

ADAPTIVE COLOUR TRANSFORM RELATED SIGNALLING FOR BOTH OF CU LEVEL AND TU LEVEL

CROSS REFERENCE TO RELATED APPLICATION

[1] The present application claims priority to provisional application US 63/037,170 filed on June 10, 2020 and application US 17/319,328 filed on May 13, 2021, which are hereby expressly incorporated by reference, in their entirety, into the present application.

BACKGROUND

1. Field

[2] The present disclosure relates to signaling of a coding unit (CU) level enable flag and transform unit (TU) level luma. coded flag for the coded block with an adaptive color transform (ACT) mode.

2. Description of Related Art

[3] ITU-T VCEG (Q6/16) and ISO/IEC MPEG (JTC 1/SC 29/WG 11) published the H.265/HEVC (High Efficiency Video Coding) standard in 2013 (version 1) 2014 (version 2) 2015 (version 3) and 2016 (version 4). In 2015, these two standard organizations jointly formed the JVET (Joint Video Exploration Team) to explore the potential of developing the next video coding standard beyond HEVC. In October 2017, they issued the Joint Call for Proposals on Video Compression with Capability beyond HEVC (CfP). By February 15, 2018, total 22 CfP responses on standard dynamic range (SDR), 12 CfP responses on high dynamic range (HDR), and 12 CfP responses on 360 video categories were submitted, respectively. In April 2018, all received CfP responses were evaluated in the 122 MPEG / 10th JVET meeting. As a result of this meeting, JVET formally launched the standardization process of next-generation video coding beyond

HEVC. The new standard was named Versatile Video Coding (VVC), and JVET was renamed as Joint Video Expert Team.

[4] However, there are technical problems such as if a coded CU block does not have a coefficient, then the ACT mode signaling may be redundant or the CU with ACT mode should have one or more than one coefficient in the coded CU block. For an inter block with ACT mode, if the `cu_coded_flag` should be 1 to represent that the CU has at least one coefficient in a transform unit, then there is no corresponding constraint for the intra CU with ACT mode, and only the intra block with ACT mode should be inferred to 1 if the TU coded flag of chrominance channels are both zero. Also, the `cu_act_enabled_flag` should be signaled twice based on different prediction mode of the current CU block. As such, there are described herein technical solutions to such problems for example.

SUMMARY

[5] According to exemplary embodiments, there is included a method and apparatus comprising memory configured to store computer program code and a processor or processors configured to access the computer program code and operate as instructed by the computer program code. The computer program code includes first obtaining code configured to cause the at least one processor to obtain video data, second obtaining code configured to cause the at least one processor to obtain a coding unit (CU) block of the video data, first determining code configured to cause the at least one processor to determine whether a flag of the CU block is set to a predetermined flag condition, second determining code configured to cause the at least one processor to determine whether a tree type of the CU block is set to a predetermined tree type, third determining code configured to cause the at least one processor to determine whether to signal an adaptive color transform (ACT) flag based on any of whether the flag of the CU block

is set to the predetermined flag condition and whether the tree type of the CU block is set to the predetermined tree type, and coding code configured to cause the at least one processor to code the video data based on a whether the ACT flag is signaled.

[6] According to exemplary embodiment, determining whether to signal the ACT flag is based on only whether the flag of the CU block is set to the predetermined flag condition.

[7] According to exemplary embodiment, determining whether to signal the ACT flag is based on both of whether the flag of the CU block is set to the predetermined flag condition and whether the tree type of the CU block is set to the predetermined tree type.

[8] According to exemplary embodiment, the predetermined tree type indicates a single tree type rather than a dual tree type.

[9] According to exemplary embodiment, determining whether to signal an adaptive color transform (ACT) flag is implemented regardless of whether a prediction mode of the CU is an intra mode.

[10] According to exemplary embodiment, the computer program code further includes fourth determining code configured to cause the at least one processor to determine whether transform unit (TU) coded flags are both zero and whether the CU is coded with an ACT mode.

[11] According to exemplary embodiment, the TU coded flags are flags of chrominance channels.

[12] According to exemplary embodiment, the computer program code further comprises fifth determining code configured to cause the at least one processor to determine whether a TU coded flag of luminance is to be inferred to be 1 based on determining that the TU coded flags are both zero and that the CU is coded with the ACT mode.

[13] According to exemplary embodiments, determining the TU coded flag of luminance is to be inferred to be 1 is implemented regardless of whether a prediction mode of the CU is an intra mode.

[14] According to exemplary embodiments, coding the video data is further based on determining whether the TU coded flag of luminance is to be inferred to be 1.

BRIEF DESCRIPTION OF THE DRAWINGS

[15] Further features, nature, and various advantages of the disclosed subject matter will be more apparent from the following detailed description and the accompanying drawings in which:

[16] Fig. 1 is a simplified illustration of a schematic diagram in accordance with embodiments.

[17] Fig. 2 is a simplified illustration of a schematic diagram in accordance with embodiments.

[18] Fig. 3 is a simplified illustration of a schematic diagram in accordance with embodiments.

[19] Fig. 4 is a simplified illustration of a schematic diagram in accordance with embodiments.

[20] Fig. 5 is a simplified illustration of a diagram in accordance with embodiments.

[21] Fig. 6 is a simplified illustration of a diagram in accordance with embodiments.

[22] Fig. 7 is a simplified illustration of a diagram in accordance with embodiments.

[23] Fig. 8 is a simplified illustration of a diagram in accordance with embodiments.

[24] Fig. 9A is a simplified illustration of a diagram in accordance with embodiments.

- [25] Fig. 9B is a simplified illustration of a diagram in accordance with embodiments.
- [26] Fig. 10 is a simplified illustration of a flowchart in accordance with embodiments.
- [27] Fig. 11 is a simplified illustration of a flowchart in accordance with embodiments.
- [28] Fig. 12 a simplified illustration of a flowchart in accordance with embodiments.
- [29] Fig. 13 a simplified illustration of a diagram in accordance with embodiments.
- [30] Fig. 14 a simplified illustration of a diagram in accordance with embodiments.
- [31] Fig. 15 a simplified illustration of a schematic diagram in accordance with embodiments.

DETAILED DESCRIPTION

[32] The proposed features discussed below may be used separately or combined in any order. Further, the embodiments may be implemented by processing circuitry (e.g., one or more processors or one or more integrated circuits). In one example, the one or more processors execute a program that is stored in a non-transitory computer-readable medium.

[33] Fig. 1 illustrates a simplified block diagram of a communication system 100 according to an embodiment of the present disclosure. The communication system 100 may include at least two terminals 102 and 103 interconnected via a network 105. For unidirectional transmission of data, a first terminal 103 may code video data at a local location for transmission to the other terminal 102 via the network 105. The second terminal 102 may receive the coded video data of the other terminal from the network 105, decode the coded data and display the recovered video data. Unidirectional data transmission may be common in media serving applications and the like.

[34] Fig. 1 illustrates a second pair of terminals 101 and 104 provided to support bidirectional transmission of coded video that may occur, for example, during videoconferencing. For bidirectional transmission of data, each terminal 101 and 104 may code video data captured at a local location for transmission to the other terminal via the network 105. Each terminal 101 and 104 also may receive the coded video data transmitted by the other terminal, may decode the coded data and may display the recovered video data at a local display device.

[35] In Fig. 1, the terminals 101, 102, 103 and 104 may be illustrated as servers, personal computers and smart phones but the principles of the present disclosure are not so limited. Embodiments of the present disclosure find application with laptop computers, tablet computers, media players and/or dedicated video conferencing equipment. The network 105 represents any number of networks that convey coded video data among the terminals 101, 102, 103 and 104, including for example wireline and/or wireless communication networks. The communication network 105 may exchange data in circuit-switched and/or packet-switched channels. Representative networks include telecommunications networks, local area networks, wide area networks and/or the Internet. For the purposes of the present discussion, the architecture and topology of the network 105 may be immaterial to the operation of the present disclosure unless explained herein below.

[36] Fig. 2 illustrates, as an example for an application for the disclosed subject matter, the placement of a video encoder and decoder in a streaming environment. The disclosed subject matter can be equally applicable to other video enabled applications, including, for example, video conferencing, digital TV, storing of compressed video on digital media including CD, DVD, memory stick and the like, and so on.

