bitstream representation of the current block, wherein an indication of the prediction mode comprises one bit that indicates whether the current video block is coded with an inter mode or a non-inter mode.

[00165] 2. A method for processing video, comprising: determining, for a conversion between a coded representation of a video block and the video block, a type of a collocated video block of the video block, wherein the type takes one of only two possible values; and performing the conversion based on the determining.

[00166] 3. The method of clause 1, wherein the two possible values include a first value indicating that the collocated video block is inter-coded and a second value that indicates that the collocated video block is coded using a mode other than inter-coded.

[00167] 4. The method of any of clauses 1 to 3, wherein the determining is identical when the current video block is coded with palette prediction mode and when the current video block is coded with an intra or intra block copy (IBC) prediction mode.

[00168] 5. The method of any of clauses 1 to 3, wherein the one or more collocated motion vectors is determined to be unavailable when the collocated video block is coded with a non-inter prediction mode.

- [00169] 6. The method of any of clauses 1 to 3, wherein the one or more collocated motion vectors is determined to be available and comprises a default motion vector when the collocated video block is coded with a non-inter prediction mode.
- [00170] 7. The method of clause 5 or 6, wherein the non-inter prediction mode is an intra prediction mode, a palette prediction mode or an intra block copy (IBC) prediction mode.
- [00171] 8. The method of any of clauses 1 to 3, wherein the determining is based on a reference index or a reference list of the collocated video block.
- [00172] 9. The method of clause 8, wherein the reference index is a predetermined value.
- [00173] 10. The method of clause 8, wherein the reference index excludes a predetermined value.
- [00174] 11. The method of clause 9 or 10, wherein the predetermined value is 0.
- [00175] 12. A method for video processing, comprising: deriving, based on a first weight table associated with a bi-prediction mode with a coding unit (CU)-level weight (BCW) process, a prediction sample for a current block, wherein the first weight table is asymmetric; and performing, based on the prediction sample, a conversion between the current block and a bitstream representation of the current block.
- [00176] 13. The method of clause 12, wherein entries in the first weight table are non-monotonically increasing.
- [00177] 14. The method of clause 12, wherein the BCW process is further based on a second weight table different from the first weight table.
- [00178] 15. The method of clause 12, wherein deriving the prediction sample comprises: converting an intermediate prediction sample to a first bit depth; applying a weight from the first weight table to the intermediate prediction sample to derive the prediction sample; and converting the prediction sample to a second bit depth.
- [00179] 16. The method of clause 15, wherein the second bit depth is a bit depth of a color component of the current video block.
- [00180] 17. A method for video processing, comprising: making a decision, based on a dimension of the current video block, regarding a selective signaling of an indication of a skip mode coding of the current video block in a bitstream representation of the current video block; and performing, based on the decision, a conversion between the current block and the bitstream representation of the current block.

[00181] 18. The method of clause 17, wherein the indication is signaled upon a determination that a slice type of a slice comprising the current video block is an I-slice, an `sps_ibc_enabled_flag` is set to true, and a height and a width of the current video block are less than or equal to N.

[00182] 19. The method of clause 17, wherein the indication is signaled upon a determination that the current video block is coded with an intra block copy (IBC) mode and a height and a width of the current video block are less than or equal to N.

[00183] 20. The method of clause 18 or 19, wherein $N = 64$.

[00184] 21. A method for video processing, comprising: making a decision, based on an application of a first coding tool to one or more blocks in a first color component of a current video block, regarding a selective application of a second coding tool to at least one block of a second color component of the current video block; and performing, based on the decision, a conversion between the current block and the bitstream representation of the current block.

[00185] 22. The method of clause 21, wherein the second coding tool is applied upon a determination that the second coding tool is identical to the first coding tool.

[00186] 23. The method of clause 21, wherein the second coding tool is not applied upon a determination that the second coding tool is different from the first coding tool.

[00187] 24. The method of clause 21, wherein the decision is further based on an indication in the bitstream representation.

[00188] 25. The method of any of clauses 21 to 24, wherein the first color component is a luma component and the second color component is a chroma component.

[00189] 26. The method of any of clauses 21 to 24, wherein the first color component is a first chroma color component and the second color component is a second chroma color component.

[00190] 27. The method of any of clauses 21 to 26, wherein the first coding tool and the second coding tool are one of temporal motion vector prediction (TMVP), alternative temporal motion vector prediction (ATMVP), a bi-prediction with a coding unit (CU)-level weight (BCW) process, a merge mode with motion vector differences (MMVD) or a position-dependent prediction combination (PDPC) process.

