



(51) International Patent Classification:

H04N 19/593 (2014.01) *H04N 19/11* (2014.01)
H04N 19/105 (2014.01) *H04N 19/136* (2014.01)
H04N 19/176 (2014.01)

(21) International Application Number:

PCT/US2017/068605

(22) International Filing Date:

27 December 2017 (27.12.2017)

(25) Filing Language:

English

(26) Publication Language:

English

(30) Priority Data:

62/439,724	28 December 2016 (28.12.2016)	US
62/440,379	29 December 2016 (29.12.2016)	US
62/459,797	16 February 2017 (16.02.2017)	US
62/482,178	05 April 2017 (05.04.2017)	US
62/522,420	20 June 2017 (20.06.2017)	US
15/855,687	27 December 2017 (27.12.2017)	US

(71) Applicant: ARRIS ENTERPRISES LLC [US/US]; 3871 Lakefield Drive, Suwanee, GA 30024 (US).

(72) Inventors: PANUSOPONE, Krit; 10725 Calle Mar De Mariposa, San Diego, CA 92130 (US). HONG, Seung-wook; 10165 Lone Dove Street, San Diego, CA 92127 (US). WANG, Limin; 13641 Shoal Summit Drive, San Diego, CA 92128 (US).

(74) Agent: SWANSON, Lori Anne, D. et al.; 3871 Lakefield Drive, Suwanee, GA 30024 (US).

(81) Designated States (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(54) Title: IMPROVED VIDEO BITSTREAM CODING

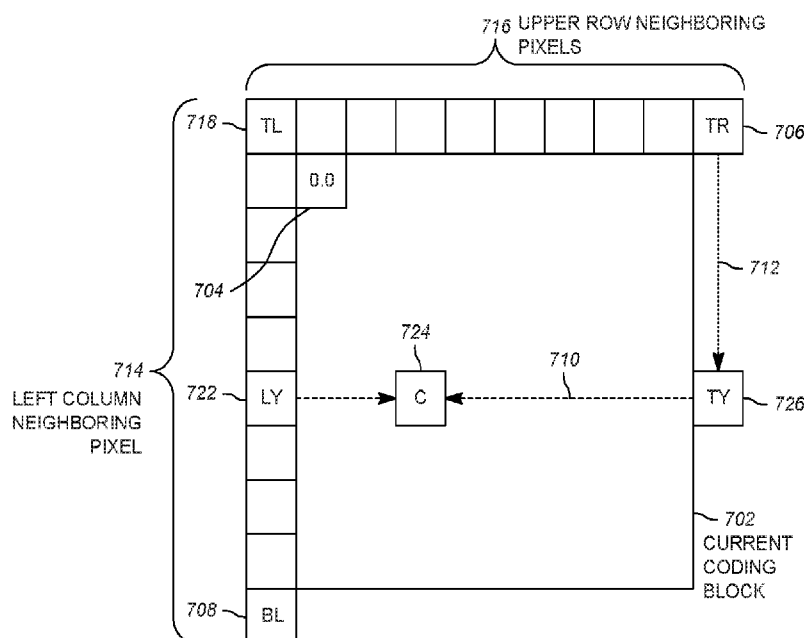


FIG. 7A

(57) Abstract: A method of decoding JVET video includes receiving a bitstream indicating how a coding tree unit was partitioned into coding units, and parsing said bitstream to generate at least one predictor based on an intra prediction mode signaled in the bitstream. The predictor may be generated by interpolating neighboring pixels for each pixel within a coding block. The computation may be more accurate by deriving a value for a bottom right neighboring pixel.



(84) Designated States (*unless otherwise indicated, for every kind of regional protection available*): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

- *with international search report (Art. 21(3))*
- *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

IMPROVED VIDEO BITSTREAM CODING

CLAIM OF PRIORITY

[0001] This Application claims priority under 35 U.S.C. §119(e) from earlier filed United States Provisional Application Serial No. 62/439724, filed December 28, 2016, from earlier filed United States Provisional Application Serial No. 62/440379, filed December 29, 2016, from earlier filed United States Provisional Application Serial No. 62/459,797, filed February 16, 2017, from earlier filed United States Provisional Application Serial No. 62/522420, filed June 20, 2017, and from earlier filed United States Provisional Application Serial No. 62/482178, filed April 5, 2017 each of which is hereby incorporated by reference.

BACKGROUND

[0002] The technical improvements in evolving video coding standards illustrate the trend of increasing coding efficiency to enable higher bit-rates, higher resolutions, and better video quality. The Joint Video Exploration Team is developing a new video coding scheme referred to as JVET. Similar to other video coding schemes like HEVC (High Efficiency Video Coding), JVET is a block-based hybrid spatial and temporal predictive coding scheme. However, relative to HEVC, JVET includes many modifications to bitstream structure, syntax, constraints, and mapping for the generation of decoded pictures. JVET has been implemented in Joint Exploration Model (JEM) encoders and decoders.

SUMMARY

[0003] The present disclosure provides a method of using planar mode to predict pixel values for a current coding unit (CU). The method includes calculating an intensity value of a bottom right neighboring pixel of the current CU. The bottom right intensity value can be used with an intensity value of a neighboring pixel from the row of horizontal boundary pixels on the upper side of the current CU to calculate intensity values for the neighboring pixels in a column along the right side of the current block. The bottom right intensity value can be used with a neighboring pixel from the column of vertical boundary pixels on the left side of the current CU to calculate intensity values for the neighboring pixels in a row along the lower side of the current block. The method further comprises calculating a first and second predictor and deriving a prediction pixel value from the first and second predictors, wherein a plurality of prediction pixel values make up a prediction block.

[0004] The present disclosure also provides an apparatus to predict pixel values for a current coding unit (CU) by way of the techniques disclosed herein.

BRIEF DESCRIPTION OF THE DRAWINGS

[0005] Further details of the present invention are explained with the help of the attached drawings in which:

[0006] FIG. 1 depicts division of a frame into a plurality of Coding Tree Units (CTUs).

[0007] FIG. 2 depicts an exemplary partitioning of a CTU into Coding Units (CUs).

[0008] FIG. 3 depicts a quadtree plus binary tree (QTBT) representation of FIG. 2's CU partitioning.

[0009] FIG. 4 depicts a simplified block diagram for CU coding in a JVET encoder.

- [0010] FIG. 5 depicts possible intra prediction modes for luma components in JVET.
- [0011] FIG. 6 depicts a simplified block diagram for CU coding in a JVET decoder.
- [0012] FIG. 7A depicts the horizontal predictor calculation.
- [0013] FIG. 7B depicts a vertical predictor calculation.
- [0014] Fig. 8 depicts an example of planar prediction plane with the JVET method that deviates from the flat plane.
- [0015] Fig. 9 illustrates the planar prediction block generated based on equation (5), with the corner intensity values identical to an example in Fig. 8.
- [0016] FIG. 10 depicts an embodiment of a computer system adapted and/or configured to process a method of CU coding.
- [0017] FIG. 11 is a flow diagram that illustrates a method for performing the disclosed techniques.
- [0018] FIG. 12 is a high-level view of a source device and destination device that may incorporate features of the systems and devices described herein.

DETAILED DESCRIPTION

- [0019] Digital video involves a large amount of data representing each and every frame of a digital video sequence, or series of frames, in an uncompressed manner. Transmitting uncompressed digital video across computer networks is usually limited by bandwidth limitations, and usually requires a large amount of storage space. Encoding the digital video may reduce both storage and bandwidth requirements.
- [0020] Frames of a video sequence, or more specifically the coding tree units within each frame, can be encoded and decoded using JVET. JVET is a video coding scheme being developed

by the Joint Video Exploration Team. Versions of JVET have been implemented in JEM (Joint Exploration Model) encoders and decoders. Similar to other video coding schemes like HEVC (High Efficiency Video Coding), JVET is a block-based hybrid spatial and temporal predictive coding scheme.

[0021] During coding with JVET, a frame is first divided into square blocks called Coding Tree Units (CTUs) 100, as shown in FIG. 1. FIG. 1 depicts division of a frame into a plurality of CTUs 100. For example, CTUs 100 can be blocks of 128x128 pixels. A frame can be an image in a video sequence, which may include a plurality of frames. A frame can include a matrix, or set of matrices, with pixel values representing intensity measures in the image. The pixel values can be defined to represent color and brightness in full color video coding, where pixels are divided into three channels. For example, in a YCbCr color space pixels can have a luma value, Y, that represents gray level intensity in the image, and two chrominance values, Cb and Cr, that represent the extent to which color differs from gray to blue and red. In other embodiments, pixel values can be represented with values in different color spaces or models. The resolution of the video can determine the number of pixels in a frame. A higher resolution can mean more pixels and a better definition of the image, but can also lead to higher bandwidth, storage, and transmission requirements.

[0022] FIG. 2 depicts an exemplary partitioning of a CTU 100 into CUs 102, which are the basic units of prediction in coding. Each CTU 100 in a frame can be partitioned into one or more CUs (Coding Units) 102. CUs 102 can be used for prediction and transform as described below. Unlike HEVC, in JVET the CUs 102 can be rectangular or square, and can be coded without further partitioning into prediction units or transform units. The CUs 102 can be as large as their root CTUs 100, or be smaller subdivisions of a root CTU 100 as small as 4x4 blocks.

[0023] In JVET, a CTU 100 can be partitioned into CUs 102 according to a quadtree plus binary tree (QTBT) scheme in which the CTU 100 can be recursively split into square blocks according to a quadtree, and those square blocks can then be recursively split horizontally or vertically according to binary trees. Parameters can be set to control splitting according to the QTBT, such as the CTU size, the minimum sizes for the quadtree and binary tree leaf nodes, the maximum size for the binary tree root node, and the maximum depth for the binary trees.

[0024] By way of a non-limiting example, FIG. 2 shows a CTU 100 partitioned into CUs 102, with solid lines indicating quadtree splitting and dashed lines indicating binary tree splitting. As illustrated, the binary splitting allows horizontal splitting and vertical splitting to define the structure of the CTU and its subdivision in to CUs.

[0025] FIG. 3 shows a QTBT block structure representation of FIG. 2's partitioning. To generate an encoded representation of a picture or image, the video encoder may generate a set of CTUs. in FIG. 3, quadtree root node represents the CTU 100. To generate a coded CTU, the video encoder may recursively perform quadtree partitioning on the coding tree blocks of a CTU to divide the coding tree blocks into coding blocks (CB)s. Thus, as used herein, a video slice (e.g., a video frame or a portion of a video frame) may be partitioned into coding blocks, which may also be referred to as coding units, and each such coding unit may include blocks, such as luma blocks and chroma blocks, that are independently decoded. Thus, because JEM supports flexibility for CU partition shapes to better match local characteristics of video data, CUs may have non-square shapes and coding may take place at the CU level or at the luma or chroma block level within the CU. Reference is made herein to encoding or decoding a coding block, where a coding block represents a block of pixels that makes up a CU or coding blocks within the CU.

[0026] As shown in FIG. 3, each child node in a CTU 100 in the quadtree portion represents one of four square blocks split from a parent square block. The square blocks represented by the quadtree leaf nodes can then be divided zero or more times using binary trees, with the quadtree leaf nodes being root nodes of the binary trees, representing the parent coding unit that is partitioned into two child coding units. At each level of the binary tree portion, a block can be divided vertically or horizontally and symmetrically or asymmetrically. For example, a flag set to “0” may indicate that the block is symmetrically split horizontally, while a flag set to “1” may indicate that the block is symmetrically split vertically.

[0027] After quadtree splitting and binary tree splitting, the blocks represented by the QTBT’s leaf nodes represent the final CUs 102 to be coded, such as coding using inter prediction or intra prediction. Inter-prediction exploits temporal redundancies between different frames. For example, temporally adjacent frames of video may include blocks of pixels that remain substantially the same. During encoding, a motion vector may interrelate movement of blocks of pixels in one frame to a block of correlating pixels in another frame. Thus, the system need not encode the block of pixels twice, but rather encode the block of pixels once and provide the motion vector to predict the correlated block of pixels.

[0028] For intra-prediction, a frame or portion of a frame may be encoded without reference to pixels in other frames. Instead, intra-prediction may exploit the spatial redundancies among blocks of pixels within a frame. For example, where spatially adjacent blocks of pixels have similar attributes, the coding process may reference a spatial correlation between adjacent blocks, exploiting the correlation by prediction of a target block based on prediction modes used in adjacent blocks.

[0029] For inter slices or slices or full frames coded with inter prediction, different partitioning structures can be used for luma and chroma components. For example, for an inter slice a CU 102 can have coding blocks for different color components, such as such as one luma CB and two chroma CBs. For intra slices or slices or full frames coded with intra prediction, the partitioning structure can be the same for luma and chroma components. Thus, each of the CTUs 100 may comprise a coding tree block of luma samples, corresponding coding tree blocks of chroma samples, and syntax structures used to code the samples of the coding tree blocks. A CTU may comprise a single coding tree block and the syntax structures used to code the samples. In either case, the CTU with coding blocks may be comprised within one or more coding units, as shown by FIG. 2 and FIG. 3.

[0030] FIG. 4 depicts a simplified block diagram for CU coding in a JVET encoder. The main stages of video coding include partitioning to identify CUs 102 as described above, followed by encoding CUs 102 using prediction at 404 or 406, generation of a residual CU 410 at 408, transformation at 412, quantization at 416, and entropy coding at 420. The encoder and encoding process illustrated in FIG. 4 also includes a decoding process that is described in more detail below.