[37] A streaming system 200 may include a capture subsystem 203, that can include a video source 201, for example a digital camera, creating, for example, an uncompressed video sample stream 213. That sample stream 213 may be emphasized as a high data volume when compared to encoded video bitstreams and can be processed by an encoder 202 coupled to the camera 201. The encoder 202 can include hardware, software, or a combination thereof to enable or implement aspects of the disclosed subject matter as described in more detail below. The encoded video bitstream 204, which may be emphasized as a lower data volume when compared to the sample stream, can be stored on a streaming server 205 for future use. One or more streaming clients 212 and 207 can access the streaming server 205 to retrieve copies 208 and 206 of the encoded video bitstream 204. A client 212 can include a video decoder 211 which decodes the incoming copy of the encoded video bitstream 208 and creates an outgoing video sample stream 210 that can be rendered on a display 209 or other rendering device (not depicted). In some streaming systems, the video bitstreams 204, 206 and 208 can be encoded according to certain video coding/compression standards. Examples of those standards are noted above and described further herein.

[38] Fig. 3 may be a functional block diagram of a video decoder 300 according to an embodiment of the present invention.

[39] A receiver 302 may receive one or more codec video sequences to be decoded by the decoder 300; in the same or another embodiment, one coded video sequence at a time, where the decoding of each coded video sequence is independent from other coded video sequences. The coded video sequence may be received from a channel 301, which may be a hardware/software link to a storage device which stores the encoded video data. The receiver 302 may receive the encoded video data with other data, for example, coded audio data and/or ancillary data streams,

that may be forwarded to their respective using entities (not depicted). The receiver 302 may separate the coded video sequence from the other data. To combat network jitter, a buffer memory 303 may be coupled in between receiver 302 and entropy decoder / parser 304 (“parser” henceforth). When receiver 302 is receiving data from a store/forward device of sufficient bandwidth and controllability, or from an isosynchronous network, the buffer 303 may not be needed, or can be small. For use on best effort packet networks such as the Internet, the buffer 303 may be required, can be comparatively large and can advantageously of adaptive size.

[40] The video decoder 300 may include a parser 304 to reconstruct symbols 313 from the entropy coded video sequence. Categories of those symbols include information used to manage operation of the decoder 300, and potentially information to control a rendering device such as a display 312 that is not an integral part of the decoder but can be coupled to it. The control information for the rendering device(s) may be in the form of Supplementary Enhancement Information (SEI messages) or Video Usability Information parameter set fragments (not depicted). The parser 304 may parse / entropy-decode the coded video sequence received. The coding of the coded video sequence can be in accordance with a video coding technology or standard, and can follow principles well known to a person skilled in the art, including variable length coding, Huffman coding, arithmetic coding with or without context sensitivity, and so forth. The parser 304 may extract from the coded video sequence, a set of subgroup parameters for at least one of the subgroups of pixels in the video decoder, based upon at least one parameters corresponding to the group. Subgroups can include Groups of Pictures (GOPs), pictures, tiles, slices, macroblocks, Coding Units (CUs), blocks, Transform Units (TUs), Prediction Units (PUs) and so forth. The entropy decoder / parser may also extract from the

coded video sequence information such as transform coefficients, quantizer parameter values, motion vectors, and so forth.

[41] The parser 304 may perform entropy decoding / parsing operation on the video sequence received from the buffer 303, so to create symbols 313. The parser 304 may receive encoded data, and selectively decode particular symbols 313. Further, the parser 304 may determine whether the particular symbols 313 are to be provided to a Motion Compensation Prediction unit 306, a scaler / inverse transform unit 305, an Intra Prediction Unit 307, or a loop filter 311.

[42] Reconstruction of the symbols 313 can involve multiple different units depending on the type of the coded video picture or parts thereof (such as: inter and intra picture, inter and intra block), and other factors. Which units are involved, and how, can be controlled by the subgroup control information that was parsed from the coded video sequence by the parser 304. The flow of such subgroup control information between the parser 304 and the multiple units below is not depicted for clarity.

[43] Beyond the functional blocks already mentioned, decoder 300 can be conceptually subdivided into a number of functional units as described below. In a practical implementation operating under commercial constraints, many of these units interact closely with each other and can, at least partly, be integrated into each other. However, for the purpose of describing the disclosed subject matter, the conceptual subdivision into the functional units below is appropriate.

[44] A first unit is the scaler / inverse transform unit 305. The scaler / inverse transform unit 305 receives quantized transform coefficient as well as control information, including which transform to use, block size, quantization factor, quantization scaling matrices, etc. as symbol(s)

313 from the parser 304. It can output blocks comprising sample values, that can be input into aggregator 310.

[45] In some cases, the output samples of the scaler / inverse transform 305 can pertain to an intra coded block; that is: a block that is not using predictive information from previously reconstructed pictures, but can use predictive information from previously reconstructed parts of the current picture. Such predictive information can be provided by an intra picture prediction unit 307. In some cases, the intra picture prediction unit 307 generates a block of the same size and shape of the block under reconstruction, using surrounding already reconstructed information fetched from the current (partly reconstructed) picture 309. The aggregator 310, in some cases, adds, on a per sample basis, the prediction information the intra prediction unit 307 has generated to the output sample information as provided by the scaler / inverse transform unit 305.

[46] In other cases, the output samples of the scaler / inverse transform unit 305 can pertain to an inter coded, and potentially motion compensated block. In such a case, a Motion Compensation Prediction unit 306 can access reference picture memory 308 to fetch samples used for prediction. After motion compensating the fetched samples in accordance with the symbols 313 pertaining to the block, these samples can be added by the aggregator 310 to the output of the scaler / inverse transform unit (in this case called the residual samples or residual signal) so to generate output sample information. The addresses within the reference picture memory from where the motion compensation unit fetches prediction samples can be controlled by motion vectors, available to the motion compensation unit in the form of symbols 313 that can have, for example X, Y, and reference picture components. Motion compensation also can

include interpolation of sample values as fetched from the reference picture memory when sub-sample exact motion vectors are in use, motion vector prediction mechanisms, and so forth.

[47] The output samples of the aggregator 310 can be subject to various loop filtering techniques in the loop filter unit 311. Video compression technologies can include in-loop filter technologies that are controlled by parameters included in the coded video bitstream and made available to the loop filter unit 311 as symbols 313 from the parser 304, but can also be responsive to meta-information obtained during the decoding of previous (in decoding order) parts of the coded picture or coded video sequence, as well as responsive to previously reconstructed and loop-filtered sample values.

[48] The output of the loop filter unit 311 can be a sample stream that can be output to the render device 312 as well as stored in the reference picture memory 557 for use in future inter-picture prediction.

[49] Certain coded pictures, once fully reconstructed, can be used as reference pictures for future prediction. Once a coded picture is fully reconstructed and the coded picture has been identified as a reference picture (by, for example, parser 304), the current reference picture 309 can become part of the reference picture buffer 308, and a fresh current picture memory can be reallocated before commencing the reconstruction of the following coded picture.

[50] The video decoder 300 may perform decoding operations according to a predetermined video compression technology that may be documented in a standard, such as ITU-T Rec. H.265. The coded video sequence may conform to a syntax specified by the video compression technology or standard being used, in the sense that it adheres to the syntax of the video compression technology or standard, as specified in the video compression technology document or standard and specifically in the profiles document therein. Also necessary for compliance can

be that the complexity of the coded video sequence is within bounds as defined by the level of the video compression technology or standard. In some cases, levels restrict the maximum picture size, maximum frame rate, maximum reconstruction sample rate (measured in, for example megasamples per second), maximum reference picture size, and so on. Limits set by levels can, in some cases, be further restricted through Hypothetical Reference Decoder (HRD) specifications and metadata for HRD buffer management signaled in the coded video sequence.

[51] In an embodiment, the receiver 302 may receive additional (redundant) data with the encoded video. The additional data may be included as part of the coded video sequence(s). The additional data may be used by the video decoder 300 to properly decode the data and/or to more accurately reconstruct the original video data. Additional data can be in the form of, for example, temporal, spatial, or signal-to-noise ratio (SNR) enhancement layers, redundant slices, redundant pictures, forward error correction codes, and so on.

[52] Fig. 4 may be a functional block diagram of a video encoder 400 according to an embodiment of the present disclosure.

[53] The encoder 400 may receive video samples from a video source 401 (that is not part of the encoder) that may capture video image(s) to be coded by the encoder 400.