[00191] 28. A method for video processing, comprising: making a decision, based on a precision of a motion vector in a current video block, regarding a selective signaling of an indication for using an alternative half-pel interpolation filter instead of a default half-pel interpolation filter;

and performing, based on the decision, a conversion between the current block and the bitstream representation of the current block.

[00192] 29. The method of clause 28, wherein the default half-pel interpolation filter is used upon a determination that the current video block is coded with a merge mode with motion vector differences (MMVD) and a precision of a reconstructed motion vector is lower than half-pel.

[00193] 30. The method of any of clauses 1 to 29, wherein performing the conversion is further based on signaling in a decoder parameter set (DPS), a sequence parameter set (SPS), a picture parameter set (PPS), an adaptive parameter set (APS), a video parameter set (VPS), a sequence header, a picture header, a slice header, or a tile group header.

[00194] 31. The method of any of clauses 1 to 30, wherein performing the conversion comprises generating the bitstream representation from the current video block.

[00195] 32. The method of any of clauses 1 to 30, wherein performing the conversion comprises generating the current video block from the bitstream representation.

[00196] 33. A video decoding apparatus comprising a processor configured to implement a method recited in any one of clauses 1 to 32.

[00197] 34. A computer program product stored on a non-transitory computer readable media, the computer program product including program code for carrying out the method in any one of clauses 1 to 32.

[00198] The second set of clauses describe certain features and aspects of the disclosed techniques in the previous section (e.g., items 1 and 2).

[00199] 1. A method of video processing, comprising: making a first determination about whether a collocated video block of a video block of a video is coded using an inter coding mode or a non-inter coding mode; making a second determination, based on the first determination, about availability of motion vectors of the collocated video block or a mode of derivation of motion vectors for the collocated video block; and performing a conversion between the video block and a coded representation of the video based on the second determination.

[00200] 2. The method of clause 1, wherein the non-inter coding mode is an intra mode, a palette mode, or an intra block copy (IBC) mode.

- [00201] 3. The method of clause 1, wherein the making of the second determination is performed such that the second determination is identical for the collocated video block coded with a palette mode and for the collocated video block coded with an intra mode.
- [00202] 4. The method of clause 1, wherein the making of the second determination is performed such that the second determination is identical for the collocated video block coded with a palette mode and for the collocated video block coded with an intra block copy mode.
- [00203] 5. The method of clause 1, wherein the making of the second determination is performed such that the second determination is identical for the collocated video block coded with an intra mode and for the collocated video block coded with an intra block copy mode.
- [00204] 6. The method of clause 1, wherein the making a first determination comprises: generating one bit to indicate whether the collocated video block is coded using the inter coding mode or the non-inter coding mode.
- [00205] 7. The method of clause 1, wherein the making of the second determination determines motion vectors of the collocated video block to be unavailable based on the first determination that the collocated video block is coded using the non-inter coding mode.
- [00206] 8. The method of clause 1, wherein the making of the second determination determines motion vectors of the collocated video block to be available based on the first determination that the collocated video block is coded using the non-inter coding mode, and wherein a default motion vector is assigned as the motion vectors of the collocated video block.
- [00207] 9. The method of clause 1, wherein another video block instead of the collocated video block is checked in case that the collocated video block is coded using the non-inter coding mode.
- [00208] 10. The method of clause 9, wherein the another video block is a neighboring block that is a nearest inter coded block that is left, right, above, or below the collocated video block.
- [00209] 11. A method for processing video, comprising: determining, for a conversion between a video block of a video and a coded representation of the video, one or more collocated motion vectors based on a reference index or a reference list of a collocated video block of the video block; and performing the conversion based on the determining.
- [00210] 12. The method of clause 11, wherein the one or more collocated motion vectors are derived in case that the reference index of the reference list of the collocated video block is not equal to a predetermined value.

[00211] 13. The method of clause 11, wherein the one or more collocated motion vectors are derived in case that the reference index of the reference list of the collocated video block is equal to a predetermined value.

[00212] 14. The method of clause 12 or 13, wherein the predetermined value is 0.

[00213] 15. The method of clause 11, wherein in case that the collocated video block is coded in a non-inter coding mode or a reference picture of the collocated video block does not satisfy a given condition, the one or more collocated motion vectors are set to be unavailable.

[00214] 16. The method of clause 11, wherein in case that the collocated video block is coded in a non-inter coding mode or a reference picture of the collocated video block does not satisfy a given condition, the one or more collocated motion vectors are set to a default value.