[0031] Given a current CU 102 (e.g., a CU prior to encoding; e.g., the original CU to be encoded for transmission in the bitstream by generating a prediction CU) the encoder can obtain a prediction CU 402 either spatially using intra prediction at 404 or temporally using inter prediction at 406. The basic idea of prediction coding is to transmit a differential, or residual, signal between the original signal and a prediction for the original signal. At the receiver side, the original signal can be reconstructed by adding the residual and the prediction, as will be described below. Because the differential signal has a lower correlation than the original signal, fewer bits are needed for its transmission.

[0032] A sequence of coding units may make up a slice, and one or more slices may make up a picture. A slice may include one or more slice segments, each in its own NAL unit. A slice or slice segment may include header information for the slice or bitstream.

[0033] A slice, such as an entire picture or a portion of a picture, coded entirely with intra-predicted CUs can be an I slice that can be decoded without reference to other slices, and as such can be a possible point where decoding can begin. A slice coded with at least some inter-predicted CUs can be a predictive (P) or bi-predictive (B) slice that can be decoded based on one or more reference pictures. P slices may use intra-prediction and inter-prediction with previously coded slices. For example, P slices may be compressed further than the I-slices by the use of inter-prediction, but need the coding of a previously coded slice to code them. B slices can use data from previous and/or subsequent slices for its coding, using intra-prediction or inter-prediction using an interpolated prediction from two different frames, thus increasing the accuracy of the motion estimation process. In some cases P slices and B slices can also or alternately be encoded using intra block copy, in which data from other portions of the same slice is used.

[0034] As will be discussed below, the intra prediction or inter prediction can be performed based on reconstructed CUs 434 from previously coded CUs 102, such as neighboring CUs 102 or CUs 102 in reference pictures.

[0035] When a CU 102 is coded spatially with intra prediction at 404, an intra prediction mode can be found that best predicts pixel values of the CU 102 based on samples from neighboring CUs 102 in the picture.

[0036] When coding a CU's luma or chroma block components, the encoder can generate a list of candidate intra prediction modes. While HEVC had 35 possible intra prediction modes for luma components, in JVET there are 67 possible intra prediction modes for luma components.

These include a planar mode that uses a three-dimensional plane of values generated from neighboring pixels, a DC mode that uses values averaged from neighboring pixels, and the 65 directional modes shown in FIG. 5 that use values copied from neighboring pixels along the indicated directions.

[0037] When generating a list of candidates intra prediction modes for a CU's luma block component, the number of candidate modes on the list can depend on the CU's size. The candidate list can include: a subset of HEVC's 35 modes with the lowest SATD (Sum of Absolute Transform Difference) costs; new directional modes added for JVET that neighbor the candidates found from the HEVC modes; and modes from a set of six most probable modes (MPMs) for the CU 102 that are identified based on intra prediction modes used for previously coded neighboring blocks as well as a list of default modes.

[0038] When coding a CU's chroma block components, a list of candidate intra prediction modes can also be generated. The list of candidate modes can include modes generated with cross-component linear model projection from luma samples, intra prediction modes found for luma CBs in particular collocated positions in the chroma block, and chroma prediction modes previously found for neighboring blocks. The encoder can find the candidate modes on the lists with the lowest rate distortion costs, and use those intra prediction modes when coding the CU's luma and chroma components. Syntax can be coded in the bitstream that indicates the intra prediction modes used to code each CU 102.

[0039] After the best intra prediction modes for a CU 102 have been selected, the encoder can generate a prediction CU 402 using those modes. When the selected modes are directional modes, a 4-tap filter can be used to improve the directional accuracy. Columns or rows at the top or left

side of the prediction block can be adjusted with boundary prediction filters, such as 2-tap or 3-tap filters.

[0040] The prediction CU 402 can be smoothed further with a position dependent intra prediction combination (PDPC) process that adjusts a prediction CU 402 generated based on filtered samples of neighboring blocks using unfiltered samples of neighboring blocks, or adaptive reference sample smoothing using 3-tap or 5-tap low pass filters to process reference samples.

[0041] When a CU 102 is coded temporally with inter prediction at 406, a set of motion vectors (MVs) can be found that points to samples in reference pictures that best predict pixel values of the CU 102. Inter prediction exploits temporal redundancy between slices by representing a displacement of a block of pixels in a slice. The displacement is determined according to the value of pixels in previous or following slices through a process called motion compensation. Motion vectors and associated reference indices that indicate pixel displacement relative to a particular reference picture can be provided in the bitstream to a decoder, along with the residual between the original pixels and the motion compensated pixels. The decoder can use the residual and signaled motion vectors and reference indices to reconstruct a block of pixels in a reconstructed slice.

[0042] In JVET, motion vector accuracy can be stored at 1/16 pel, and the difference between a motion vector and a CU's predicted motion vector can be coded with either quarter-pel resolution or integer-pel resolution.

[0043] In JVET motion vectors can be found for multiple sub-CUs within a CU 102, using techniques such as advanced temporal motion vector prediction (ATMVP), spatial-temporal motion vector prediction (STMVP), affine motion compensation prediction, pattern matched motion vector derivation (PMMVD), and/or bi-directional optical flow (BIO).

[0044] Using ATMVP, the encoder can find a temporal vector for the CU 102 that points to a corresponding block in a reference picture. The temporal vector can be found based on motion vectors and reference pictures found for previously coded neighboring CUs 102. Using the reference block pointed to by a temporal vector for the entire CU 102, a motion vector can be found for each sub-CU within the CU 102.

[0045] STMVP can find motion vectors for sub-CUs by scaling and averaging motion vectors found for neighboring blocks previously coded with inter prediction, together with a temporal vector.

[0046] Affine motion compensation prediction can be used to predict a field of motion vectors for each sub-CU in a block, based on two control motion vectors found for the top corners of the block. For example, motion vectors for sub-CUs can be derived based on top corner motion vectors found for each 4x4 block within the CU 102.

[0047] PMMVD can find an initial motion vector for the current CU 102 using bilateral matching or template matching. Bilateral matching can look at the current CU 102 and reference blocks in two different reference pictures along a motion trajectory, while template matching can look at corresponding blocks in the current CU 102 and a reference picture identified by a template. The initial motion vector found for the CU 102 can then be refined individually for each sub-CU.

[0048] BIO can be used when inter prediction is performed with bi-prediction based on earlier and later reference pictures, and allows motion vectors to be found for sub-CUs based on the gradient of the difference between the two reference pictures.

[0049] In some situations, local illumination compensation (LIC) can be used at the CU level to find values for a scaling factor parameter and an offset parameter, based on samples neighboring

the current CU 102 and corresponding samples neighboring a reference block identified by a candidate motion vector. In JVET, the LIC parameters can change and be signaled at the CU level.

[0050] For some of the above methods the motion vectors found for each of a CU's sub-CUs can be signaled to decoders at the CU level. For other methods, such as PMMVD and BIO, motion information is not signaled in the bitstream to save overhead, and decoders can derive the motion vectors through the same processes.

[0051] After the motion vectors for a CU 102 have been found, the encoder can generate a prediction CU 402 using those motion vectors. In some cases, when motion vectors have been found for individual sub-CUs, Overlapped Block Motion Compensation (OBMC) can be used when generating a prediction CU 402 by combining those motion vectors with motion vectors previously found for one or more neighboring sub-CUs.

[0052] When bi-prediction is used, JVET can use decoder-side motion vector refinement (DMVR) to find motion vectors. DMVR allows a motion vector to be found based on two motion vectors found for bi-prediction using a bilateral template matching process. In DMVR, a weighted combination of prediction CUs 402 generated with each of the two motion vectors can be found, and the two motion vectors can be refined by replacing them with new motion vectors that best point to the combined prediction CU 402. The two refined motion vectors can be used to generate the final prediction CU 402.

[0053] At 408, once a prediction CU 402 has been found with intra prediction at 404 or inter prediction at 406 as described above, the encoder can subtract the prediction CU 402 from the current CU 102 find a residual CU 410.

[0054] The encoder can use one or more transform operations at 412 to convert the residual CU 410 into transform coefficients 414 that express the residual CU 410 in a transform domain,

such as using a discrete cosine block transform (DCT-transform) to convert data into the transform domain. JVET allows more types of transform operations than HEVC, including DCT-II, DST-VII, DST-VII, DCT-VIII, DST-I, and DCT-V operations. The allowed transform operations can be grouped into sub-sets, and an indication of which sub-sets and which specific operations in those sub-sets were used can be signaled by the encoder. In some cases, large block-size transforms can be used to zero out high frequency transform coefficients in CUs 102 larger than a certain size, such that only lower-frequency transform coefficients are maintained for those CUs 102.

[0055] In some cases, a mode dependent non-separable secondary transform (MDNSST) can be applied to low frequency transform coefficients 414 after a forward core transform. The MDNSST operation can use a Hypercube-Givens Transform (Hit) based on rotation data. When used, an index value identifying a particular MDNSST operation can be signaled by the encoder.

[0056] At 416, the encoder can quantize the transform coefficients 414 into quantized transform coefficients 416. The quantization of each coefficient may be computed by dividing a value of the coefficient by a quantization step, which is derived from a quantization parameter (QP). In some embodiments, the Qstep is defined as $2(QP-4)/6$. Because high precision transform coefficients 414 can be converted into quantized transform coefficients 416 with a finite number of possible values, quantization can assist with data compression. Thus, quantization of the transform coefficients may limit a number of bits generated and sent by the transformation process. However, while quantization is a lossy operation, and the loss by quantization cannot be recovered, the quantization process presents a trade-off between quality of the reconstructed sequence and an amount of information needed to represent the sequence. For example, a lower QP value can result in better quality decoded video, although a higher amount of data may be required for

representation and transmission. In contrast, a high QP value can result in lower quality reconstructed video sequences but with lower data and bandwidth needs.

[0057] JVET may utilize variance-based adaptive quantization techniques, which allows every CU 102 to use a different quantization parameter for its coding process (instead of using the same frame QP in the coding of every CU 102 of the frame). The variance-based adaptive quantization techniques adaptively lower the quantization parameter of certain blocks while increasing it in others. To select a specific QP for a CU 102, the CU's variance is computed. In brief, if a CU's variance is higher than the average variance of the frame, a higher QP than the frame's QP may be set for the CU 102. If the CU 102 presents a lower variance than the average variance of the frame, a lower QP may be assigned.

[0058] At 420, the encoder can find final compression bits 422 by entropy coding the quantized transform coefficients 418. Entropy coding aims to remove statistical redundancies of the information to be transmitted. In JVET, CABAC (Context Adaptive Binary Arithmetic Coding) can be used to code the quantized transform coefficients 418, which uses probability measures to remove the statistical redundancies. For CUs 102 with non-zero quantized transform coefficients 418, the quantized transform coefficients 418 can be converted into binary. Each bit ("bin") of the binary representation can then be encoded using a context model. A CU 102 can be broken up into three regions, each with its own set of context models to use for pixels within that region.

[0059] Multiple scan passes can be performed to encode the bins. During passes to encode the first three bins (bin0, bin1, and bin2), an index value that indicates which context model to use for the bin can be found by finding the sum of that bin position in up to five previously coded neighboring quantized transform coefficients 418 identified by a template.

[0060] A context model can be based on probabilities of a bin's value being '0' or '1'. As values are coded, the probabilities in the context model can be updated based on the actual number of '0' and '1' values encountered. While HEVC used fixed tables to re-initialize context models for each new picture, in JVET the probabilities of context models for new inter-predicted pictures can be initialized based on context models developed for previously coded inter-predicted pictures.

[0061] The encoder can produce a bitstream that contains entropy encoded bits 422 of residual CUs 410, prediction information such as selected intra prediction modes or motion vectors, indicators of how the CUs 102 were partitioned from a CTU 100 according to the QTBT structure, and/or other information about the encoded video. The bitstream can be decoded by a decoder as discussed below. In some embodiments, the encoder can save overhead in the bitstream by omitting information from the bitstream that indicates which intra prediction modes were used to encode CUs 102, and the decoder can use template matching when decoding CUs 102 encoded with intra prediction.

[0062] In addition to using the quantized transform coefficients 418 to find the final compression bits 422, the encoder can also use the quantized transform coefficients 418 to generate reconstructed CUs 434 by following the same decoding process that a decoder would use to generate reconstructed CUs 434. Thus, once the transformation coefficients have been computed and quantized by the encoder, the quantized transform coefficients 418 may be transmitted to the decoding loop in the encoder. After quantization of a CU's transform coefficients, a decoding loop allows the encoder to generate a reconstructed CU 434 identical to the one the decoder generates in the decoding process. Accordingly, the encoder can use the same reconstructed CUs 434 that a decoder would use for neighboring CUs 102 or reference pictures when performing intra prediction

or inter prediction for a new CU 102. Reconstructed CUs 102, reconstructed slices, or full reconstructed images or frames may serve as references for further prediction stages.

[0063] At the encoder's decoding loop (and see below, for the same operations in the decoder) to obtain pixel values for the reconstructed image, a dequantization process may be performed. To dequantize a frame, for example, a quantized value for each pixel of a frame is multiplied by the quantization step, e.g., (Qstep) described above, to obtain reconstructed dequantized transform coefficients 426. For example, in the decoding process shown in FIG. 4 in the encoder, the quantized transform coefficients 418 of a residual CU 410 can be dequantized at 424 to find dequantized transform coefficients 426. If an MDNSST operation was performed during encoding, that operation can be reversed after dequantization.