[54] The video source 401 may provide the source video sequence to be coded by the encoder (303) in the form of a digital video sample stream that can be of any suitable bit depth (for example: 8 bit, 10 bit, 12 bit, ...), any colorspace (for example, BT.601 Y CrCb, RGB, ...) and any suitable sampling structure (for example Y CrCb 4:2:0, Y CrCb 4:4:4). In a media serving system, the video source 401 may be a storage device storing previously prepared video. In a videoconferencing system, the video source 401 may be a camera that captures local image information as a video sequence. Video data may be provided as a plurality of individual

pictures that impart motion when viewed in sequence. The pictures themselves may be organized as a spatial array of pixels, wherein each pixel can comprise one or more samples depending on the sampling structure, color space, etc. in use. A person skilled in the art can readily understand the relationship between pixels and samples. The description below focuses on samples.

[55] According to an embodiment, the encoder 400 may code and compress the pictures of the source video sequence into a coded video sequence 410 in real time or under any other time constraints as required by the application. Enforcing appropriate coding speed is one function of Controller 402. Controller controls other functional units as described below and is functionally coupled to these units. The coupling is not depicted for clarity. Parameters set by controller can include rate control related parameters (picture skip, quantizer, lambda value of rate-distortion optimization techniques, ...), picture size, group of pictures (GOP) layout, maximum motion vector search range, and so forth. A person skilled in the art can readily identify other functions of controller 402 as they may pertain to video encoder 400 optimized for a certain system design.

[56] Some video encoders operate in what a person skilled in the art readily recognizes as a “coding loop.” As an oversimplified description, a coding loop can consist of the encoding part of an encoder 402 (“source coder” henceforth) (responsible for creating symbols based on an input picture to be coded, and a reference picture(s)), and a (local) decoder 406 embedded in the encoder 400 that reconstructs the symbols to create the sample data that a (remote) decoder also would create (as any compression between symbols and coded video bitstream is lossless in the video compression technologies considered in the disclosed subject matter). That reconstructed sample stream is input to the reference picture memory 405. As the decoding of a symbol stream leads to bit-exact results independent of decoder location (local or remote), the reference picture

buffer content is also bit exact between local encoder and remote encoder. In other words, the prediction part of an encoder “sees” as reference picture samples exactly the same sample values as a decoder would “see” when using prediction during decoding. This fundamental principle of reference picture synchronicity (and resulting drift, if synchronicity cannot be maintained, for example because of channel errors) is well known to a person skilled in the art.

[57] The operation of the “local” decoder 406 can be the same as of a “remote” decoder 300, which has already been described in detail above in conjunction with Fig. 3. Briefly referring also to Fig. 4, however, as symbols are available and encoding and/or decoding of symbols to a coded video sequence by entropy coder 408 and parser 304 can be lossless, the entropy decoding parts of decoder 300, including channel 301, receiver 302, buffer 303, and parser 304 may not be fully implemented in local decoder 406.

[58] An observation that can be made at this point is that any decoder technology except the parsing/entropy decoding that is present in a decoder also necessarily needs to be present, in substantially identical functional form, in a corresponding encoder. The description of encoder technologies can be abbreviated as they are the inverse of the comprehensively described decoder technologies. Only in certain areas a more detail description is required and provided below.

[59] As part of its operation, the source coder 403 may perform motion compensated predictive coding, which codes an input frame predictively with reference to one or more previously-coded frames from the video sequence that were designated as “reference frames.” In this manner, the coding engine 407 codes differences between pixel blocks of an input frame and pixel blocks of reference frame(s) that may be selected as prediction reference(s) to the input frame.

[60] The local video decoder 406 may decode coded video data of frames that may be designated as reference frames, based on symbols created by the source coder 403. Operations of the coding engine 407 may advantageously be lossy processes. When the coded video data may be decoded at a video decoder (not shown in Fig. 4), the reconstructed video sequence typically may be a replica of the source video sequence with some errors. The local video decoder 406 replicates decoding processes that may be performed by the video decoder on reference frames and may cause reconstructed reference frames to be stored in the reference picture cache 405. In this manner, the encoder 400 may store copies of reconstructed reference frames locally that have common content as the reconstructed reference frames that will be obtained by a far-end video decoder (absent transmission errors).

[61] The predictor 404 may perform prediction searches for the coding engine 407. That is, for a new frame to be coded, the predictor 404 may search the reference picture memory 405 for sample data (as candidate reference pixel blocks) or certain metadata such as reference picture motion vectors, block shapes, and so on, that may serve as an appropriate prediction reference for the new pictures. The predictor 404 may operate on a sample block-by-pixel block basis to find appropriate prediction references. In some cases, as determined by search results obtained by the predictor 404, an input picture may have prediction references drawn from multiple reference pictures stored in the reference picture memory 405.

[62] The controller 402 may manage coding operations of the video coder 403, including, for example, setting of parameters and subgroup parameters used for encoding the video data.

[63] Output of all aforementioned functional units may be subjected to entropy coding in the entropy coder 408. The entropy coder translates the symbols as generated by the various functional units into a coded video sequence, by loss-less compressing the symbols according to

technologies known to a person skilled in the art as, for example Huffman coding, variable length coding, arithmetic coding, and so forth.

[64] The transmitter 409 may buffer the coded video sequence(s) as created by the entropy coder 408 to prepare it for transmission via a communication channel 411, which may be a hardware/software link to a storage device which would store the encoded video data. The transmitter 409 may merge coded video data from the video coder 403 with other data to be transmitted, for example, coded audio data and/or ancillary data streams (sources not shown).

[65] The controller 402 may manage operation of the encoder 400. During coding, the controller 405 may assign to each coded picture a certain coded picture type, which may affect the coding techniques that may be applied to the respective picture. For example, pictures often may be assigned as one of the following frame types:

[66] An Intra Picture (I picture) may be one that may be coded and decoded without using any other frame in the sequence as a source of prediction. Some video codecs allow for different types of Intra pictures, including, for example Independent Decoder Refresh Pictures. A person skilled in the art is aware of those variants of I pictures and their respective applications and features.

[67] A Predictive picture (P picture) may be one that may be coded and decoded using intra prediction or inter prediction using at most one motion vector and reference index to predict the sample values of each block.

[68] A Bi-directionally Predictive Picture (B Picture) may be one that may be coded and decoded using intra prediction or inter prediction using at most two motion vectors and reference indices to predict the sample values of each block. Similarly, multiple-predictive pictures can

use more than two reference pictures and associated metadata for the reconstruction of a single block.

[69] Source pictures commonly may be subdivided spatially into a plurality of sample blocks (for example, blocks of 4 x 4, 8 x 8, 4 x 8, or 16 x 16 samples each) and coded on a block-by-block basis. Blocks may be coded predictively with reference to other (already coded) blocks as determined by the coding assignment applied to the blocks' respective pictures. For example, blocks of I pictures may be coded non-predictively or they may be coded predictively with reference to already coded blocks of the same picture (spatial prediction or intra prediction). Pixel blocks of P pictures may be coded non-predictively, via spatial prediction or via temporal prediction with reference to one previously coded reference pictures. Blocks of B pictures may be coded non-predictively, via spatial prediction or via temporal prediction with reference to one or two previously coded reference pictures.

[70] The video coder 400 may perform coding operations according to a predetermined video coding technology or standard, such as ITU-T Rec. H.265. In its operation, the video coder 400 may perform various compression operations, including predictive coding operations that exploit temporal and spatial redundancies in the input video sequence. The coded video data, therefore, may conform to a syntax specified by the video coding technology or standard being used.

[71] In an embodiment, the transmitter 409 may transmit additional data with the encoded video. The source coder 403 may include such data as part of the coded video sequence. Additional data may comprise temporal/spatial/SNR enhancement layers, other forms of redundant data such as redundant pictures and slices, Supplementary Enhancement Information (SEI) messages, Visual Usability Information (VUI) parameter set fragments, and so on.

[72] Fig. 5 illustrates intra prediction modes used in HEVC and JEM. 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 in JEM on top of HEVC are depicted as dotted arrows in Figure 1 (b), 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. As shown in Fig. 5, the directional intra prediction modes as identified by dotted arrows, which is associated with an odd intra prediction mode index, are called odd intra prediction modes. The directional intra prediction modes as identified by solid arrows, which are associated with an even intra prediction mode index, are called even intra prediction modes. In this document, the directional intra prediction modes, as indicated by solid or dotted arrows in Fig. 5 are also referred as angular modes.

[73] In JEM, a total of 67 intra prediction modes are used for luma intra prediction. To code an intra mode, a most probable mode (MPM) list of size 6 is built based on the intra modes of the neighboring blocks. If intra mode is not from the MPM list, a flag is signaled to indicate whether intra mode belongs to the selected modes. In JEM-3.0, there are 16 selected modes, which are chosen uniformly as every fourth angular mode. In JVET-D0114 and JVET-G0060, 16 secondary MPMs are derived to replace the uniformly selected modes.