[00215] 17. The method of any of clauses 1 to 16, wherein the conversion includes encoding the video into the coded representation.

[00216] 18. The method of any of clauses 1 to 16, wherein the conversion includes decoding the coded representation to generate the video.

[00217] 19. A video processing apparatus comprising a processor configured to implement a method recited in any one or more of clauses 1 to 16.

[00218] 20. 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.

[00219] 21. A computer readable medium that stores a coded representation or a bitstream representation generated according to any of the above described methods.

[00220] The third set of clauses describe certain features and aspects of the disclosed techniques in the previous section (e.g., items 3 to 7).

[00221] 1. A method of video processing (e.g., method 1530 shown in FIG. 15C), comprising: deriving (1532), for a conversion between a current block of a video and a coded representation of the video, a prediction block for the current block that is determined according to a bi-directional coding unit level weighting (BCW) tool in which weights are used to determine a weighted sum of two initial prediction blocks used to determine the prediction block; and performing (1534) the conversion using the prediction block, wherein the deriving of the prediction block comprises: applying the weights to two intermediate prediction samples that respectively belong to the two initial prediction blocks to derive a final prediction sample; and performing a right shift operation for the final prediction sample to convert a bit depth of the

final prediction sample, wherein the right shift operation is $\text{pbSample} \gg (\text{shift1} + 3)$, wherein pbSample represents the final prediction sample, shift1 is set equal to $\text{Max}(2, 14 - \text{bitDepth})$; and determining the prediction block based on the final prediction sample.

[00222] 2. The method of clause 1, wherein, before the performing the right shift operation, an offset is added to the final prediction sample.

[00223] 3. The method of clause 1, wherein, the offset is $1 \ll (\text{shift1} + 2)$.

[00224] 4. The method of clause 3, wherein the final prediction sample, $\text{pbSamples}[x][y]$, is derived such that $\text{pbSamples}[x][y] = \text{Clip3}(0, (1 \ll \text{bitDepth}) - 1, (w_0 * \text{predSamplesL0}[x][y] + w_1 * \text{predSamplesL1}[x][y] + \text{offset3}) \gg (\text{shift1} + 3))$, wherein w_0 and w_1 are the weights, and $\text{predSamplesL0}[x][y]$ and $\text{predSamplesL1}[x][y]$ are the two intermediate prediction samples.

[00225] 5. The method of clause 1, wherein at least one of the weights belongs to a weight table, wherein entries of the weight table are non-monotonically increasing.

[00226] 6. A method of video processing, comprising: deriving, for a conversion between a current block of a video and a coded representation of the video, a prediction block for the current block that is determined according to a bi-directional coding unit level weighting (BCW) tool in which weights are used to determine a weighted sum of two initial prediction blocks used to determine the prediction block; and performing the conversion using the prediction block, wherein the deriving of the prediction block comprises: applying the weights to two intermediate prediction sample that respectively belong to the two initial prediction blocks to derive a final prediction sample; and adding an offset to the final prediction sample, wherein the offset is $1 \ll (\text{shift1} + 2)$, wherein shift1 is set equal to $\text{Max}(2, 14 - \text{bitDepth})$; and determining the prediction block based on the final prediction sample.

[00227] 7. The method of clause 6, wherein the determining the prediction block based on the final prediction sample comprises: performing a right shift operation for the final prediction sample to convert a bit depth of the final prediction sample.

[00228] 8. The method of clause 1 or 7, wherein the right shift operation is $\text{pbSample} \gg (\text{shift1} + 3)$, wherein pbSample represents the final prediction sample.

[00229] 9. The method of clause 8, wherein the final prediction sample, $\text{pbSamples}[x][y]$, is derived such that $\text{pbSamples}[x][y] = \text{Clip3}(0, (1 \ll \text{bitDepth}) - 1, (w_0 * \text{predSamplesL0}[x][y] + w_1 * \text{predSamplesL1}[x][y] + \text{offset3}) \gg (\text{shift1} + 3))$, wherein

wherein w_0 and w_1 are the weights, and $\text{predSamplesL0}[x][y]$ and $\text{predSamplesL1}[x][y]$ are the two intermediate prediction samples.

[00230] 10. The method of clause 6, wherein at least one of the weights belongs to a weight table, wherein entries of the weight table are non-monotonically increasing.

[00231] 11. A method of video processing (e.g., method 1540 shown in FIG. 15D), comprising: deriving (1542), for a conversion between a current block of a video and a coded representation of the video, a motion vector difference for a merge mode motion vector predictor for the current block according to a rule; and performing (1544) the conversion based on the deriving, wherein the rule specifies whether to clip the motion vector difference to a same range as a range used for clipping motion vectors.