[0064] At 428, the dequantized transform coefficients 426 can be inverse transformed to find a reconstructed residual CU 430, such as by applying a DCT to the values to obtain the reconstructed image. At 432 the reconstructed residual CU 430 can be added to a corresponding prediction CU 402 found with intra prediction at 404 or inter prediction at 406, in order to find a reconstructed CU 434. While in some embodiments the encoder can perform intra prediction at 404 as described above, in other embodiments the encoder can perform intra prediction template matching to generate a prediction CU 402 in the same way that a decoder would use template matching for intra prediction if information identifying the intra prediction mode used for the CU 102 is omitted from the bitstream.

[0065] At 436, one or more filters can be applied to the reconstructed data during the decoding process (in the encoder or, as described below, in the decoder), at either a picture level or CU level. For example, the encoder can apply a deblocking filter, a sample adaptive offset (SAO) filter, and/or an adaptive loop filter (ALF). The encoder's decoding process may implement filters to

estimate and transmit to a decoder the optimal filter parameters that can address potential artifacts in the reconstructed image. Such improvements increase the objective and subjective quality of the reconstructed video. In deblocking filtering, pixels near a sub-CU boundary may be modified, whereas in SAO, pixels in a CTU 100 may be modified using either an edge offset or band offset classification. JVET's ALF can use filters with circularly symmetric shapes for each 2x2 block. An indication of the size and identity of the filter used for each 2x2 block can be signaled.

[0066] If reconstructed pictures are reference pictures, they can be stored in a reference buffer 438 for inter prediction of future CUs 102 at 406.

[0067] During the above steps, JVET allows a content adaptive clipping operations to be used to adjust color values to fit between lower and upper clipping bounds. The clipping bounds can change for each slice, and parameters identifying the bounds can be signaled in the bitstream.

[0068] FIG. 6 depicts a simplified block diagram for CU coding in a JVET decoder. A JVET decoder can receive a bitstream, or bits, 602 containing information about encoded video data. The encoded video data may represent partitioned luma blocks and partitioned chroma blocks, where the chroma blocks may be partitioned and coded independently of the luma blocks. The decoder may determine the respective coding mode respective to the blocks within each CU 102.

[0069] The bitstream can indicate how CUs 102 of a picture were partitioned from a CTU 100 according to a QTBT structure. The decoder may determine the tree structure as part of obtaining syntax elements from the bitstream. The tree structure may specify how the initial video block, such as a CTB, is partitioned into smaller video blocks, such as coding units.

[0070] As described herein, for each respective non-leaf node of the tree structure at each depth level of the tree structure, there are various splitting patterns for the respective non-leave node. By way of a non-limiting example, the bitstream can signal how CUs 102 were partitioned from each

CTU 100 in a QTBT using quadtree partitioning, symmetric binary partitioning, and/or asymmetric binary partitioning. The video block corresponding to such respective non-leaf node may be partitioned into video blocks corresponding to the child nodes of the respective non-leaf node.

[0071] The bitstream can also indicate prediction information for the CUs 102 such as intra prediction modes or motion vectors. Bits 602 represent entropy encoded residual CUs. In some embodiments, syntax can be coded in the bitstream that indicates the intra prediction modes used to code each CU 102. In some embodiments, the encoder can have omitted information in the bitstream about intra prediction modes used to encode some or all CUs 102 coded using intra prediction, and as such the decoder can use template matching for intra prediction

[0072] At 604 the decoder can decode the entropy encoded bits 602 using the CABAC context models signaled in the bitstream by the encoder. The decoder can use parameters signaled by the encoder to update the context models' probabilities in the same way they were updated during encoding.

[0073] After reversing the entropy encoding at 604 to find quantized transform coefficients 606, the decoder can dequantize them at 608 to find dequantized transform coefficients 610. If an MDNSST operation was performed during encoding, that operation can be reversed by the decoder after dequantization.

[0074] At 612, the dequantized transform coefficients 610 can be inverse transformed to find a reconstructed residual CU 614. At 616, the reconstructed residual CU 614 can be added to a corresponding prediction CU 626 found with intra prediction at 622 or inter prediction at 624, in order to find a reconstructed CU 618. In some embodiments, the decoder can find the prediction CU 626 using template matching for intra prediction.

[0075] At 620, one or more filters can be applied to the reconstructed data, at a picture level or CU level, for example. For example, the decoder can apply a deblocking filter, a sample adaptive offset (SAO) filter, and/or an adaptive loop filter (ALF). As described above, the in-loop filters located in the decoding loop of the encoder may be used to estimate optimal filter parameters to increase the objective and subjective quality of a frame. These parameters are transmitted to the decoder to filter the reconstructed frame at 620 to match the filtered reconstructed frame in the encoder.

[0076] After reconstructed pictures have been generated by finding reconstructed CUs 618 and applying signaled filters, the decoder can output the reconstructed pictures as output video 628. If reconstructed pictures are to be used as reference pictures, they can be stored in a reference buffer 630 for inter prediction of future CUs 102 at 624.

[0077] In some embodiments, the bitstream received by a JVET decoder can include syntax identifying which intra prediction mode was used to encode a CU 102 with intra prediction, such that the decoder can directly use the signaled intra prediction mode at 622 to generate a prediction CU 626. In some embodiments, such syntax can be omitted to save overhead by reducing the number of bits in the bitstream. In these embodiments, when the decoder is not provided with an indication of which intra prediction mode was used to encode a CU 102, the decoder can use template matching for intra prediction at 622 to derive the intra prediction mode it should use to generate a prediction CU 626. In some embodiments, an encoder can similarly use template matching for intra prediction at 404 when generating a prediction CU 402 to combine with a reconstructed residual CU 430 at 432 within its decoding loop.

[0078] As described herein, intra coding is a main tool for video compression. It utilizes the spatial neighbors of a pixel to create a predictor, from which a prediction residual between the

pixel and its predictor is determined. Video encoder then compresses the residuals, resulting in the coding bitstream. The developing video coding standard, JVET, allows 67 possible intra prediction modes, including planar mode, DC mode, and 65 angular direction modes, as shown in FIG. 5. Each intra prediction mode has unique prediction generation method, based on either left-side neighbor or top-side neighbor. Each intra coding unit (CU) selects at least one intra prediction mode to be used, which needs to be signaled as overhead in bitstream. For example, where a single CU includes one luma coding block and two chroma coding blocks, each of the luma coding block and chroma coding block can have their own intra prediction mode. Thus, if signaled in the bitstream, multiple intra prediction modes may be identified.

[0079] Disclosed herein are various techniques to implement an improved intra prediction coding technique using a refined planar prediction. In embodiments, the neighboring pixels used in intra prediction are calculated more accurately, resulting in a more efficient and more accurate prediction of pixels within the current coding block.

[0080] Planar mode is often the most frequently used intra coding mode in HEVC and JVET. Planar mode is often suited for blocks with a smooth image whose pixel values gradually change with a small planar gradient. FIGs. 7A and 7B show the HEVC and JVET planar predictor generation process for a coding unit (block) 702 with height $H = 8$ and width $W = 8$, where the (0,0) coordinates corresponds to the top-left (TL) position 704 within the coding CU, where TL position 718 is a top left neighboring corner pixel. TR 706 denotes top right position and BL 708 denotes bottom left position. The dashed line 710 indicates interpolation and the dotted line 712 indicates replication. FIG. 7A depicts the horizontal predictor calculation and FIG. 7B depicts a vertical predictor calculation.

[0081] Planar mode in HEVC and JVET (HEVC Planar) generates a first order approximation of the prediction for a current coding unit (CU) by forming a plane based on the intensity values of the neighboring pixels. Due to a raster-scan coding order, the reconstructed left column neighboring pixels 714 and the reconstructed top row neighboring pixels 716 are available for a current coding block 702, but not the right column neighboring pixels and the bottom row neighboring pixels. The planar predictor generation process sets the intensity values of all the right column neighboring pixels to be the same as the intensity value of the top right neighboring pixel 706, and the intensity values of all the bottom row pixels to be the same as the intensity value of the bottom left neighboring pixel 708.

[0082] Once the neighboring pixels surrounding a predicting block are defined, the horizontal and the vertical predictors ($P_h(x, y)$ and $P_v(x, y)$, respectively) for each pixel within the current coding block 702 are determined according to equations (1) and (2) below.

$$P_h(x, y) = (W - 1 - x) * R(-1, y) + (x + 1) * R(W, -1) \quad (1)$$

$$P_v(x, y) = (H - 1 - y) * R(x, -1) + (y + 1) * R(-1, H) \quad (2)$$

Where:

$R(x, y)$ denotes the intensity value of the reconstructed neighboring pixel at (x, y) coordinate,

W is block width, and

H is block height.

[0083] As depicted in FIG. 7A, a horizontal predictor is interpolated from referencing reconstructed reference sample L_y 722, a pixel in the left column neighboring pixels 714, that is at the same y-coordinate as the current sample C 724, and referencing sample T_y 726 which is a copy of previously reconstructed sample TR 718 that is adjacent to the top-right pixel of the coding

unit. Thus, the horizontal predictor is calculated using linear interpolation between a value of a respective horizontal boundary pixel and a value of a vertical boundary pixel.

[0084] As depicted in FIG. 7B, a vertical predictor is interpolated from reconstructed sample Tx 728 from the top row neighboring pixels at the same x-coordinate as the current prediction pixel C and referencing sample Lx 730 which is a copy of the previously reconstructed sample BL 708, a neighboring left pixel that is adjacent to the bottom left sample of the coding unit. Thus, the vertical predictor is calculated using linear interpolation between a value of a respective vertical boundary pixel and a value of a horizontal boundary value.

[0085] Once the horizontal and vertical predictor is determined for a coding unit pixel, the final predictor can be determined. For each pixel in a coding block, the final planar predictor, $P(x, y)$, is the intensity value that may be computed by averaging the horizontal and vertical predictors according to equation (3), with certain adjustments in embodiments in which the current CU or coding block is non-square.

$$P(x, y) = (H * P_h(x, y) + W * P_v(x, y)) / (2 * H * W) \quad (3)$$

[0086] The encoder may also signal a residual between the prediction coding unit and the current coding unit in the bitstream to the decoder.

[0087] Certain constraints with the current planar mode in JVET may cause coding inefficiencies. For example, as described above, all the right column neighboring pixels are set to the same intensity value, which is the value of the top right neighboring pixel. Similarly, all the bottom row neighboring pixels are set to the same intensity value, which is the value of the bottom left neighboring pixel. Setting all right column pixels the same, and all bottom row pixels the same, conflicts with plane prediction assumption. Thus, these constraints cause the planar prediction block to deviate from an ideal flat plane. FIG. 8 depicts an example of planar prediction plane with

the JVET method that deviates from the flat plane (TR denotes top right position and BL denotes bottom left position).

[0088] Disclosed herein are embodiments for deriving an intensity value of a bottom right neighboring pixel $P(W,H)$ for a current coding block. In embodiments, a coding unit may be the smallest coding block(s) within a CTU that is coded. In embodiments, a coding unit may include luma and chroma coding blocks that are independently coded. The proposed techniques include computing an intensity value of the bottom right neighboring pixel (lifting), and then computing the intensity values of the bottom row and the right column neighboring pixels, using the derived intensity value of the bottom right neighboring pixel along with the intensity values of other corner neighboring pixels, such as the top right neighboring pixel and the bottom left neighboring pixel. It is noted that a corner pixel at coordinates $x=0, y=0$ within the coding block is different from the corner neighboring pixels (e.g., $P(W,H)$) which is outside the coding block.

[0089] An example of a lifting process is a weighted average of the top right and the bottom left neighboring pixels, as defined in equation (4).

$$P(W, H) = (W * R(W, -1) + H * R(-1, H)) / (H + W) \quad (4)$$

[0090] Another example of lifting process is maintaining a flat plane based on the intensity value of the top left, the top right, and the bottom left neighboring pixels, as defined in equation (5).

$$P(W, H) = R(W, -1) + R(-1, H) - R(-1, -1) \quad (5)$$

[0091] Fig. 9 illustrates the planar prediction block generated based on equation (5), with the corner intensity values identical to an example in Fig. 8. As shown, a flat planar prediction plane results from using equation (5) given the same three corner points as in Fig. 8 (TR denotes top-right position, TL denotes top-left position and BL denotes bottom-left position).

[0092] With the derived intensity value of the bottom right neighboring pixel of $P(W, H)$, the intensity values of the bottom row neighboring pixels, $P_b(x, H)$, and the right column neighboring pixels, $P_r(W, y)$, can be computed accordingly. One example of such computation follows linear interpolation according to equations (6) and (7).

$$P_b(x, H) = \frac{(W-1-x)*R(-1, H) + (x+1)*P(W, H)}{W} \quad (6)$$

$$P_r(W, y) = \frac{(H-1-y)*R(W, -1) + (y+1)*P(W, H)}{H} \quad (7)$$

[0093] Once the neighboring pixels surrounding a current coding block are defined, the horizontal and the vertical predictors ($P_h(x, y)$ and $P_v(x, y)$, respectively) for each pixel within the coding block are determined according to equations (8) and (9) below.

$$P_h(x, y) = (W - 1 - x) * R(-1, y) + (x + 1) * P_r(W, y) \quad (8)$$

$$P_v(x, y) = (H - 1 - y) * R(x, -1) + (y + 1) * P_b(x, H) \quad (9)$$

[0094] Notice that in equations (8) and (9), $P_h(x, y)$ and $P_v(x, y)$ are shown as scaled up version of horizontal and vertical predictors. These factors will be compensated in the final predictor calculation step.