[74] Fig. 6 illustrates N reference tiers exploited for intra directional modes. There is a block unit 611, a segment A 601, a segment B 602, a segment C 603, a segment D 604, a segment E 605, a segment F 606, a first reference tier 610, a second reference tier 609, a third reference tier 608 and a fourth reference tier 607.

[75] In both HEVC and JEM, as well as some other standards such as H.264/AVC, the reference samples used for predicting the current block are restricted to a nearest reference line

(row or column). In the method of multiple reference line intra prediction, the number of candidate reference lines (row or columns) are increased from one (i.e. the nearest) to N for the intra directional modes, where N is an integer greater than or equal to one. Fig. 2 takes 4x4 prediction unit (PU) as an example to show the concept of the multiple line intra directional prediction method. An intra-directional mode could arbitrarily choose one of N reference tiers to generate the predictors. In other words, the predictor $p(x,y)$ is generated from one of the reference samples $S_1, S_2, \dots$, and S_N . A flag is signaled to indicate which reference tier is chosen for an intra-directional mode. If N is set as 1, the intra directional prediction method is the same as the traditional method in JEM 2.0. In Fig. 6, the reference lines 610, 609, 608 and 607 are composed of six segments 601, 602, 603, 604, 605 and 606 together with the top-left reference sample. In this document, a reference tier is also called a reference line. The coordinate of the top-left pixel within current block unit is (0,0) and the top left pixel in the 1st reference line is (-1,-1).

[76] In JEM, for the luma component, the neighboring samples used for intra prediction sample generations are filtered before the generation process. The filtering is controlled by the given intra prediction mode and transform block size. If the intra prediction mode is DC or the transform block size is equal to 4x4, neighboring samples are not filtered. If the distance between the given intra prediction mode and vertical mode (or horizontal mode) is larger than predefined threshold, the filtering process is enabled. For neighboring sample filtering, [1, 2, 1] filter and bi-linear filters are used.

[77] A position dependent intra prediction combination (PDPC) method is an intra prediction method which invokes a combination of the un-filtered boundary reference samples and HEVC

style intra prediction with filtered boundary reference samples. Each prediction sample

$pred[x][y]$ located at (x, y) is calculated as follows:

$$pred[x][y] = (wL * R_{-1,y} + wT * R_{x,-1} + wTL * R_{-1,-1} + (64 - wL - wT - wTL) * pred[x][y] + 32) \gg 6 \quad (\text{Eq. 1})$$

where $R_{x,-1}, R_{-1,y}$ represent the unfiltered reference samples located at top and left of current sample (x, y) , respectively, and $R_{-1,-1}$ represents the unfiltered reference sample located at the top-left corner of the current block. The weightings are calculated as below,

$$wT = 32 \gg ((y \ll 1) \gg shift) \quad (\text{Eq. 2})$$

$$wL = 32 \gg ((x \ll 1) \gg shift) \quad (\text{Eq. 3})$$

$$wTL = -(wL \gg 4) - (wT \gg 4) \quad (\text{Eq. 4})$$

$$shift = (\log_2(width) + \log_2(height) + 2) \gg 2 \quad (\text{Eq. 5}).$$

[78] Fig. 7 illustrates a diagram 700 in which DC mode PDPC weights (wL , wT , wTL) for $(0, 0)$ and $(1, 0)$ positions inside one 4×4 block. If PDPC is applied to DC, planar, horizontal, and vertical intra modes, additional boundary filters are not needed, such as the HEVC DC mode boundary filter or horizontal/vertical mode edge filters. Fig. 7 illustrates the definition of reference samples $R_{x,-1}$, $R_{-1,y}$ and $R_{-1,-1}$ for PDPC applied to the top-right diagonal mode. The prediction sample $pred(x', y')$ is located at (x', y') within the prediction block. The coordinate x of the reference sample $R_{x,-1}$ is given by: $x = x' + y' + 1$, and the coordinate y of the reference sample $R_{-1,y}$ is similarly given by: $y = x' + y' + 1$.

[79] Fig. 8 illustrates a Local Illumination Compensation (LIC) diagram 800 and is based on a linear model for illumination changes, using a scaling factor a and an offset b . And it is enabled or disabled adaptively for each inter-mode coded coding unit (CU).

[80] When LIC applies for a CU, a least square error method is employed to derive the parameters a and b by using the neighboring samples of the current CU and their corresponding

reference samples. More specifically, as illustrated in Figure 8, the subsampled (2:1 subsampling) neighboring samples of the CU and the corresponding samples (identified by motion information of the current CU or sub-CU) in the reference picture are used. The IC parameters are derived and applied for each prediction direction separately.

[81] When a CU is coded with merge mode, the LIC flag is copied from neighboring blocks, in a way similar to motion information copy in merge mode; otherwise, an LIC flag is signaled for the CU to indicate whether LIC applies or not.

[82] Fig. 9A illustrates intra prediction modes 900 used in HEVC. In HEVC, there are total 35 intra prediction modes, among which mode 10 is horizontal mode, mode 26 is vertical mode, and mode 2, mode 18 and mode 34 are diagonal modes. The intra prediction modes are signaled by three most probable modes (MPMs) and 32 remaining modes.

[83] Fig. 9B illustrates, in embodiments of VVC, there are total 87 intra prediction modes where mode 18 is horizontal mode, mode 50 is vertical mode, and mode 2, mode 34 and mode 66 are diagonal modes. Modes -1 ~ -10 and Modes 67 ~ 76 are called Wide-Angle Intra Prediction (WAIP) modes.

[84] The prediction sample $\text{pred}(x,y)$ located at position (x, y) is predicted using an intra prediction mode (DC, planar, angular) and a linear combination of reference samples according to the PDPC expression:

$$\text{pred}(x,y) = (w_L \times R_{-1,y} + w_T \times R_{x,-1} - w_{TL} \times R_{-1,-1} + (64 - w_L - w_T + w_{TL}) \times \text{pred}(x,y) + 32) \gg 6 \quad (6)$$

where $R_{x,-1}$, $R_{-1,y}$ represent the reference samples located at the top and left of current sample (x, y) , respectively, and $R_{-1,-1}$ represents the reference sample located at the top-left corner of the current block.

[85] For the DC mode the weights are calculated as follows for a block with dimensions width and height:

$$wT = 32 \gg ((y < 1) \gg nScale), wL = 32 \gg ((x < 1) \gg nScale), wTL = (wL \gg 4) + (wT \gg 4), \quad (7)$$

with $nScale = (\log_2(\text{width}) - 2 + \log_2(\text{height}) - 2 + 2) \gg 2$, where wT denotes the weighting factor for the reference sample located in the above reference line with the same horizontal coordinate, wL denotes the weighting factor for the reference sample located in the left reference line with the same vertical coordinate, and wTL denotes the weighting factor for the top-left reference sample of the current block, $nScale$ specifies how fast weighting factors decrease along the axis (wL decreasing from left to right or wT decreasing from top to bottom), namely weighting factor decrement rate, and it is the same along x-axis (from left to right) and y-axis (from top to bottom) in current design. And 32 denotes the initial weighting factors for the neighboring samples, and the initial weighting factor is also the top (left or top-left) weightings assigned to top-left sample in current CB, and the weighting factors of neighboring samples in PDPC process should be equal to or less than this initial weighting factor.

[86] For planar mode $wTL = 0$, while for horizontal mode $wTL = wT$ and for vertical mode $wTL = wL$. The PDPC weights can be calculated with adds and shifts only. The value of $\text{pred}(x,y)$ can be computed in a single step using Eq. 1.

[87] Herein the proposed methods may be used separately or combined in any order. Further, each of the methods (or embodiments), encoder, and decoder may be implemented by processing circuitry (e.g., one or more processors or one or more integrated circuits). In one example, the one or more processors execute a program that is stored in a non-transitory computer-readable

medium. In the following, the term block may be interpreted as a prediction block, a coding block, or a coding unit, i.e. CU.

