(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
18 July 2024 (18.07.2024)

WIPO | PCT

(10) International Publication Number
WO 2024/152003 A1

(51) International Patent Classification:
H04N 19/122 (2014.01) *H04N 19/70* (2014.01)
H04N 19/61 (2014.01) *H04N 19/176* (2014.01)
H04N 19/157 (2014.01)

(21) International Application Number:
PCT/US2024/011479

(22) International Filing Date:
12 January 2024 (12.01.2024)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
63/439,039 13 January 2023 (13.01.2023) US

(71) Applicant: **BEIJING DAJIA INTERNET INFORMATION TECHNOLOGY CO., LTD.** [CN/CN]; Room 101, 8th Floor, Building 12, No. 16, Xierqi West Road, Haidian District, Beijing 100085 (CN).

(72) Inventor; and
(71) Applicant *(for US only)*: **YAN, Ning** [CN/US]; 8910 University Center Lane, Suite 400, San Diego, California 92122 (US).

(72) Inventors: **XIU, Xiaoyu**; 8910 University Center Lane, Suite 400, San Diego, California 92122 (US). **CHEN, Wei**; 8910 University Center Lane, Suite 400, San Diego, California 92122 (US). **JHU, Hong-Jheng**; 8910 University Center Lane, Suite 400, San Diego, California 92122 (US). **KUO, Che-Wei**; 8910 University Center Lane, Suite 400, San Diego, California 92122 (US). **MA, Changyue**; 8910 University Center Lane, Suite 400, San Diego, California 92122 (US). **WANG, Xianglin**; 8910 University Center Lane, Suite 400, San Diego, California 92122 (US). **YU, Bing**; No. 29 Xi'erqi Middle Road, Qinghe Street, Haidian District, Beijing 100085 (CN).

(74) Agent: **TAN, Hao**; Arch & Lake LLP, 203 N. LaSalle Street, Suite 2100, Chicago, Illinois 60601 (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, CV, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IQ, IR, IS, IT, JM, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, MG, MK, MN, MU, 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, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

(54) **Title:** METHODS AND APPARATUS FOR TRANSFORM TRAINING AND CODING

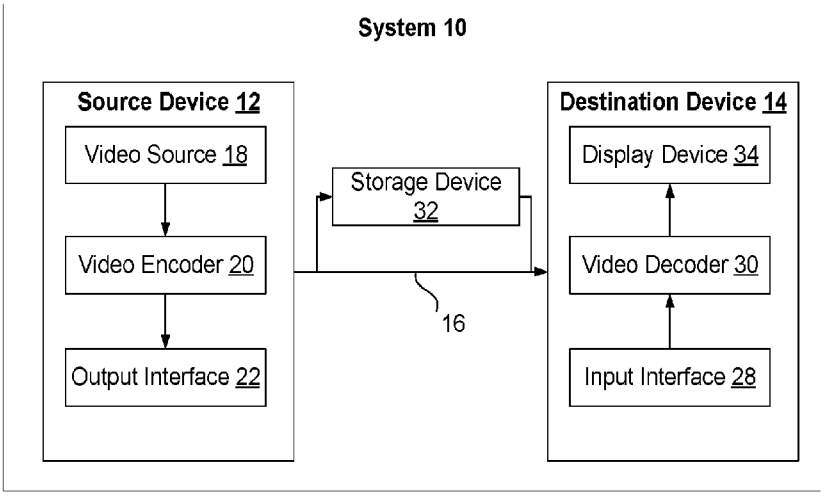

FIG. 1

(57) **Abstract:** This disclosure is related to video coding and compression. More specifically, this disclosure relates to methods and apparatus for transform training and coding. A method for video decoding is provided. The method includes: obtaining, by a decoder, a plurality of Karhunen-Loève Transform (KLT) transform kernels for a current block in an intra mode; and selecting, by the decoder, a KLT transform kernel for the current block in the intra mode from the plurality of KLT transform kernels.

WO 2024/152003 A1

[Continued on next page]

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, CV, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SC, 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, ME, 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).

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

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))*

Methods and Apparatus for Transform Training and Coding

CROSS-REFERENCE TO RELATED APPLICATION

[0001] The present application is based upon and claims priority to U.S. Provisional Application No. 63/439,039, entitled "Methods and Apparatus for Transform Training and Coding," filed on January 13, 2023, the entirety of which is incorporated by reference for all purposes.