[0095] It may also be beneficial to use filtered version of neighboring pixels intensity, instead of actual neighboring pixel intensity for lifting part of the bottom right position adjustment. This is especially true when the coding content is noisy. An example of such filtering operation is described in equations (10) and (11).

$$R'(-1, H) = \frac{(R(-1, H-3) + 3*R(-1, H-2) + 5*R(-1, H-1) + 7*R(-1, H))}{16} \quad (10)$$

$$R'(W, -1) = \frac{(R(W-3, -1) + 3*R(W-2, -1) + 5*R(W-1, -1) + 7*R(W, -1))}{16} \quad (11)$$

[0096] Another example of filtering operation is described in equations (10a) and (11a).

$$R'(-1, H) = \frac{(R(-1, H-2) + 2*R(-1, H-1) + 5*R(-1, H))}{8} \quad (10a)$$

$$R'(W, -1) = \frac{(R(W-2, -1) + 2 * R(W-1, -1) + 5 * R(W, -1))}{8} \quad (11a)$$

[0097] The filtering operation may be selectively employed based on whether neighboring pixels have already been filtered by other prior process (such as mode dependent intra smoothing in HEVC). In embodiments, when neighboring pixels have not been filtered by prior task, the filtering operation is used. Otherwise, the filtering operation is not used.

[0098] Alternatively, it may be beneficial to set top-right and bottom-left corner positions to be $R(W - 1, -1)$ and $R(-1, H - 1)$, respectively. In this situation, interpolation for intermediate predictors can be described as an example by equations (12) - (16).

$$P(W - 1, H - 1) = (W * R(W - 1, -1) + H * R(-1, H - 1)) / (H + W) \quad (12)$$

$$P_b(x, H - 1) = \frac{(W-1-x) * R(-1, H-1) + (x+1) * P(W-1, H-1)}{W} \quad (13)$$

$$P_r(W - 1, y) = \frac{(H-1-y) * R(W-1, -1) + (y+1) * P(W-1, H-1)}{H} \quad (14)$$

$$P_h(x, y) = (W - 1 - x) * R(-1, y) + (x + 1) * P_r(W - 1, y) \quad (15)$$

$$P_v(x, y) = (H - 1 - y) * R(x, -1) + (y + 1) * P_b(x, H - 1) \quad (16)$$

[0099] The execution of the sequences of instructions required to practice the embodiments can be performed by a computer system 1000 as shown in FIG. 10. In an embodiment, execution of the sequences of instructions is performed by a single computer system 1000. According to other embodiments, two or more computer systems 1000 coupled by a communication link 1015 can perform the sequence of instructions in coordination with one another. Although a description of only one computer system 1000 will be presented below, however, it should be understood that any number of computer systems 1000 can be employed to practice the embodiments.

[00100] A computer system 1000 according to an embodiment will now be described with reference to FIG. 10, which is a block diagram of the functional components of a computer system

1000. As used herein, the term computer system 1000 is broadly used to describe any computing device that can store and independently run one or more programs.

[00101] Each computer system 1000 can include a communication interface 1014 coupled to the bus 1006. The communication interface 1014 provides two-way communication between computer systems 1000. The communication interface 1014 of a respective computer system 1000 transmits and receives electrical, electromagnetic or optical signals, which include data streams representing various types of signal information, e.g., instructions, messages and data. A communication link 1015 links one computer system 1000 with another computer system 1000. For example, the communication link 1015 can be a LAN, in which case the communication interface 1014 can be a LAN card, or the communication link 1015 can be a PSTN, in which case the communication interface 1014 can be an integrated services digital network (ISDN) card or a modem, or the communication link 1015 can be the Internet, in which case the communication interface 1014 can be a dial-up, cable or wireless modem.

[00102] A computer system 1000 can transmit and receive messages, data, and instructions, including program, i.e., application, code, through its respective communication link 1015 and communication interface 1014. Received program code can be executed by the respective processor(s) 1007 as it is received, and/or stored in the storage device 1010, or other associated non-volatile media, for later execution.

[00103] In an embodiment, the computer system 1000 operates in conjunction with a data storage system 1031, e.g., a data storage system 1031 that contains a database 1032 that is readily accessible by the computer system 1000. The computer system 1000 communicates with the data storage system 1031 through a data interface 1033. A data interface 1033, which is coupled to the bus 1006, transmits and receives electrical, electromagnetic or optical signals, which include data

streams representing various types of signal information, e.g., instructions, messages and data. In embodiments, the functions of the data interface 1033 can be performed by the communication interface 1014.

[00104] Computer system 1000 includes a bus 1006 or other communication mechanism for communicating instructions, messages and data, collectively, information, and one or more processors 1007 coupled with the bus 1006 for processing information. Computer system 1000 also includes a main memory 1008, such as a random-access memory (RAM) or other dynamic storage device, coupled to the bus 1006 for storing dynamic data and instructions to be executed by the processor(s) 1007. The main memory 1008 also can be used for storing temporary data, i.e., variables, or other intermediate information during execution of instructions by the processor(s) 1007.

[00105] The computer system 1000 can further include a read only memory (ROM) 1009 or other static storage device coupled to the bus 1006 for storing static data and instructions for the processor(s) 1007. A storage device 1010, such as a magnetic disk or optical disk, can also be provided and coupled to the bus 1006 for storing data and instructions for the processor(s) 1007.

[00106] A computer system 1000 can be coupled via the bus 1006 to a display device 1011, such as, but not limited to, a cathode ray tube (CRT) or a liquid-crystal display (LCD) monitor, for displaying information to a user. An input device 1012, e.g., alphanumeric and other keys, is coupled to the bus 1006 for communicating information and command selections to the processor(s) 1007.

[00107] According to one embodiment, an individual computer system 1000 performs specific operations by their respective processor(s) 1007 executing one or more sequences of one or more instructions contained in the main memory 1008. Such instructions can be read into the main

memory 1008 from another computer-usable medium, such as the ROM 1009 or the storage device 1010. Execution of the sequences of instructions contained in the main memory 1008 causes the processor(s) 1007 to perform the processes described herein. In alternative embodiments, hard-wired circuitry can be used in place of or in combination with software instructions. Thus, embodiments are not limited to any specific combination of hardware circuitry and/or software.

[00108] The term “computer-usable medium,” as used herein, refers to any medium that provides information or is usable by the processor(s) 1007. Such a medium can take many forms, including, but not limited to, non-volatile, volatile and transmission media. Non-volatile media, i.e., media that can retain information in the absence of power, includes the ROM 1009, CD ROM, magnetic tape, and magnetic discs. Volatile media, i.e., media that cannot retain information in the absence of power, includes the main memory 1008. Transmission media includes coaxial cables, copper wire and fiber optics, including the wires that comprise the bus 1006. Transmission media can also take the form of carrier waves; i.e., electromagnetic waves that can be modulated, as in frequency, amplitude or phase, to transmit information signals. Additionally, transmission media can take the form of acoustic or light waves, such as those generated during radio wave and infrared data communications. In the foregoing specification, the embodiments have been described with reference to specific elements thereof. It will, however, be evident that various modifications and changes can be made thereto without departing from the broader spirit and scope of the embodiments. For example, the reader is to understand that the specific ordering and combination of process actions shown in the process flow diagrams described herein is merely illustrative, and that using different or additional process actions, or a different combination or ordering of process actions can be used to enact the embodiments. The specification and drawings are, accordingly, to be regarded in an illustrative rather than restrictive sense

[00109] It should also be noted that the present invention can be implemented in a variety of computer systems. The various techniques described herein can be implemented in hardware or software, or a combination of both. Preferably, the techniques are implemented in computer programs executing on programmable computers that each include a processor, a storage medium readable by the processor (including volatile and non-volatile memory and/or storage elements), at least one input device, and at least one output device. Program code is applied to data entered using the input device to perform the functions described above and to generate output information. The output information is applied to one or more output devices. Each program is preferably implemented in a high-level procedural or object-oriented programming language to communicate with a computer system. However, the programs can be implemented in assembly or machine language, if desired. In any case, the language can be a compiled or interpreted language. Each such computer program is preferably stored on a storage medium or device (e.g., ROM or magnetic disk) that is readable by a general or special purpose programmable computer for configuring and operating the computer when the storage medium or device is read by the computer to perform the procedures described above. The system can also be considered to be implemented as a computer-readable storage medium, configured with a computer program, where the storage medium so configured causes a computer to operate in a specific and predefined manner. Further, the storage elements of the exemplary computing applications can be relational or sequential (flat file) type computing databases that are capable of storing data in various combinations and configurations.

[00110] FIG. 11 is a flow diagram that illustrates a method for performing the disclosed techniques, but it should be understood that the techniques described herein with respect to the remaining figures similarly capture the methods available using the disclosed techniques. As illustrated in FIG. 11, the method at 1102 includes calculating an intensity value of a bottom right

neighboring pixel of the current coding block, such as a coding unit or a luma or chroma block. At 1104, the bottom right intensity value can be used with a neighboring pixel from the column of vertical boundary pixels on the left side of the current coding block to calculate intensity values for the neighboring pixels in a row along the lower side of the current block. At 1106, the bottom right intensity value can be used with an intensity value of a neighboring pixel from the row of horizontal boundary pixels on the upper side of the current coding block to calculate intensity values for the neighboring pixels in a column along the right side of the current block. The method at 1108 and 1110 further comprises calculating a first and second predictor and, at 1112, the method includes deriving a prediction pixel value from the first and second predictors, wherein a plurality of prediction pixel values make up a prediction block.

[00111] FIG. 12 is a high-level view of a source device 12 and destination device 10 that may incorporate features of the systems and devices described herein. As shown in FIG. 12, example video coding system 10 includes a source device 12 and a destination device 14 where, in this example, the source device 12 generates encoded video data. Accordingly, source device 12 may be referred to as a video encoding device. Destination device 14 may decode the encoded video data generated by source device 12. Accordingly, destination device 14 may be referred to as a video decoding device. Source device 12 and destination device 14 may be examples of video coding devices.

[00112] Destination device 14 may receive encoded video data from source device 12 via a channel 16. Channel 16 may comprise a type of medium or device capable of moving the encoded video data from source device 12 to destination device 14. In one example, channel 16 may comprise a communication medium that enables source device 12 to transmit encoded video data directly to destination device 14 in real-time.

[00113] In this example, source device 12 may modulate the encoded video data according to a communication standard, such as a wireless communication protocol, and may transmit the modulated video data to destination device 14. The communication medium may comprise a wireless or wired communication medium, such as a radio frequency (RF) spectrum or one or more physical transmission lines. The communication medium may form part of a packet-based network, such as a local area network, a wide-area network, or a global network such as the Internet. The communication medium may include routers, switches, base stations, or other equipment that facilitates communication from source device 12 to destination device 14. In another example, channel 16 may correspond to a storage medium that stores the encoded video data generated by source device 12.

[00114] In the example of FIG. 12, source device 12 includes a video source 18, video encoder 20, and an output interface 22. In some cases, output interface 28 may include a modulator/demodulator (modem) and/or a transmitter. In source device 12, video source 18 may include a source such as a video capture device, e.g., a video camera, a video archive containing previously captured video data, a video feed interface to receive video data from a video content provider, and/or a computer graphics system for generating video data, or a combination of such sources.

[00115] Video encoder 20 may encode the captured, pre-captured, or computer-generated video data. An input image may be received by the video encoder 20 and stored in the input frame memory 21. The general-purpose processor 23 may load information from here and perform encoding. The program for driving the general-purpose processor may be loaded from a storage device, such as the example memory modules depicted in FIG. 12. The general-purpose processor

may use processing memory 22 to perform the encoding, and the output of the encoding information by the general processor may be stored in a buffer, such as output buffer 26.

[00116] The video encoder 20 may include a resampling module 25 which may be configured to code (e.g., encode) video data in a scalable video coding scheme that defines at least one base layer and at least one enhancement layer. Resampling module 25 may resample at least some video data as part of an encoding process, wherein resampling may be performed in an adaptive manner using resampling filters.

[00117] The encoded video data, e.g., a coded bit stream, may be transmitted directly to destination device 14 via output interface 28 of source device 12. In the example of FIG. 12, destination device 14 includes an input interface 38, a video decoder 30, and a display device 32. In some cases, input interface 28 may include a receiver and/or a modem. Input interface 38 of destination device 14 receives encoded video data over channel 16. The encoded video data may include a variety of syntax elements generated by video encoder 20 that represent the video data. Such syntax elements may be included with the encoded video data transmitted on a communication medium, stored on a storage medium, or stored a file server.

[00118] The encoded video data may also be stored onto a storage medium or a file server for later access by destination device 14 for decoding and/or playback. For example, the coded bitstream may be temporarily stored in the input buffer 31, then loaded in to the general-purpose processor 33. The program for driving the general-purpose processor may be loaded from a storage device or memory. The general-purpose processor may use a process memory 32 to perform the decoding. The video decoder 30 may also include a resampling module 35 similar to the resampling module 25 employed in the video encoder 20.