[88] Fig. 10 illustrates exemplary embodiments of a flowchart 1000 such that at S100, data may be received such that at S101 it may be determined whether to implement processing for a unit such as a coding unit and or a transform unit. If so, at S102, then it may be determined if the coding unit is including an intra mode prediction. If so, then at S103, it may be determined if the coding unit is including a flag indicating whether SPS ACT is enabled, and if there is such indication in such flag, at S104, then it may also be determined if a tree type for the coding unit is a single tree type or not. According to exemplary embodiments, an `sps_act_enabled_flag` equal to 1 specifies that adaptive color transform may be used and the `cu_act_enabled_flag` may be present in the coding unit syntax.; an `sps_act_enabled_flag` equal to 0 may specify that adaptive color transform may not be used and `cu_act_enabled_flag` may not be present in the coding unit syntax; and when `sps_act_enabled_flag` is not present, may be inferred to be equal to 0.

[89] If it is determined at S102, S103, and S104 that the intra mode, the flag indicating enablement of the SPS ACT, and that a tree is a single tree type, then at S105, the processing may set a flag indicating that ACT is enabled in that coding unit at S105. According to exemplary embodiments, a `cu_act_enabled_flag` equal to 1 may specify that the residuals of the current coding unit are coded in YCgCo color space; a `cu_act_enabled_flag` equal to 0 may specify that the residuals of the current coding unit are coded in original color space; and when a `cu_act_enabled_flag` is not present, it may be inferred to be equal to 0. As such, based on such syntax, an inter block could be encoded with ACT mode if the `cu_coded_flag` is 1 which may be

interpreted as meaning that ACT mode could be enabled for inter block if there is more than one coefficient in the current CU.

[90] Then at S106 further processing discussed herein may be implemented as well as looping to S101 described above. Alternatively at S102, if it is determined that the mode is not set for intra, and/or at S103 if it is determined that an `sps_act_enabled` flag does not include such enabled indication, then at S107, it may be determined if there is an indication regarding a PLT prediction flag, and if not, then at S108 a determination of a value of a `general_merge_flag`. If such values at S102, S107, and S108 are set as discussed below, then the processing may set, at S109, a `cu_coded_flag`, whereafter at S110, or from S107 and S108, if there is currently set such `cu_coded_flag`, at S111 it may be determined if the coding unit is including a flag indicating whether SPS ACT is enabled, and if there is such indication in such flag, at S112, then it may be determined if an intra mode is now indicated, and if not, the processing may proceed at S104 as noted above.

[91] Fig. 11 illustrates exemplary embodiments of a flowchart 1100 such that at such that at S100, data may be received such that at S101 it may be determined whether to implement processing for a coding unit and or a transform unit. For example, regardless of at S201 or at S202 if there is a current CU in a prediction mode Intra (S201) or not (S202), the processing may proceed at S203 to determine when a TU coded flag of chrominance channels are both zero. If so, at S203, then by the illustrated Y', then such determination at S203 may be sufficient to proceed to S204 in inferring that a TU coded flag of luminance should be inferred to 1, and therefore, such inference may be made regardless of whether the current prediction mode of a current CU block is `MODE_INTRA` or not. Further, it may also be, in addition to such positive determination at S203, that the TU coded flag of chrominance channels are both zero, whether

also the current CU is coded with an ACT mode, and if so, then the proceed may then proceed to S204 rather than straightaway after S203. Nonetheless, at S204 thereafter and with negative determinations at S203, the processing may proceed as described at any of S100 and S106 as described above. According to embodiments, for a coding block with ACT on, a `tu_y_coded_flag` may not be signaled in the bitstream and should be inferred to 1 when the TU coded flag of chrominance channels are both zero. Further, a `cu_act_enabled_flag` signal may be signaled according to such exemplary embodiments without the checking of the prediction mode of current CU block (S201 and or S202) such that in embodiments for example, for a coding unit, only two conditions, `sps_act_enabled_flag` and `treeType` is `SINGLE_TREE`, for the signaling of the `cu_act_enabled_flag` may be checked and therefore the advantage of avoiding a conditionally signaling of the `cu_act_enabled_flag` twice based on the prediction mode may be achieved as described below and also shown in Fig. 12 for example.

[92] For embodiments with a `tu_y_coded_flag` of a current CU block that may intentionally not be signaled when the TU coded flags of both chroma channels are 0 and an ACT flag is 1, see Table 1, where, among other things, a ...`CuPredMode[ chType ][ x0 ][ y0 ]` == `MODE_INTRA` “&& !`cu_act_enabled_flag`” ... is included with signaling in TU level for ACT:

Table 1

<code>transform_unit(x0, y0, tbWidth, tbHeight, treeType, subTuIndex, chType) {</code>	Descriptor
...	

<pre> if((treeType == SINGLE_TREE treeType == DUAL_TREE_CHROMA) && ChromaArrayType != 0 && (IntraSubPartitionsSplitType == ISP_NO_SPLIT && ((subTuIndex == 0 && cu_sbt_pos_flag) (subTuIndex == 1 && !cu_sbt_pos_flag))) (IntraSubPartitionsSplitType != ISP_NO_SPLIT && (subTuIndex == NumIntraSubPartitions - 1))) { </pre>	
tu_cb_coded_flag [xC][yC]	ae(v)
tu_cr_coded_flag [xC][yC]	ae(v)
}	
if(treeType == SINGLE_TREE treeType == DUAL_TREE_LUMA) {	
<pre> if((IntraSubPartitionsSplitType == ISP_NO_SPLIT && !(cu_sbt_flag && ((subTuIndex == 0 && cu_sbt_pos_flag) (subTuIndex == 1 && !cu_sbt_pos_flag))) && ((CuPredMode[chType][x0][y0] == MODE_INTRA && !cu_act_enabled_flag) (chromaAvailable && (tu_cb_coded_flag[xC][yC] tu_cr_coded_flag[xC][yC]))) CbWidth[chType][x0][y0] > MaxTbSizeY CbHeight[chType][x0][y0] > MaxTbSizeY)) (IntraSubPartitionsSplitType != ISP_NO_SPLIT && (subTuIndex < NumIntraSubPartitions - 1 !InferTuCbfLuma))) </pre>	
tu_y_coded_flag [x0][y0]	ae(v)
if(IntraSubPartitionsSplitType != ISP_NO_SPLIT)	
InferTuCbfLuma = InferTuCbfLuma && !tu_y_coded_flag[x0][y0]	
}	

<pre> if((CbWidth[chType][x0][y0] > 64 CbHeight[chType][x0][y0] > 64 tu_y_coded_flag[x0][y0] (chromaAvailable && (tu_cb_coded_flag[xC][yC] tu_cr_coded_flag[xC][yC])) && treeType != DUAL_TREE_CHROMA && pps_cu_qp_delta_enabled_flag && !IsCuQpDeltaCoded) { </pre>	
cu_qp_delta_abs	ae(v)
if(cu_qp_delta_abs)	
cu_qp_delta_sign_flag	ae(v)
}	

[93] According to embodiments with a condition of cu_act_enabled_flag signaling for the current CU block including sps_act_enabled_flag and tree_type only according to exemplary embodiments as shown by Table 2 with respect to cu_act_enabled_flag signaling in CU level for ACT:

Table 2

coding_unit(x0, y0, cbWidth, cbHeight, cqtDepth, treeType, modeType) {	Descriptor
chType = treeType == DUAL_TREE_CHROMA ? 1 : 0	
...	
if(sps_act_enabled_flag && treeType == SINGLE_TREE)	
cu_act_enabled_flag	ae(v)
...	
if(CuPredMode[chType][x0][y0] != MODE_INTRA && !pred_mode_plt_flag && general_merge_flag[x0][y0] == 0)	
cu_coded_flag cu_act_enabled_flag	ae(v)
if(cu_coded_flag) {	
...	
LfnstDcOnly = 1	ae(v)
...	

}	
---	--

[94] For example, Fig. 12 illustrates exemplary embodiments of a flowchart 1200 such that at S100, data may be received such that at S301 it may be determined whether to implement processing for a coding unit and or a transform unit. If so, at S302, it may be determined if the coding unit is including a flag indicating whether SPS ACT is enabled, and if there is such indication in such flag, at S304, then it may also be determined if a tree type for the coding unit is a single tree type or not. According to such exemplary embodiments, an `sps_act_enabled_flag` equal to 1 specifies that adaptive color transform may be used and the `cu_act_enabled_flag` may be present in the coding unit syntax.; an `sps_act_enabled_flag` equal to 0 specifies that adaptive color transform may not be used and `cu_act_enabled_flag` may not be present in the coding unit syntax; and when `sps_act_enabled_flag` is not present, may be inferred to be equal to 0.