[00232] 12. The method of clause 11, wherein the rule specifies that the motion vector difference is clipped to the same range as the range used for clipping motion vectors.

[00233] 13. The method of clause 11, wherein the range used for clipping motion vectors is 18 bits.

[00234] 14. The method of clause 13, wherein the range used for clipping motion vectors is $[-2^{17}, 2^{17} - 1]$.

[00235] 15. The method of clause 11, wherein the rule specifies that the motion vector difference is clipped to $[-2^{17}+1, 2^{17} - 1]$.

[00236] 16. The method of clause 11, wherein the rule specifies not to clip the motion vector difference.

[00237] 17. The method of clause 16, wherein the rule specifies to clip a result obtained by adding the motion vector difference to a motion vector prediction candidate to 18 bits.

[00238] 18. A method of video processing, comprising: deriving, for a conversion between a current block of a video and a coded representation of the video, a prediction block for the current block that is determined according to a bi-directional coding unit level weighting (BCW) tool in which weights are used to determine a weighted sum of two initial prediction blocks used to determine the prediction block, wherein at least one of the weights belongs to a weight table, wherein the weight table is organized according to a rule; and performing the conversion using the prediction block, wherein the rule specifies that entries of the weight table are non-monotonically increasing.

[00239] 19. The method of clause 18, wherein the weight table includes an $(i+1)$ th entry whose

value is smaller than that of i-th entry.

[00240] 20. A method of video processing, comprising: deriving, for a conversion between a current block of a video and a coded representation of the video, a prediction block for the current block that is determined according to a bi-directional coding unit level weighting (BCW) tool in which weights are used to determine a weighted sum of two initial prediction blocks used to determine the prediction block, wherein at least one of the weights belongs to a weight table, and the weight table is selected from multiple weight tables; and performing the conversion using the prediction block.

[00241] 21. The method of clause 20, wherein the multiple weight tables include {4, 5, 3, 10, 2}, {4, 3, 5, 10, 2}, {4, 5, 3, 10, 1}, {4, 3, 5, 10, 1}, or {4, 5, 3, 10, -1}.

[00242] 22. A method of video processing, comprising: deriving, for a conversion between a current block of a video and a coded representation of the video, a prediction block for the current block that is determined according to a bi-directional coding unit level weighting (BCW) tool in which weights are used to determine a weighted sum of two initial prediction blocks used to determine the prediction block; and performing the conversion using the prediction block, wherein the deriving of the prediction block comprises: converting an intermediate prediction sample to a first bit depth; applying the weights to the intermediate prediction sample to derive a final prediction sample; and converting the final prediction sample to a second bit depth.

[00243] 23. The method of clause 22, wherein the second bit depth is a bit depth of a color component of the current block.

[00244] 24. The method of clause 22, wherein the first bit depth is not equal to a bit depth of the intermediate prediction sample.

[00245] 25. The method of clause 22, wherein the deriving of the prediction block comprises a right shift operation that converts a sample from a higher bit depth to a lower bit depth or left shift operation that converts the sample from the lower bit depth to the higher bit depth.

[00246] 26. The method of clause 25, wherein an offset is added before the right shift operation or the left shift operation.

[00247] 27. The method of clause 22, wherein the final prediction sample, $pbSamples[x][y]$, is derived such that $pbSamples[x][y] = Clip3(0, (1 \ll bitDepth) - 1, (w0 * ((predSamplesL0[x][y] + offset1) \gg shift1) + w1 * ((predSamplesL1[x][y] + offset1) \gg shift1) + 4) \gg 3)$, where $w0$ and $w1$ are the weights, $shift1$ is set equal to $Max(2, 14 -$

bitDepth) and offset1 is set equal to $1 \ll (\text{shift1} - 1)$.

[00248] 28. A method of video processing (method 1550 shown in FIG. 15E), comprising: deriving (1552), for a conversion between a current block of a video and a coded representation of the video, a prediction block for the current block that is determined according to a bi-directional coding unit level weighting (BCW) tool in which a first weight and a second weight are used to determine a weighted sum of two initial prediction blocks used to determine the prediction block, wherein at least one of the first weight and the second weight is included in a weight table, wherein the weight table is organized according to a rule; and performing the conversion using the prediction block, wherein the rule specifies that the weight table is asymmetric.