[00119] FIG. 12 depicts the resampling module 35 separately from the general-purpose processor 33, but it would be appreciated by one of skill in the art that the resampling function may be performed by a program executed by the general-purpose processor, and the processing in the video encoder may be accomplished using one or more processors. The decoded image(s) may be stored in the output frame buffer 36 and then sent out to the input interface 38.

[00120] Display device 38 may be integrated with or may be external to destination device 14. In some examples, destination device 14 may include an integrated display device and may also be configured to interface with an external display device. In other examples, destination device 14 may be a display device. In general, display device 38 displays the decoded video data to a user.

[00121] Video encoder 20 and video decoder 30 may operate according to a video compression standard. ITU-T VCEG (Q6/16) and ISO/IEC MPEG (JTC 1/SC 29/WG 11) are studying the potential need for standardization of future video coding technology with a compression capability that significantly exceeds that of the current High Efficiency Video Coding HEVC standard (including its current extensions and near-term extensions for screen content coding and high-dynamic-range coding). The groups are working together on this exploration activity in a joint collaboration effort known as the Joint Video Exploration Team (JVET) to evaluate compression technology designs proposed by their experts in this area. A recent capture of JVET development is described in the “Algorithm Description of Joint Exploration Test Model 5 (JEM 5)”, JVET-E1001-V2, authored by J. Chen, E. Alshina, G. Sullivan, J. Ohm, J. Boyce.

[00122] Additionally or alternatively, video encoder 20 and video decoder 30 may operate according to other proprietary or industry standards that function with the disclosed JVET features. Thus, other standards such as the ITU-T H.264 standard, alternatively referred to as MPEG-4, Part 10, Advanced Video Coding (AVC), or extensions of such standards. Thus, while newly developed

for JVET, techniques of this disclosure are not limited to any particular coding standard or technique. Other examples of video compression standards and techniques include MPEG-2, ITU-T H.263 and proprietary or open source compression formats and related formats.

[00123] Video encoder 20 and video decoder 30 may be implemented in hardware, software, firmware or any combination thereof. For example, the video encoder 20 and decoder 30 may employ one or more processors, digital signal processors (DSPs), application specific integrated circuits (ASICs), field programmable gate arrays (FPGAs), discrete logic, or any combinations thereof. When the video encoder 20 and decoder 30 are implemented partially in software, a device may store instructions for the software in a suitable, non-transitory computer-readable storage medium and may execute the instructions in hardware using one or more processors to perform the techniques of this disclosure. Each of video encoder 20 and video decoder 30 may be included in one or more encoders or decoders, either of which may be integrated as part of a combined encoder/decoder (CODEC) in a respective device.

[00124] Aspects of the subject matter described herein may be described in the general context of computer-executable instructions, such as program modules, being executed by a computer, such as the general-purpose processors 23 and 33 described above. Generally, program modules include routines, programs, objects, components, data structures, and so forth, which perform particular tasks or implement particular abstract data types. Aspects of the subject matter described herein may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote computer storage media including memory storage devices.

[00125] Examples of memory include random access memory (RAM), read only memory (ROM), or both. Memory may store instructions, such as source code or binary code, for performing the techniques described above. Memory may also be used for storing variables or other intermediate information during execution of instructions to be executed by a processor, such as processor 23 and 33.

[00126] A storage device may also store instructions, instructions, such as source code or binary code, for performing the techniques described above. A storage device may additionally store data used and manipulated by the computer processor. For example, a storage device in a video encoder 20 or a video decoder 30 may be a database that is accessed by computer system 23 or 33. Other examples of storage device include random access memory (RAM), read only memory (ROM), a hard drive, a magnetic disk, an optical disk, a CD-ROM, a DVD, a flash memory, a USB memory card, or any other medium from which a computer can read.

[00127] A memory or storage device may be an example of a non-transitory computer-readable storage medium for use by or in connection with the video encoder and/or decoder. The non-transitory computer-readable storage medium contains instructions for controlling a computer system to be configured to perform functions described by particular embodiments. The instructions, when executed by one or more computer processors, may be configured to perform that which is described in particular embodiments.

[00128] Also, it is noted that some embodiments have been described as a process which can be depicted as a flow diagram or block diagram. Although each may describe the operations as a sequential process, many of the operations can be performed in parallel or concurrently. In addition, the order of the operations may be rearranged. A process may have additional steps not included in the figures.

[00129] Particular embodiments may be implemented in a non-transitory computer-readable storage medium for use by or in connection with the instruction execution system, apparatus, system, or machine. The computer-readable storage medium contains instructions for controlling a computer system to perform a method described by particular embodiments. The computer system may include one or more computing devices. The instructions, when executed by one or more computer processors, may be configured to perform that which is described in particular embodiments

[00130] As used in the description herein and throughout the claims that follow, “a”, “an”, and “the” includes plural references unless the context clearly dictates otherwise. Also, as used in the description herein and throughout the claims that follow, the meaning of “in” includes “in” and “on” unless the context clearly dictates otherwise.

[00131] Although the subject matter has been described in language specific to structural features and/or methodological acts, it is to be understood that the subject matter defined in the appended claims is not necessarily limited to the specific features or acts described above.

[00132] Although embodiments have been disclosed herein in detail and in language specific to structural features and/or methodological acts above, it is to be understood that those skilled in the art will readily appreciate that many additional modifications are possible in the exemplary embodiments without materially departing from the novel teachings and advantages of the invention. Moreover, it is to be understood that the subject matter defined in the appended claims is not necessarily limited to the specific features or acts described above. Accordingly, these and all such modifications are intended to be included within the scope of this invention construed in breadth and scope in accordance with the appended claims.

CLAIMS

1. A method for using planar mode to predict pixel values for a current coding block having height H , width W , and a top-left (TL) pixel within the current coding block is defined by coordinates $x = 0$ and $y = 0$, where constructed or reconstructed neighboring pixels form a row of horizontal boundary pixels on an upper side of the current coding block and constructed or reconstructed neighboring pixels form a column of vertical boundary pixels on a left side of the current coding block, the neighboring pixels including a top right neighboring pixel, a top left neighboring pixel, and a bottom left neighboring pixel, the method configured to:

calculate an intensity value of a bottom right neighboring pixel of the current coding block;

use an intensity value of a neighboring pixel from the column of vertical boundary pixels on the left side of the current coding block and the calculated intensity value of the bottom right neighboring pixel, calculate intensity values for the neighboring pixels in a row along the lower side of the current coding block;

use an intensity value of a neighboring pixel from the row of horizontal boundary pixels on the upper side of the current coding block and the calculated intensity value of the bottom right neighboring pixel, calculate intensity values for the neighboring pixels in a column along the right side of the current coding block;

calculate a first predictor for a pixel at coordinate (x, y) in the current coding block by interpolating between an intensity value of a respective vertical neighboring pixel in the row of neighboring pixels along the upper side of the current coding block and at least one of the calculated intensity values for the neighboring pixels in the row along the lower side of the current coding block;

calculate a second predictor for the pixel at coordinate (x, y) in the current coding block by interpolating between an intensity value of a respective horizontal neighboring pixel from the row of neighboring pixels along the left side of the current coding block and at least one of the calculated intensity values for the neighboring pixels in the column along the right side of the current coding block; and

derive a prediction pixel value from the first and second predictors, wherein a plurality of prediction pixel values make up a prediction block.

2. The method of claim 1, wherein deriving the prediction pixel value comprises averaging the first and second predictors.

3. The method of claim 1, wherein the intensity value of the neighboring pixel from the row of horizontal boundary pixels on the upper side of the current coding block, used in calculating the intensity values for the neighboring pixels in the column along the right side of the current coding block, is a top right corner neighboring pixel.

4. The method of claim 1, wherein the intensity value of the neighboring pixel from the column of vertical boundary pixels on the left side of the current coding block, used in calculating the intensity values for the neighboring pixels in the column along the left side of the current coding block, is a bottom left corner neighboring pixel.

5. The method of claim 1, wherein calculating the intensity value of the bottom right neighboring pixel comprises calculating a weighted average of intensity values of two or more reconstructed neighboring pixels.

6. The method of claim 5, wherein the intensity value of at least one of the neighboring pixels is filtered for use in the weighted average calculation of the intensity values.

7. The method of claim 5, where $R(W, -1)$ denotes an intensity value of a top right corner neighboring pixel at coordinates $W, -1$ and $R(-1, H)$ denotes an intensity value of a bottom left corner neighboring pixel at coordinates $-1, H$, and the weighted average calculation is for the bottom right neighboring pixel $P(W, H)$ and comprises as at least one of:

$$P(W, H) = (W * R(W, -1) + H * R(-1, H)) / (H + W), \text{ or}$$

$$P(W, H) = R(W, -1) + R(-1, H) - R(0, 0).$$

8. The method of claim 7, wherein the top right corner neighboring pixel is filtered before use in the weighted average calculation of the bottom right neighboring pixel according to at least one of:

$$R_{filtered}(W, -1) = \frac{(R(W - 3, -1) + 3 * R(W - 2, -1) + 5 * R(W - 1, -1) + 7 * R(W, -1))}{16}$$

$$R_{filtered}(W, -1) = \frac{(R(W - 2, -1) + 2 * R(W - 1, -1) + 5 * R(W, -1))}{8}.$$

9. The method of claim 7, wherein the bottom left corner neighboring pixel is filtered before use in the weighted average calculation of the bottom right neighboring pixel according to at least one of:

$$R_{filtered}(-1, H) = \frac{(R(-1, H - 3) + 3 * R(-1, H - 2) + 5 * R(-1, H - 1) + 7 * R(-1, H))}{16}$$

$$R_{filtered}(-1, H) = \frac{(R(-1, H-2)+2*R(-1, H-1)+5*R(-1, H))}{8}.$$

10. The method of claim 1, wherein the neighboring pixel used in the calculation of intensity values of the neighboring pixels along the lower side of the current coding block is a bottom left corner neighboring pixel $R(-1, H)$ at coordinates $x = -1$ and $y = H$.

11. The method of claim 10, wherein the intensity values of the neighboring pixels along the lower side of the current coding block is computed by $P_b(x, H) = \frac{(W-1-x)*R(-1, H) + (x+1)*P(W, H)}{W}$.

12. The method of claim 11, wherein for each neighboring pixel $R(m, H) \dots R(W-1, H)$ along the lower side of the current coding block where integer $m < W$, filter the respective intensity value for use in the second predictor calculation.

13. The method of claim 1, wherein the neighboring pixel used in the calculation of the intensity values of the neighboring pixels in the column along the right side of the current coding block is a top right corner neighboring pixel $R(W, -1)$ at coordinates $x = W$ and $y = -1$ coordinates.

14. The method of claim 13, wherein the intensity values for the neighboring pixels in a column along the right side of the current coding block are computed by $P_r(W, y) = \frac{(H-1-y)*R(W, -1) + (y+1)*P(W, H)}{H}$

where $R(W, -1)$ denotes an intensity value of the top right corner neighboring pixel at $x=W$ and $y = -1$ coordinates.

15. The method of claim 14, wherein for each neighboring pixel $R(W, n) \dots R(W, H-1)$ in the column along the right side of the current coding block where integer $n < H$, filter the respective intensity value for use in the first predictor calculation.

16. The method of claim 14, wherein the first predictor is a vertical predictor and is determined according to $P_h(x, y) = (W - 1 - x) * R(-1, y) + (x + 1) * P_r(W, y)$.

17. The method of claim 11, wherein the second predictor is a horizontal predictor and is determined according to $P_v(x, y) = (H - 1 - y) * R(x, -1) + (y + 1) * P_b(x, H)$.

18. The method of claim 1, where the top right neighboring pixel is at coordinates $W-1, -1$ and the bottom left neighboring pixel is at coordinates $-1, H-1$, wherein calculating the intensity value of the bottom right neighboring pixel $P(W-1, H-1)$ comprises calculating a weighted average of intensity values of the top right neighboring pixel $R(W-1, -1)$ and the bottom left neighboring pixel $R(-1, H-1)$ according to $P(W - 1, H - 1) = (W * R(W - 1, -1) + H * R(-1, H - 1)) / (H + W)$.

19. The method of claim 1, wherein the neighboring pixel used in the calculation of the intensity values of the neighboring pixels in the row along the lower side of the current coding block is the bottom left neighboring pixel $R(-1, H-1)$, the intensity value of a neighboring pixel in

the row along the lower side of the current coding block computed by $P_b(x, H-1) = \frac{(W-1-x)*R(-1, H-1) + (x+1)*P(W-1, H-1)}{W}$

where $R(-1, H-1)$ denotes an intensity value of the bottom left neighboring pixel at $x = -1$ and $y = H-1$ coordinates.

20. The method of claim 1, wherein the neighboring pixel used in the calculation of the intensity values of the neighboring pixels in the column along the right side of the current coding block is the top right neighboring pixel $R(W-1, -1)$ at coordinates $x = W-1$ and $y = -1$, and the intensity values of the neighboring pixels in the column along the right side of the current coding block is computed by $P_r(W-1, y) = \frac{(H-1-y)*R(W-1, -1) + (y+1)*P(W-1, H-1)}{H}$.

21. The method of claim 19, wherein the first predictor is a horizontal predictor and is determined according to $P_h(x, y) = (W-1-x) * R(-1, y) + (x+1) * P_r(W-1, y)$.

22. The method of claim 20, wherein the second predictor is a vertical predictor and is determined according to $P_v(x, y) = (H-1-y) * R(x, -1) + (y+1) * P_b(x, H-1)$.