[95] If it is determined at S302 and S304 that the flag indicates enablement of the SPS ACT, and that a tree is a single tree type, then at S305, the processing may set a flag indicating that ACT is enabled in that coding unit at S305. According to exemplary embodiments, a `cu_act_enabled_flag` equal to 1 may specify that the residuals of the current coding unit are coded in YCgCo color space; a `cu_act_enabled_flag` equal to 0 may specify that the residuals of the current coding unit are coded in original color space; and when a `cu_act_enabled_flag` is not present, it may be inferred to be equal to 0. As such, based on such syntax, an inter block could be encoded with ACT mode if the `cu_coded_flag` is 1 which may be interpreted as meaning that ACT mode could be enabled for inter block if there is more than one coefficient in the current CU.

[96] Then at S106 further processing discussed herein may be implemented as well as looping to S301 described above. Alternatively at S302, if it is determined that an `sps_act_enabled` flag

does not include such enabled indication, then at S307, it may be determined if there is an indication regarding a PLT prediction flag, and if not, then at S808 a determination of a value of a general_merge_flag. If such values at S302, S307, and S308 are set accordingly as shown in Fig. 8, then the processing may set, at S309, a cu_coded_flag described above.

[97] Accordingly, such embodiments solve various technical problems when for example, if the coded CU block doesn't have any coefficient, the ACT mode should not be signaled anymore whereby accordingly the CU with the ACT mode should have one or more than one coefficient in the coded CU block, and for the inter block with ACT mode, the cu_coded_flag should be 1 to represent that the CU has at least one coefficient in transform unit thereby solving absent constraints for the intra CU with ACT mode.

[98] Such features represent advantageous coding tools such as for RGB videos. For example, see the illustration 1300 in Fig. 13 in which there is illustrated an encoding flow 1301 and decoding flow 1302 wherein there is illustrated in-loop ACT adopted into screen coding models (SCM) (such as a software test model of screen content coding extension of HEVC), where ACT is illustrated as operated in a residue domain, and a CU-level flag may be signaled to indicate a usage of the color-space transform. Such color transform used in SCM may be, according to exemplary embodiments, as follows:

$$\text{Forward transform: } \begin{bmatrix} Y \\ C_g \\ C_o \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 1 & 2 & 1 \\ -1 & 2 & -1 \\ 2 & 0 & -2 \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (8)$$

$$\text{Backward transform: } \begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & -1 & 1 \\ 1 & 1 & 0 \\ 1 & -1 & -1 \end{bmatrix} \times \begin{bmatrix} Y \\ C_g \\ C_o \end{bmatrix} \quad (9)$$

[99] Further, in the illustration 1400 in Fig. 14 there is illustrated a decoding process according to exemplary embodiments with the ACT such that, in view of the above-described embodiments and flowcharts, included the ACT tool in HEVC into the VVC framework to

enhance the efficiency of video coding whereby decoding with the ACT may be so applied. As in Fig. 14, it is shown in the illustration 1400 that a color space conversion may be carried out in a residual domain, and specifically, an additional decoding module, namely an inverse ACT, may be introduced, such as after an inverse transform so as to convert residuals from a YCgCo domain back to an original domain. Accordingly, viewing the above described Figs. 10-12 among the other disclosures, advantages are achieved over features in the VVC, when the maximum transform size is not smaller than the width or height of one coding unit (CU), one CU leaf node may also used as the unit of transform processing, and therefore, in embodiments herein, an ACT flag may be signaled for one CU to select the color space for coding its residuals, and following such HEVC ACT design, for inter and intra-block copy (IBC) CUs, the ACT may be only enabled when there is at least one non-zero coefficient in the CU, and for intra CUs, the ACT may only be enabled when chroma components select the same intra prediction mode of luma component, i.e., a DM mode, thereby advantageously at least avoiding such unnecessary or otherwise redundant signaling according to exemplary embodiments.

[100] According to exemplary embodiments, core transforms used for the color space conversions may be with respect to the following forward and inverse YCgCo color transform matrices, as described as follows, as applied. For example:

$$\begin{aligned}
 \text{forward transform : } \begin{bmatrix} C'_0 \\ C'_1 \\ C'_2 \end{bmatrix} &= \begin{bmatrix} 2 & 1 & 1 \\ 2 & -1 & -1 \\ 0 & -2 & 2 \end{bmatrix} \begin{bmatrix} C_0 \\ C_1 \\ C_2 \end{bmatrix} / 4 \\
 \text{inversed transform : } \begin{bmatrix} C_0 \\ C_1 \\ C_2 \end{bmatrix} &= \begin{bmatrix} 1 & 1 & 0 \\ 1 & -1 & -1 \\ 1 & -1 & 1 \end{bmatrix} \begin{bmatrix} C'_0 \\ C'_1 \\ C'_2 \end{bmatrix}
 \end{aligned} \tag{10}$$

[101] Additionally, to compensate the dynamic range change of residuals signals before and after color transform, the QP adjustments, such as of (-5, -5, -3) may be, applied to the transform

residuals. On the other hand, as shown in (1), forward and inverse color transforms may need to access the residuals of all three components. Correspondingly, in embodiments of the present application, the technical improvement of allowing for the ACT to be disabled in the following scenarios where not all residuals of three components are available. For example, viewing Figs. 10-12 and descriptions, there is a separate-tree partition case such that when a separate-tree is applied, luma and chroma samples inside one CTU are partitioned by different structures, which may result in that the CUs in the luma-tree only contains luma component and the CUs in the chroma-tree only contains two chroma components and also there is an intra sub-partition prediction (ISP) case in which the ISP sub-partition may only be applied to luma while chroma signals are coded without splitting, and in such ISP design, except the last ISP sub-partitions, the other sub-partitions only contain luma component according to embodiments.

[102] Accordingly, there may be such CU level signaling of ACT where CU level ACT related signaling may be included according to coding syntax tables, such also as the above tables and/or Table 3:

Table 3

coding_unit(x0, y0, cbWidth, cbHeight, cqtDepth, treeType, modeType) {	Descriptor
chType = treeType == DUAL_TREE_CHROMA ? 1 : 0	
...	
if(CuPredMode[chType][x0][y0] == MODE_INTRA && sps_act_enabled_flag && treeType == SINGLE_TREE)	
cu_act_enabled_flag	ae(v)
...	
if(CuPredMode[chType][x0][y0] != MODE_INTRA && !pred_mode_plt_flag && general_merge_flag[x0][y0] == 0)	
cu_coded_flag	ae(v)
if(cu_coded_flag) {	

...	
if(sps_act_enabled_flag && CuPredMode[chType][x0][y0] != MODE_INTRA && treeType == SINGLE_TREE)	
cu_act_enabled_flag	ac(v)
LfnstDcOnly = 1	ac(v)
...	
}	

[103] In embodiments, an `sps_act_enabled_flag` equal to 1 may specify that an adaptive color transform may be used and the `cu_act_enabled_flag` may be present in the coding unit syntax, and `sps_act_enabled_flag` equal to 0 may specify that an adaptive color transform may not be used and that a `cu_act_enabled_flag` may not be present in the coding unit syntax, and when an `sps_act_enabled_flag` is not present, it may be inferred to be equal to 0. In embodiments, a `cu_act_enabled_flag` equal to 1 may specify that the residuals of the current coding unit are coded in YCgCo color space, a `cu_act_enabled_flag` equal to 0 may specify that the residuals of the current coding unit are coded in original color space, and when a `cu_act_enabled_flag` is not present, it may be inferred to be equal to 0. According to exemplary embodiments, based on the above syntax, an inter block may be encoded with ACT mode if the `cu_coded_flag` is 1 and thereby an ACT mode may be enabled for an inter block in a case such as if there is more than one coefficient in a current CU.