[00249] 29. The method of clause 28, wherein the weight table includes an entry whose value corresponds to the first weight, W , and does not include any entry whose value corresponds to the second weight, $(1-W)$, whereby W is in the range of $[a, b]$ and a sum of a and b , $(a+b)$, is equal to 1.

[00250] 30. The method of clause 28, wherein the weight table includes an entry whose value corresponds to the first weight, W , and does not include any entry whose value corresponds to the second weight, $(2^N - W)$, in case that a prediction sample is generated by $(W_0 * P_0 + W_1 * P_1) \gg N$, whereby W_0 and W_1 are the first weight and the second weight applied to the initial two prediction blocks P_0 and P_1 , respectively; and $(W_0 + W_1)$ is equal to $(1 \ll N)$.

[00251] 31. The method of any of clauses 1 to 30, wherein the performing of the conversion includes generating the coded representation from the current video block.

[00252] 32. The method of any of clauses 1 to 30, wherein the performing of the conversion includes generating the current video block from the coded representation.

[00253] 33. A video processing apparatus comprising a processor configured to implement a method recited in any one or more of clauses 1 to 32.

[00254] 34. 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 32.

[00255] 35. A computer readable medium that stores a coded representation or a bitstream representation generated according to any of the above described methods.

[00256] From the foregoing, it will be appreciated that specific embodiments of the presently disclosed technology have been described herein for purposes of illustration, but that various

modifications may be made without deviating from the scope of the invention. Accordingly, the presently disclosed technology is not limited except as by the appended claims.

[00257] Implementations of the subject matter and the functional operations described in this patent document can be implemented in various systems, digital electronic circuitry, or in computer software, firmware, or hardware, including the structures disclosed in this specification and their structural equivalents, or in combinations of one or more of them. Implementations of the subject matter described in this specification can be implemented as one or more computer program products, i.e., one or more modules of computer program instructions encoded on a tangible and non-transitory computer readable medium for execution by, or to control the operation of, data processing apparatus. The computer readable medium can be a machine-readable storage device, a machine-readable storage substrate, a memory device, a composition of matter effecting a machine-readable propagated signal, or a combination of one or more of them. The term “data processing unit” or “data processing apparatus” encompasses all apparatus, devices, and machines for processing data, including by way of example a programmable processor, a computer, or multiple processors or computers. The apparatus can include, in addition to hardware, code that creates an execution environment for the computer program in question, e.g., code that constitutes processor firmware, a protocol stack, a database management system, an operating system, or a combination of one or more of them.

[00258] 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.

[00259] The processes and logic flows described in this specification can be performed by one or more programmable processors executing one or more computer programs to perform functions

by operating on input data and generating output. The processes and logic flows can also be performed by, and apparatus can also be implemented as, special purpose logic circuitry, e.g., an FPGA (field programmable gate array) or an ASIC (application specific integrated circuit).

[00260] Processors suitable for the execution of a computer program include, by way of example, both general and special purpose microprocessors, and any one or more processors of any kind of digital computer. Generally, a processor will receive instructions and data from a read only memory or a random access memory or both. The essential elements of a computer are a processor for performing instructions and one or more memory devices for storing instructions and data. Generally, a computer will also include, or be operatively coupled to receive data from or transfer data to, or both, one or more mass storage devices for storing data, e.g., magnetic, magneto optical disks, or optical disks. However, a computer need not have such devices. Computer readable media suitable for storing computer program instructions and data include all forms of nonvolatile memory, media and memory devices, including by way of example semiconductor memory devices, e.g., EPROM, EEPROM, and flash memory devices. The processor and the memory can be supplemented by, or incorporated in, special purpose logic circuitry.

[00261] It is intended that the specification, together with the drawings, be considered exemplary only, where exemplary means an example. As used herein, the singular forms “a”, “an” and “the” are intended to include the plural forms as well, unless the context clearly indicates otherwise. Additionally, the use of “or” is intended to include “and/or”, unless the context clearly indicates otherwise.

[00262] While this patent document contains many specifics, these should not be construed as limitations on the scope of any invention or of what may be claimed, but rather as descriptions of features that may be specific to particular embodiments of particular inventions. Certain features that are described in this patent document in the context of separate embodiments can also be implemented in combination in a single embodiment. Conversely, various features that are described in the context of a single embodiment can also be implemented in multiple embodiments separately or in any suitable subcombination. Moreover, although features may be described above as acting in certain combinations and even initially claimed as such, one or more features from a claimed combination can in some cases be excised from the combination, and the claimed combination may be directed to a subcombination or variation of a subcombination.

[00263] 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.

[00264] 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.