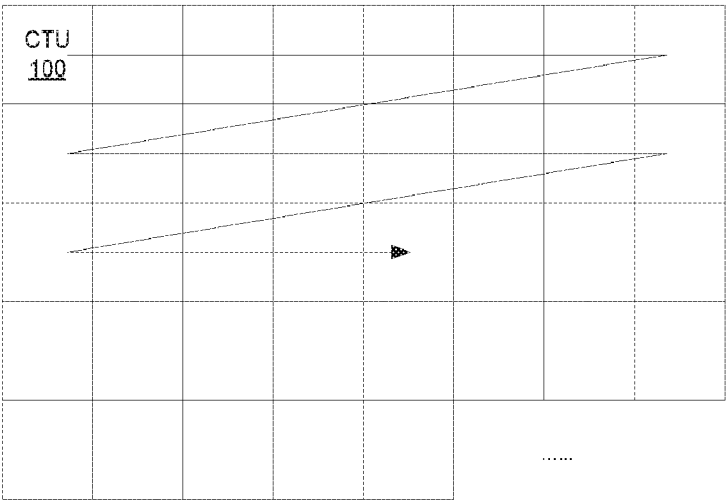


FIG. 1

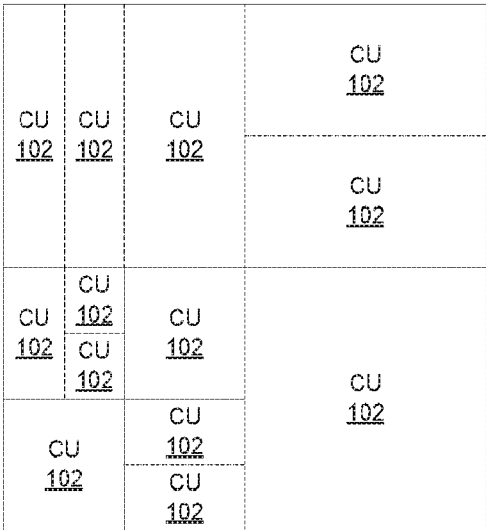


FIG. 2

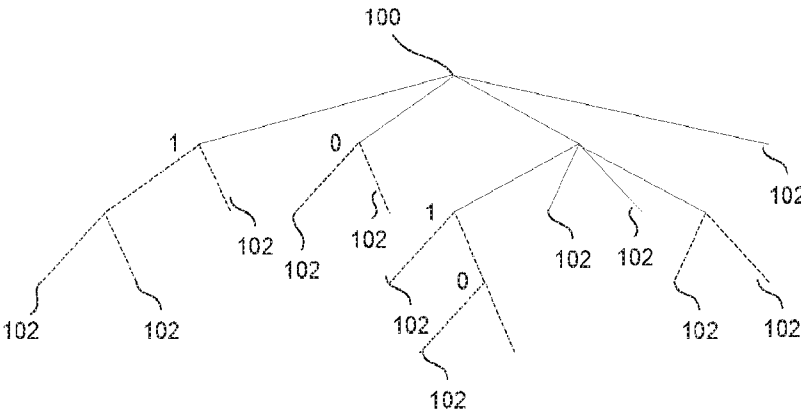


FIG. 3

3/11

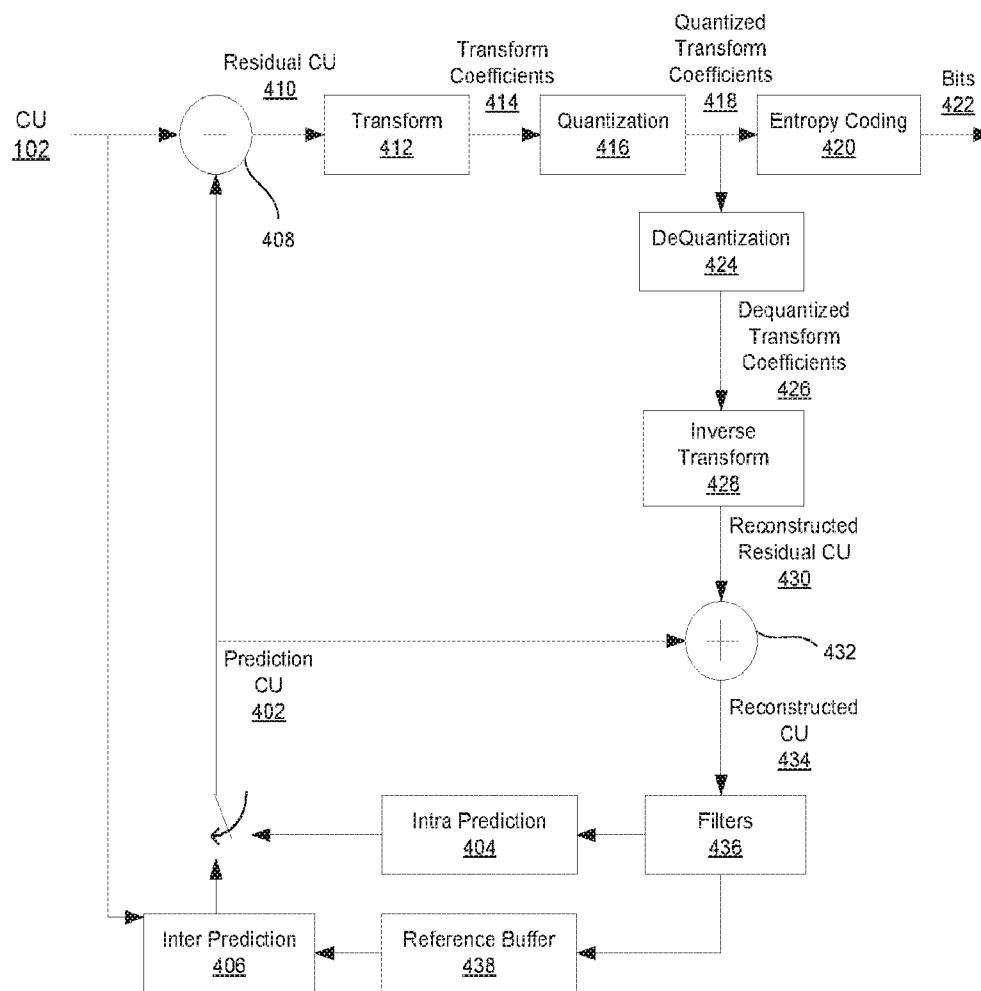
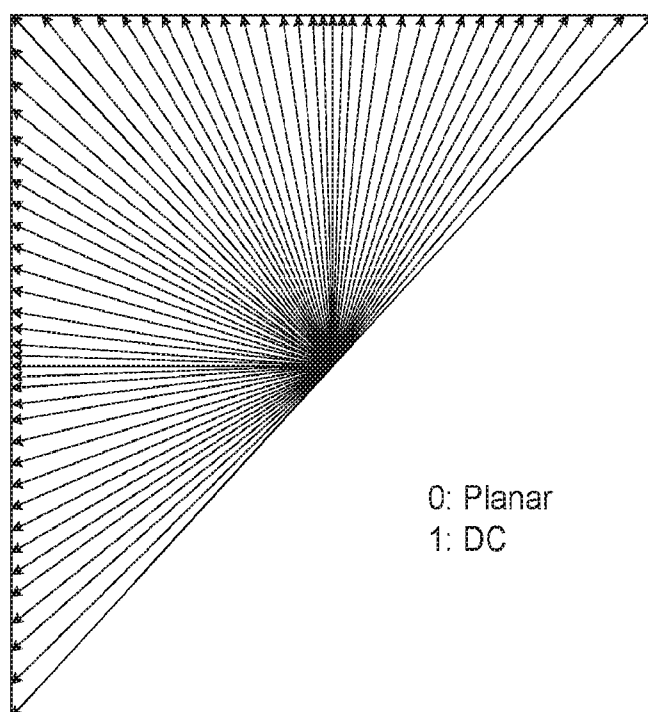