[104] Further, syntax with respect to a TU level luma coded flag signaling for an ACT block, such as a TU coded flag for three color channels, may be included according to a following transform unit syntax table, Table 4:

Table 4

transform_unit(x0, y0, tbWidth, tbHeight, treeType, subTuIndex, chType) {	Descriptor
...	
<pre> if((treeType == SINGLE_TREE treeType == DUAL_TREE_CHROMA) && ChromaArrayType != 0 && (IntraSubPartitionsSplitType == ISP_NO_SPLIT && ((subTuIndex == 0 && cu_sbt_pos_flag) (subTuIndex == 1 && !cu_sbt_pos_flag))) (IntraSubPartitionsSplitType != ISP_NO_SPLIT && (subTuIndex == NumIntraSubPartitions - 1))) { </pre>	
tu_cb_coded_flag [xC][yC]	ae(v)
tu_cr_coded_flag [xC][yC]	ae(v)
}	
if(treeType == SINGLE_TREE treeType == DUAL_TREE_LUMA) {	
<pre> if((IntraSubPartitionsSplitType == ISP_NO_SPLIT && !(cu_sbt_flag && ((subTuIndex == 0 && cu_sbt_pos_flag) (subTuIndex == 1 && !cu_sbt_pos_flag))) && (CuPredMode[chType][x0][y0] == MODE_INTRA (chromaAvailable && (tu_cb_coded_flag[xC][yC] tu_cr_coded_flag[xC][yC]))) CbWidth[chType][x0][y0] > MaxTbSizeY CbHeight[chType][x0][y0] > MaxTbSizeY)) (IntraSubPartitionsSplitType != ISP_NO_SPLIT && (subTuIndex < NumIntraSubPartitions - 1 !InferTuCbfLuma))) </pre>	

tu_y_coded_flag[x0][y0]	ae(v)
if(IntraSubPartitionsSplitType != ISP_NO_SPLIT)	
InferTuCbfLuma = InferTuCbfLuma && !tu_y_coded_flag[x0][y0]	
}	

[105] According to exemplary embodiments, the TU coded flag of a luma.component related semantics may also be shown as follows: a tu_y_coded_flag[x0][y0] equal to 1 may specify that the luma transform block contains one or more transform coefficient levels not equal to 0, and the array indices x0, y0 may specify the location (x0, y0) of the top-left luma sample of the considered transform block relative to the top-left luma sample of the picture, and when tu_y_coded_flag[x0][y0] is not present, its value may be inferred as follows: if cu_sbt_flag is equal to 1 and one of the following (a), (b) conditions is true, tu_y_coded_flag[x0][y0] is inferred to be equal to 0 (a) subTuIndex is equal to 0 and cu_sbt_pos_flag may be equal to 1, (b) subTuIndex may be equal to 1 and cu_sbt_pos_flag may be equal to 0, otherwise, if treeType is equal to DUAL_TREE_CHROMA, tu_y_coded_flag[x0][y0] may be inferred to be equal to 0, and further otherwise, tu_y_coded_flag[x0][y0] may be inferred to be equal to 1. In such syntax and related semantics, there may not be a condition check about an ACT block for a TU coded flag.

[106] Further, for an exemplary tu_y_coded_flag of an ACT block, the TU coded flag for a luminance component may be described as follows:

```

if(!isChroma(partitioner.chType))
{
    if(!CU::isIntra(cu) && trDepth == 0 && !chromaCbfs.sigChroma(area.chromaFormat))

```

```

{
    TU::setCbfAtDepth(tu, COMPONENT_Y, trDepth, 1);
}
else if (cu.sbtInfo && tu.noResidual)
{
    TU::setCbfAtDepth(tu, COMPONENT_Y, trDepth, 0);
}
else if (cu.sbtInfo && !chromaCbfs.sigChroma(area.chromaFormat))
{
    assert(!tu.noResidual);
    TU::setCbfAtDepth(tu, COMPONENT_Y, trDepth, 1);
}
else
{
    bool lumaCbfsInferredACT = (cu.colorTransform && cu.predMode == MODE_INTRA
&&
                                trDepth == 0 && !chromaCbfs.sigChroma(area.chromaFormat));
    bool lastCbfsInferred    = lumaCbfsInferredACT; // ISP and ACT are mutually exclusive
    bool previousCbf         = false;
    bool rootCbfSoFar        = false;
    if (cu.ispMode)
    {
        ...
    }
}

```

```

    }

    bool cbfY = lastCbfsInferred ? true :

        cbf_comp(cs, tu.Y(), trDepth, previousCbf, cu.ispMode);

    TU::setCbfAtDepth(tu, COMPONENT_Y, trDepth, (cbfY ? 1 : 0));

    }

    }

```

[107] Accordingly, the TU coded flag of luminance may be inferred to 1 when the TU coded flag of chrominance channels are both zero and the current CU is an intra block and is coded with ACT mode.

[108] As described herein, there may be one or more hardware processor and computer components, such as buffers, arithmetic logic units, memory instructions, configured to determine or store predetermined delta values (differences) between ones of the values described herein according to exemplary embodiments.

[109] Accordingly, by exemplary embodiments described herein, the technical problems noted above may be advantageously improved upon by one or more of these technical solutions. That is, according to embodiments, to address one or more different technical problems, this disclosure describes novel technical aspects in which an access unit delimiter (AUD) may be advantageously signaled to indicate which slice_type values are present in the slices of the coded pictures in the access unit containing the access unit delimiter NAL unit. The pic_type may be useful to identify whether the AU is independent or dependent from outer AU. Further, it is asserted that such novel syntax element signaling is advantageous in indications of random access AU and robustness of AU boundary detection respectively according to exemplary embodiments and therefore advantageous for improved accuracy and efficiency for example.

[110] The techniques described above, can be implemented as computer software using computer-readable instructions and physically stored in one or more computer-readable media or by a specifically configured one or more hardware processors. For example, FIG. 12 shows a computer system 1200 suitable for implementing certain embodiments of the disclosed subject matter.

[111] The computer software can be coded using any suitable machine code or computer language, that may be subject to assembly, compilation, linking, or like mechanisms to create code comprising instructions that can be executed directly, or through interpretation, micro-code execution, and the like, by computer central processing units (CPUs), Graphics Processing Units (GPUs), and the like.

[112] The instructions can be executed on various types of computers or components thereof, including, for example, personal computers, tablet computers, servers, smartphones, gaming devices, internet of things devices, and the like.

[113] The components shown in FIG. 15 for computer system 1500 are exemplary in nature and are not intended to suggest any limitation as to the scope of use or functionality of the computer software implementing embodiments of the present disclosure. Neither should the configuration of components be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the exemplary embodiment of a computer system 1500.

[114] Computer system 1500 may include certain human interface input devices. Such a human interface input device may be responsive to input by one or more human users through, for example, tactile input (such as: keystrokes, swipes, data glove movements), audio input (such as: voice, clapping), visual input (such as: gestures), olfactory input (not depicted). The human

interface devices can also be used to capture certain media not necessarily directly related to conscious input by a human, such as audio (such as: speech, music, ambient sound), images (such as: scanned images, photographic images obtain from a still image camera), video (such as two-dimensional video, three-dimensional video including stereoscopic video).

[115] Input human interface devices may include one or more of (only one of each depicted): keyboard 1501, mouse 1502, trackpad 1503, touch screen 1510, joystick 1505, microphone 1506, scanner 1508, camera 1507.

[116] Computer system 1500 may also include certain human interface output devices. Such human interface output devices may be stimulating the senses of one or more human users through, for example, tactile output, sound, light, and smell/taste. Such human interface output devices may include tactile output devices (for example tactile feedback by the touch-screen 1510, or joystick 1505, but there can also be tactile feedback devices that do not serve as input devices), audio output devices (such as: speakers 1509, headphones (not depicted)), visual output devices (such as screens 1510 to include CRT screens, LCD screens, plasma screens, OLED screens, each with or without touch-screen input capability, each with or without tactile feedback capability—some of which may be capable to output two dimensional visual output or more than three dimensional output through means such as stereographic output; virtual-reality glasses (not depicted), holographic displays and smoke tanks (not depicted)), and printers (not depicted).

[117] Computer system 1500 can also include human accessible storage devices and their associated media such as optical media including CD/DVD ROM/RW 1520 with CD/DVD 1511 or the like media, thumb-drive 1522, removable hard drive or solid state drive 1523, legacy magnetic media such as tape and floppy disc (not depicted), specialized ROM/ASIC/PLD based devices such as security dongles (not depicted), and the like.

[118] Those skilled in the art should also understand that term “computer readable media” as used in connection with the presently disclosed subject matter does not encompass transmission media, carrier waves, or other transitory signals.

[119] Computer system 1500 can also include interface 1599 to one or more communication networks 1598. Networks 1598 can for example be wireless, wireline, optical. Networks 1598 can further be local, wide-area, metropolitan, vehicular and industrial, real-time, delay-tolerant, and so on. Examples of networks 1598 include local area networks such as Ethernet, wireless LANs, cellular networks to include GSM, 3G, 4G, 5G, LTE and the like, TV wireline or wireless wide area digital networks to include cable TV, satellite TV, and terrestrial broadcast TV, vehicular and industrial to include CANBus, and so forth. Certain networks 1598 commonly require external network interface adapters that attached to certain general-purpose data ports or peripheral buses (1550 and 1551) (such as, for example USB ports of the computer system 1500; others are commonly integrated into the core of the computer system 1500 by attachment to a system bus as described below (for example Ethernet interface into a PC computer system or cellular network interface into a smartphone computer system). Using any of these networks 1598, computer system 1500 can communicate with other entities. Such communication can be uni-directional, receive only (for example, broadcast TV), uni-directional send-only (for example CANbus to certain CANbus devices), or bi-directional, for example to other computer systems using local or wide area digital networks. Certain protocols and protocol stacks can be used on each of those networks and network interfaces as described above.