FIG. 4

4/11

*FIG. 5*

5/11

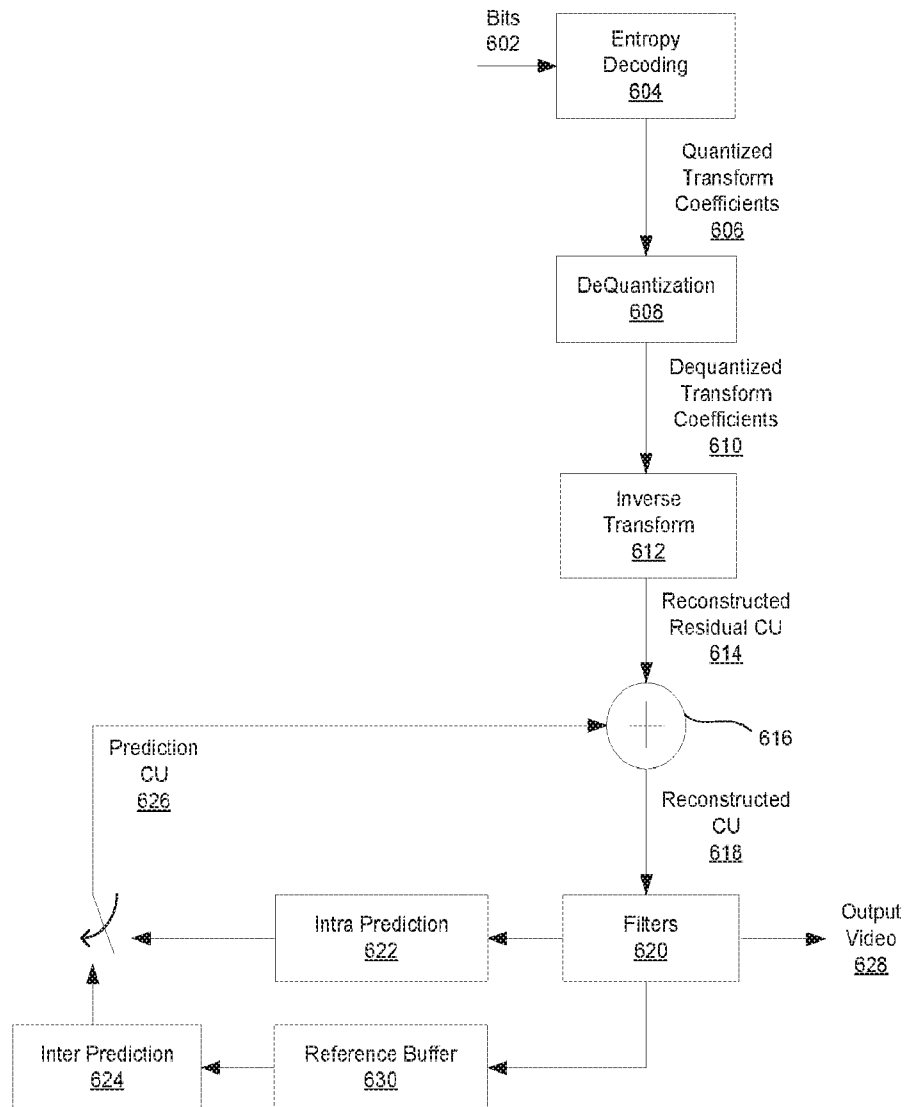


FIG. 6

6/11

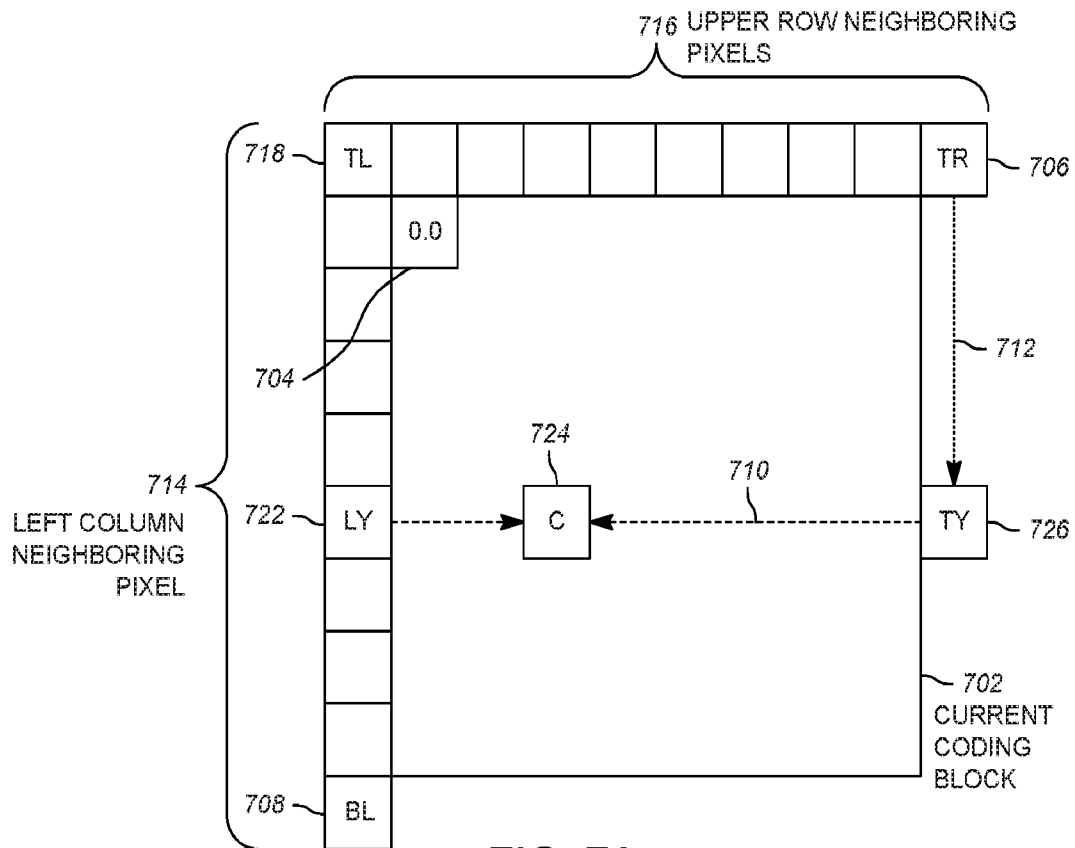


FIG. 7A

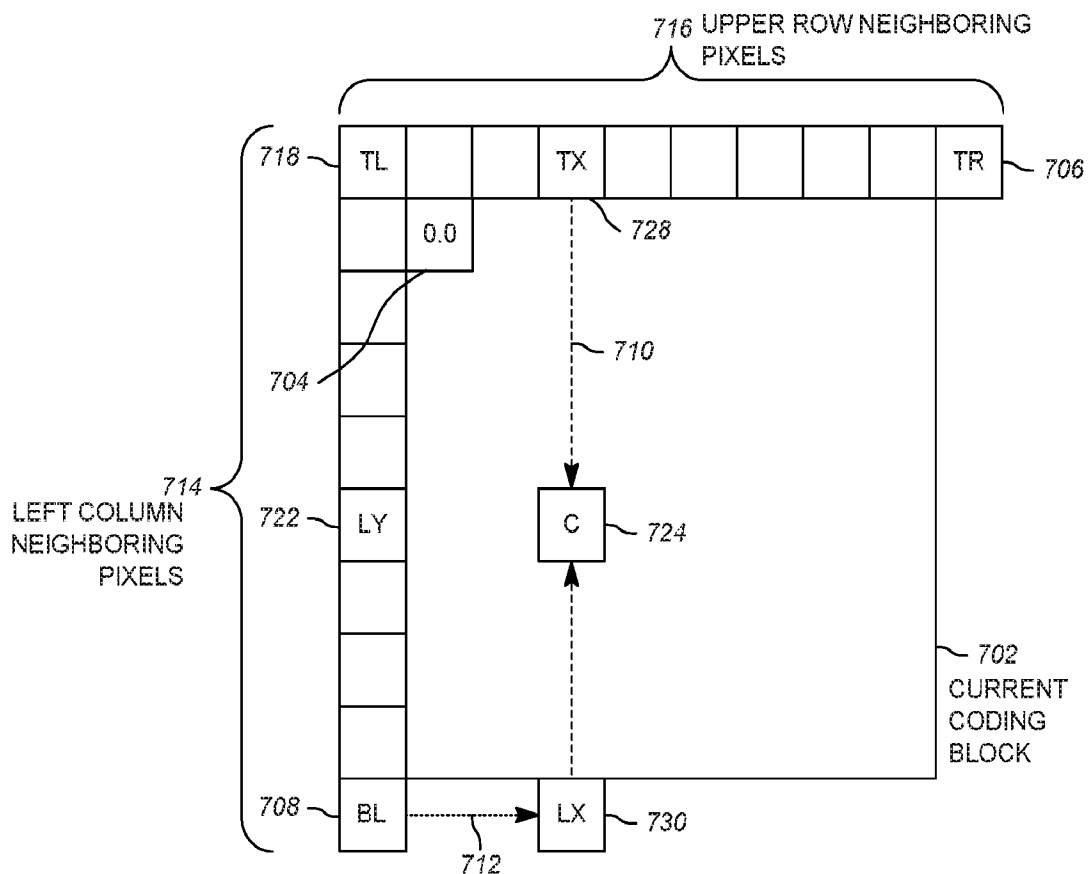


FIG. 7B

7/11

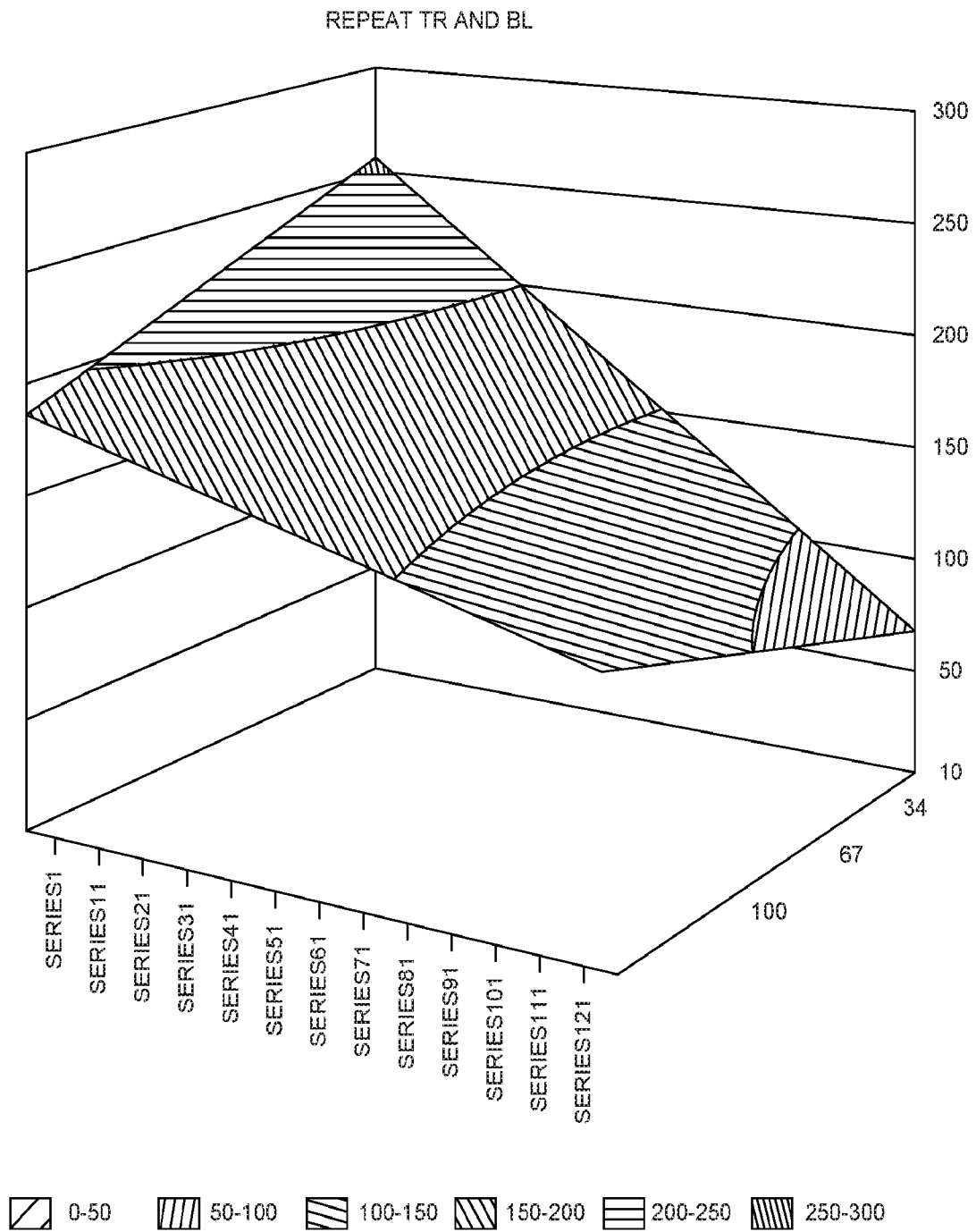


FIG. 8

8/11

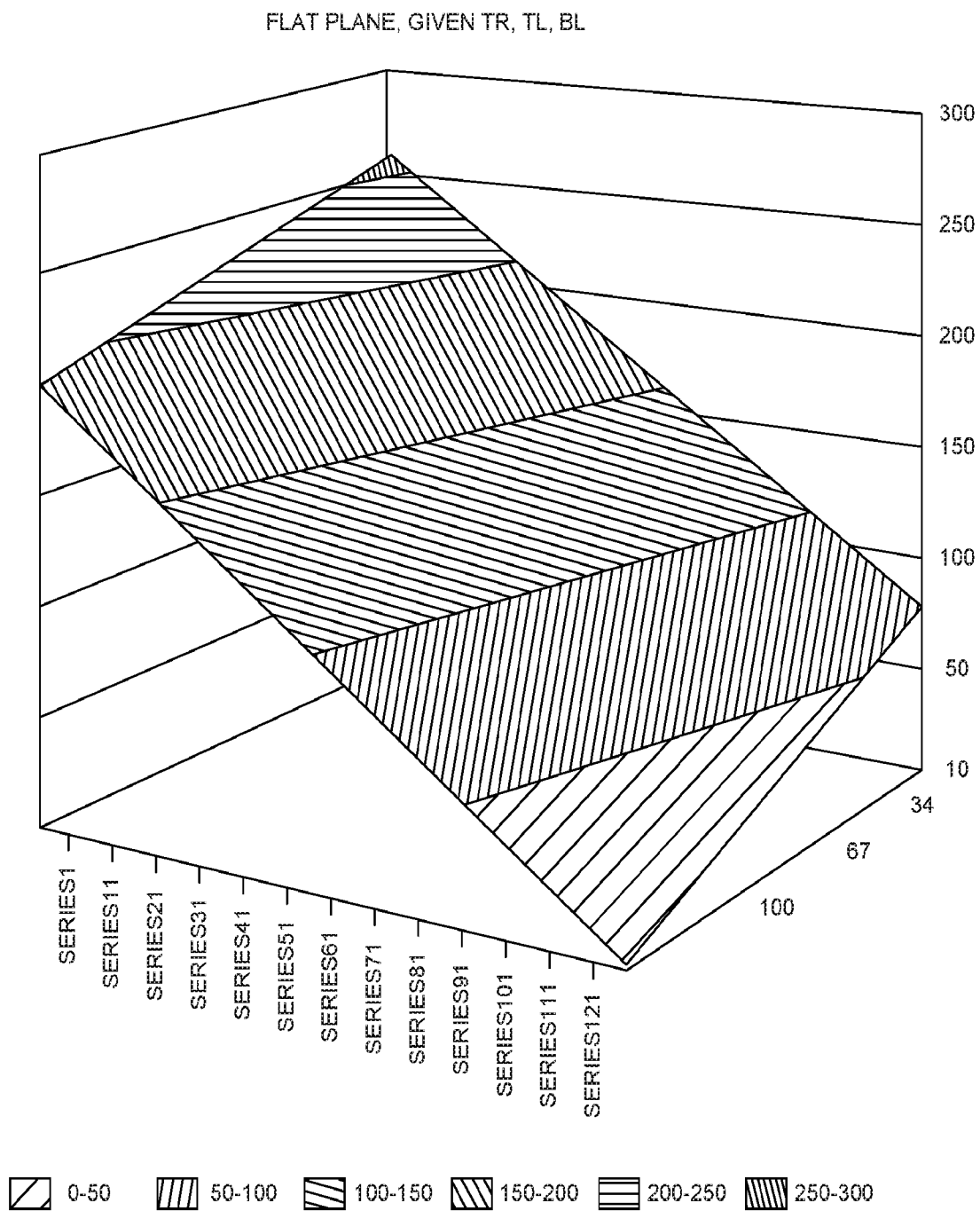


FIG. 9

9/11

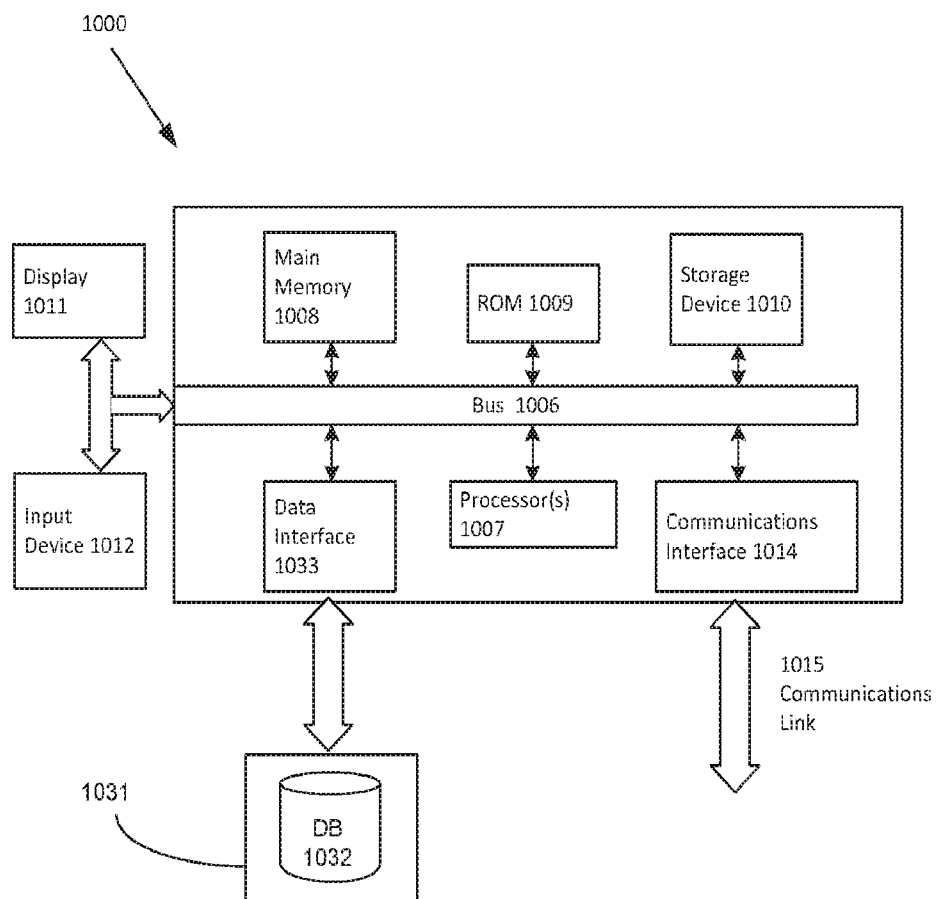


FIG. 10

10/11

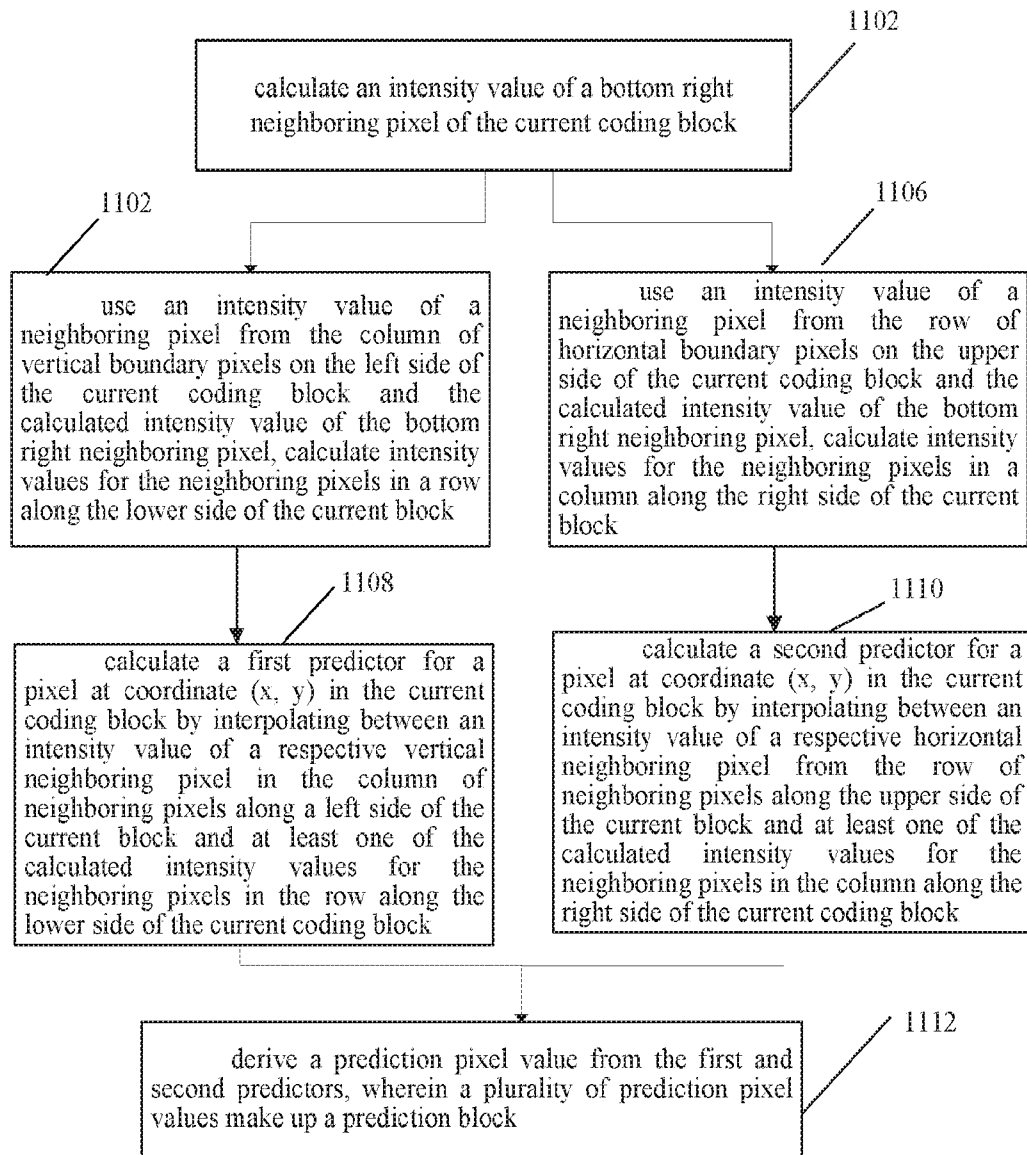


FIG. 11

11/11

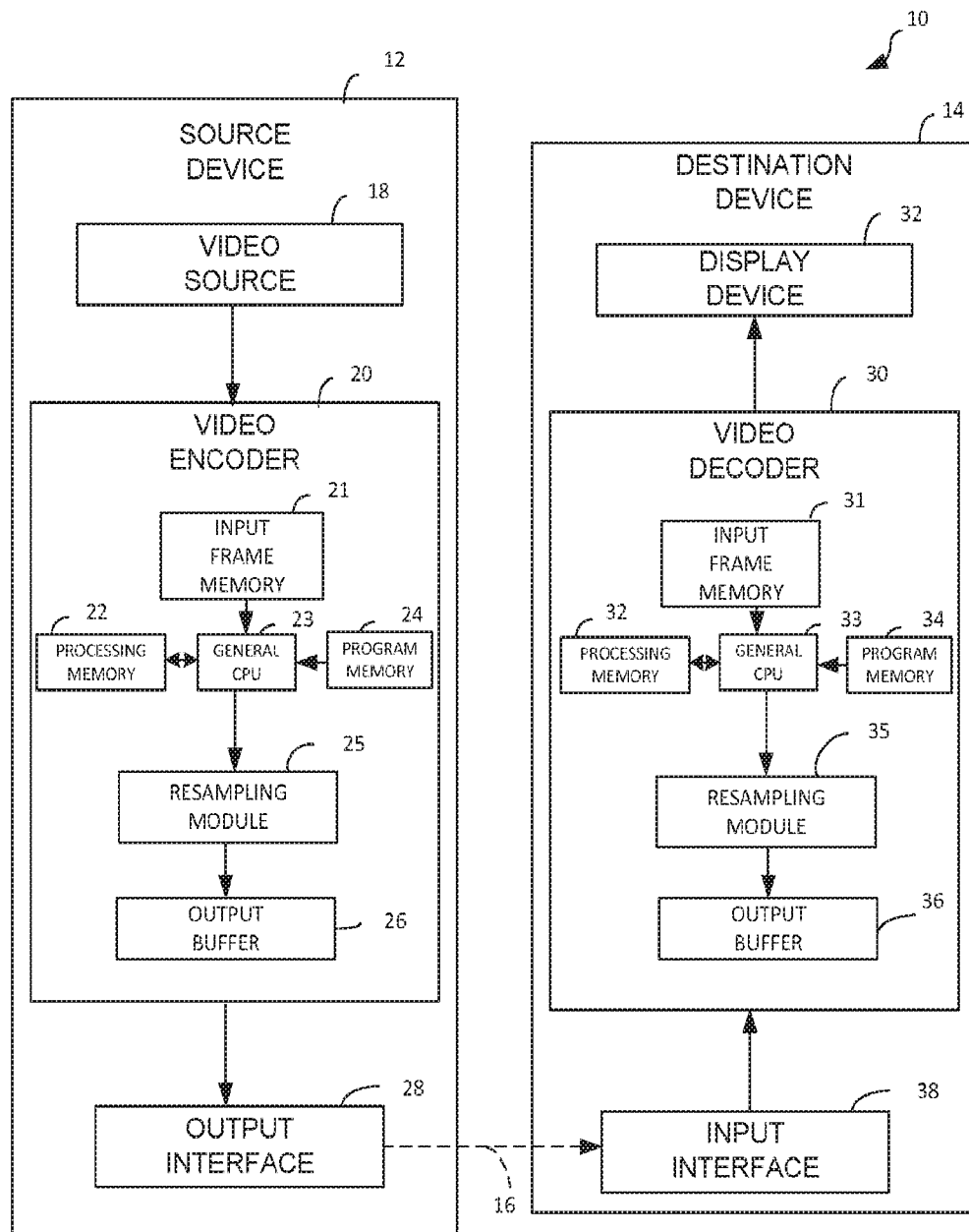


FIG. 12

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2017/068605

A. CLASSIFICATION OF SUBJECT MATTER
INV. H04N19/593 H04N19/105 H04N19/176 H04N19/11 H04N19/136
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
H04N

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	JANI LAINEMA ET AL: "Intra Coding of the HEVC Standard", IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, USA, vol. 22, no. 12, 1 December 2012 (2012-12-01), pages 1792-1801, XP011487148, ISSN: 1051-8215, DOI: 10.1109/TCSVT.2012.2221525 page 1795, paragraph II B; figure 1 ----- -/--	1-22



Further documents are listed in the continuation of Box C.



See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search

24 May 2018

Date of mailing of the international search report

08/06/2018

Name and mailing address of the ISA/

European Patent Office, P.B. 5818 Patentlaan 2
NL - 2280 HV Rijswijk
Tel. (+31-70) 340-2040,
Fax: (+31-70) 340-3016

Authorized officer

Di Cagno, Gianluca

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2017/068605

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	UGUR (NOKIA) K ET AL: "Description of video coding technology proposal by Tandberg, Nokia, Ericsson", 1. JCT-VC MEETING; 15-4-2010 - 23-4-2010; DRESDEN; (JOINT COLLABORATIVE TEAM ON VIDEO CODING OF ISO/IEC JTC1/SC29/WG11 AND ITU-T SG.16); URL: HTTP://WFTP3.ITU.INT/AV-ARCH/JCTVC-SITE/,, no. JCTVC-A119, 12 April 2010 (2010-04-12) , XP030009029, page 6, paragraph 2.2.1; figure 2 -----	1-22
A	EP 2 890 130 A1 (NIPPON TELEGRAPH & TELEPHONE [JP]) 1 July 2015 (2015-07-01) paragraph [0010] paragraph [0108] - paragraph [0113] claims 1-3 -----	1-22
A	EP 2 728 884 A2 (SAMSUNG ELECTRONICS CO LTD [KR]) 7 May 2014 (2014-05-07) figures 20B, 21 paragraphs [0172] - [0174] -----	1-22

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No

PCT/US2017/068605

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 2890130	A1	01-07-2015	CN 104838650 A 12-08-2015
			EP 2890130 A1 01-07-2015
			JP 5841670 B2 13-01-2016
			JP 6053220 B2 27-12-2016
			JP 2016027756 A 18-02-2016
			JP WO2014050971 A1 22-08-2016
			KR 20150042268 A 20-04-2015
			US 2015245021 A1 27-08-2015
			WO 2014050971 A1 03-04-2014

EP 2728884	A2	07-05-2014	AU 2012276407 A1 06-02-2014
			AU 2016201361 A1 24-03-2016
			AU 2017201660 A1 30-03-2017
			BR 112013033710 A2 27-06-2017
			CA 2840486 A1 03-01-2013
			CN 103765901 A 30-04-2014
			CN 104918055 A 16-09-2015
			CN 104954805 A 30-09-2015
			CN 105100808 A 25-11-2015
			CN 105100809 A 25-11-2015
			EP 2728884 A2 07-05-2014
			EP 2919466 A2 16-09-2015
			EP 2919467 A2 16-09-2015
			EP 2919468 A2 16-09-2015
			EP 2919469 A2 16-09-2015
			EP 3247115 A1 22-11-2017
			JP 5956572 B2 27-07-2016
			JP 6101734 B2 22-03-2017
			JP 6101735 B2 22-03-2017
			JP 6101736 B2 22-03-2017
			JP 6101737 B2 22-03-2017
			JP 2014523187 A 08-09-2014
			JP 2015167388 A 24-09-2015
			JP 2015167389 A 24-09-2015
			JP 2015181254 A 15-10-2015
			JP 2015181255 A 15-10-2015
			KR 20130002297 A 07-01-2013
			KR 20150009500 A 26-01-2015
			KR 20150059143 A 29-05-2015
			KR 20150059144 A 29-05-2015
			KR 20150059145 A 29-05-2015
			KR 20150059146 A 29-05-2015
			MX 337647 B 14-03-2016
			MX 349194 B 18-07-2017
			PH 12016500446 A1 06-06-2016
			PH 12016500448 A1 06-06-2016
			PH 12016500450 A1 06-06-2016
			PH 12016500451 A1 06-06-2016
			RU 2627033 C1 03-08-2017
			RU 2014102592 A 10-08-2015
			TW 201309037 A 16-02-2013
			TW 201639373 A 01-11-2016
			TW 201743618 A 16-12-2017
			US 2014133565 A1 15-05-2014
			US 2015264379 A1 17-09-2015
			US 2015264380 A1 17-09-2015
			US 2015264381 A1 17-09-2015
			US 2015264382 A1 17-09-2015

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No

PCT/US2017/068605

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
		US 2015288980 A1	08-10-2015
		WO 2013002586 A2	03-01-2013
		ZA 201400651 B	30-08-2017