[120] Aforementioned human interface devices, human-accessible storage devices, and network interfaces can be attached to a core 1540 of the computer system 1500.

[121] The core 1540 can include one or more Central Processing Units (CPU) 1541, Graphics Processing Units (GPU) 1542, a graphics adapter 1517, specialized programmable processing units in the form of Field Programmable Gate Areas (FPGA) 1543, hardware accelerators for certain tasks 1544, and so forth. These devices, along with Read-only memory (ROM) 1545, Random-access memory 1546, internal mass storage such as internal non-user accessible hard drives, SSDs, and the like 1547, may be connected through a system bus 1548. In some computer systems, the system bus 1548 can be accessible in the form of one or more physical plugs to enable extensions by additional CPUs, GPU, and the like. The peripheral devices can be attached either directly to the core's system bus 1548, or through a peripheral bus 1551. Architectures for a peripheral bus include PCI, USB, and the like.

[122] CPUs 1541, GPUs 1542, FPGAs 1543, and accelerators 1544 can execute certain instructions that, in combination, can make up the aforementioned computer code. That computer code can be stored in ROM 1545 or RAM 1546. Transitional data can be also be stored in RAM 1546, whereas permanent data can be stored for example, in the internal mass storage 1547. Fast storage and retrieval to any of the memory devices can be enabled through the use of cache memory, that can be closely associated with one or more CPU 1541, GPU 1542, mass storage 1547, ROM 1545, RAM 1546, and the like.

[123] The computer readable media can have computer code thereon for performing various computer-implemented operations. The media and computer code can be those specially designed and constructed for the purposes of the present disclosure, or they can be of the kind well known and available to those having skill in the computer software arts.

[124] As an example and not by way of limitation, the computer system having architecture 1500, and specifically the core 1540 can provide functionality as a result of processor(s)

(including CPUs, GPUs, FPGA, accelerators, and the like) executing software embodied in one or more tangible, computer-readable media. Such computer-readable media can be media associated with user-accessible mass storage as introduced above, as well as certain storage of the core 1540 that are of non-transitory nature, such as core-internal mass storage 1547 or ROM 1545. The software implementing various embodiments of the present disclosure can be stored in such devices and executed by core 1540. A computer-readable medium can include one or more memory devices or chips, according to particular needs. The software can cause the core 1540 and specifically the processors therein (including CPU, GPU, FPGA, and the like) to execute particular processes or particular parts of particular processes described herein, including defining data structures stored in RAM 1546 and modifying such data structures according to the processes defined by the software. In addition or as an alternative, the computer system can provide functionality as a result of logic hardwired or otherwise embodied in a circuit (for example: accelerator 1544), which can operate in place of or together with software to execute particular processes or particular parts of particular processes described herein. Reference to software can encompass logic, and vice versa, where appropriate. Reference to a computer-readable media can encompass a circuit (such as an integrated circuit (IC)) storing software for execution, a circuit embodying logic for execution, or both, where appropriate. The present disclosure encompasses any suitable combination of hardware and software.

[125] While this disclosure has described several exemplary embodiments, there are alterations, permutations, and various substitute equivalents, which fall within the scope of the disclosure. It will thus be appreciated that those skilled in the art will be able to devise numerous systems and methods which, although not explicitly shown or described herein, embody the principles of the disclosure and are thus within the spirit and scope thereof.

WHAT IS CLAIMED IS:

1. A method for video coding performed by at least one processor, the method comprising:
 - obtaining video data;
 - obtaining a coding unit (CU) block of the video data;
 - determining whether a flag of the CU block is set to a predetermined flag condition;
 - determining whether a tree type of the CU block is set to a predetermined tree type;
 - determining whether to signal an adaptive color transform (ACT) flag based on any of whether the flag of the CU block is set to the predetermined flag condition and whether the tree type of the CU block is set to the predetermined tree type; and
 - coding the video data based on a whether the ACT flag is signaled.
2. The method according to claim 1, wherein determining whether to signal the ACT flag is based on only whether the flag of the CU block is set to the predetermined flag condition.
3. The method according to claim 1, wherein determining whether to signal the ACT flag is based on both of whether the flag of the CU block is set to the predetermined flag condition and whether the tree type of the CU block is set to the predetermined tree type.
4. The method according to claim 1, wherein the predetermined tree type indicates a single tree type, and wherein the predetermined flag condition comprises `sps_act_enabled_flag` being equal to 1.

5. The method according to claim 1, wherein determining whether to signal an adaptive color transform (ACT) flag is implemented regardless of whether a prediction mode of the CU is an intra mode.

6. The method according to claim 1, further comprises determining whether transform unit (TU) coded flags are both zero and whether the CU is coded with an ACT mode.

7. The method according to claim 6, wherein the TU coded flags are flags of chrominance channels.

8. The method according to claim 7, further comprising:
determining a TU coded flag of luminance is to be inferred to be 1 based on determining that the TU coded flags are both zero and that the CU is coded with the ACT mode.

9. The method according to claim 8, wherein determining the TU coded flag of luminance is to be inferred to be 1 is implemented regardless of whether a prediction mode of the CU is an intra mode.

10. The method according to claim 9, wherein coding the video data is further based on determining whether the TU coded flag of luminance is to be inferred to be 1.

11. An apparatus for video coding performed by at least one processor, the apparatus comprising:

at least one memory configured to store computer program code;

at least one processor configured to access the computer program code and operate as instructed by the computer program code, the computer program code including:

- first obtaining code configured to cause the at least one processor to obtain video data;
- second obtaining code configured to cause the at least one processor to obtain a coding unit (CU) block of the video data;
- first determining code configured to cause the at least one processor to determine whether a flag of the CU block is set to a predetermined flag condition;
- second determining code configured to cause the at least one processor to determine whether a tree type of the CU block is set to a predetermined tree type;
- third determining code configured to cause the at least one processor to determine whether to signal an adaptive color transform (ACT) flag based on any of whether the flag of the CU block is set to the predetermined flag condition and whether the tree type of the CU block is set to the predetermined tree type; and
- coding code configured to cause the at least one processor to code the video data based on a whether the ACT flag is signaled.

12. The apparatus according to claim 11, wherein determining whether to signal the ACT flag is based on only whether the flag of the CU block is set to the predetermined flag condition.

13. The apparatus according to claim 11, wherein determining whether to signal the ACT flag is based on both of whether the flag of the CU block is set to the predetermined flag condition and whether the tree type of the CU block is set to the predetermined tree type.

14. The apparatus according to claim 11, wherein the predetermined tree type indicates a single tree type, and wherein the predetermined flag condition comprises `sps_act_enabled_flag` being equal to 1.

15. The apparatus according to claim 11, wherein determining whether to signal an adaptive color transform (ACT) flag is implemented regardless of whether a prediction mode of the CU is an intra mode.

16. The apparatus according to claim 11, further comprising fourth determining code configured to cause the at least one processor to determine whether transform unit (TU) coded flags are both zero and whether the CU is coded with an ACT mode.

17. The apparatus according to claim 16, wherein the TU coded flags are flags of chrominance channels.

18. The apparatus according to claim 17, further comprising:
fifth determining code configured to cause the at least one processor to determine whether a TU coded flag of luminance is to be inferred to be 1 based on determining that the TU coded flags are both zero and that the CU is coded with the ACT mode.

19. The apparatus according to claim 18, wherein determining the TU coded flag of luminance is to be inferred to be 1 is implemented regardless of whether a prediction mode of the CU is an intra mode.

20. A non-transitory computer readable medium storing a program causing a computer to execute a process, the process comprising:

- obtaining video data;
- obtaining a coding unit (CU) block;
- determining whether a flag of the CU block is set to a predetermined flag condition;
- determining whether a tree type of the CU block is set to a predetermined tree type;
- determining whether to signal an adaptive color transform (ACT) flag based on any of whether the flag of the CU block is set to the predetermined flag condition and whether the tree type of the CU block is set to the predetermined tree type; and
- coding the video data based on a whether the ACT flag is signaled.