BACKGROUND

[0002] Digital video is supported by a variety of electronic devices, such as digital televisions, laptop or desktop computers, tablet computers, digital cameras, digital recording devices, digital media players, video gaming consoles, smart phones, video teleconferencing devices, video streaming devices, etc. The electronic devices transmit and receive or otherwise communicate digital video data across a communication network, and/or store the digital video data on a storage device. Due to a limited bandwidth capacity of the communication network and limited memory resources of the storage device, video coding may be used to compress the video data according to one or more video coding standards before it is communicated or stored. For example, video coding standards include Versatile Video Coding (VVC), Joint Exploration test Model (JEM), High-Efficiency Video Coding (HEVC/H.265), Advanced Video Coding (AVC/H.264), Moving Picture Expert Group (MPEG) coding, or the like. Video coding generally utilizes prediction methods (e.g., inter-prediction, intra-prediction, or the like) that take advantage of redundancy inherent in the video data. Video coding aims to compress video data into a form that uses a lower bit rate, while avoiding or minimizing degradations to video quality.

SUMMARY

[0003] Embodiments of the present disclosure provide for transform training and coding.

[0004] In a first aspect, some embodiments of the present disclosure provide a method for video decoding including: obtaining, by a decoder, a plurality of Karhunen-Loève Transform (KLT) transform kernels for a current block in an intra mode; and selecting, by the decoder, a KLT transform kernel for the current block in the intra mode from the plurality of KLT transform kernels.

[0005] In a second aspect, some embodiments of the present disclosure provide a method for video encoding, including: obtaining, by an encoder, a plurality of Karhunen-Loève Transform

(KLT) transform kernels for a current block in an intra mode; determining, by the encoder, that KLT is enabled for the current block; and selecting, by the encoder, a KLT transform kernel for the current block in the intra mode from the plurality of KLT transform kernels.

[0006] In a third aspect, some embodiments of the present disclosure provide an apparatus for video decoding. The apparatus includes one or more processors; and a memory coupled to the one or more processors and configured to store instructions executable by the one or more processors, wherein the one or more processors, upon execution of the instructions are configured to perform the method according to the first aspect.

[0007] In a fourth aspect, some embodiments of the present disclosure provide an apparatus for video encoding. The apparatus includes one or more processors; and a memory coupled to the one or more processors and configured to store instructions executable by the one or more processors, wherein the one or more processors, upon execution of the instructions are configured to perform the method according to the second aspect.

[0008] In a fifth aspect, some embodiments of the present disclosure provide a non-transitory computer readable storage medium for storing computer-executable instructions that, when executed by one or more computer processors, cause the one or more computer processors to perform the method according to the first aspect.

[0009] In a sixth aspect, some embodiments of the present disclosure provide a non-transitory computer readable storage medium for storing computer-executable instructions that, when executed by one or more computer processors, cause the one or more computer processors to perform the method according to the second aspect.

[0010] In a seventh aspect, some embodiments of the present disclosure provide a non-transitory computer-readable storage medium for storing a bitstream to be decoded by the method according to the first aspect.

[0011] In an eighth aspect, some embodiments of the present disclosure provide a non-transitory computer-readable storage medium for storing a bitstream to be decoded by the method according to the second aspect.

[0012] It is to be understood that both the foregoing general description and the following detailed description are examples only and are not restrictive of the present disclosure.

BRIEF DESCRIPTION OF THE DRAWINGS

[0013] The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate examples consistent with the present disclosure and, together with the description, serve to explain the principles of the disclosure.

[0014] FIG. 1 is a block diagram illustrating an exemplary system for encoding and decoding video blocks in accordance with some implementations of the present disclosure.

[0015] FIG. 2 is a block diagram illustrating an exemplary video encoder in accordance with some implementations of the present disclosure.

[0016] FIG. 3A is a block diagram illustrating an exemplary video decoder in accordance with some implementations of the present disclosure.

[0017] FIG. 3B is an illustration of a general block diagram of a video decoder in accordance with some implementations of the present disclosure.

[0018] FIG. 3C is an illustration of a low-frequency non-separable transform (LFNST) process in accordance with some implementations of the present disclosure.

[0019] FIGS. 4A through 4E are block diagrams illustrating how a frame is recursively partitioned into multiple video blocks of different sizes and shapes in accordance with some implementations of the present disclosure. Particularly, FIG. 4A illustrates generating an encoded representation of a frame; FIG. 4B illustrates composition of a CTU, FIG. 4C illustrates division of CTUs, FIG. 4D depicts a quad-tree data structure illustrating the end result of the partition process of the CTU as depicted in FIG. 4C, and FIG. 4E illustrates five possible partitioning types of a coding block.

[0020] FIG. 5 is SBT position, type and transform type in accordance with some examples of the present disclosure.

[0021] FIG. 6 is the ROI for LFNST16 in accordance with some examples of the present disclosure.

[0022] FIG. 7 is an illustration of the ROI for LFNST8 in accordance with some examples of the present disclosure.

[0023] FIG. 8 is a flowchart of KLT exploring the non-local correlations in accordance with some examples of the present disclosure.

[0024] FIG. 9 is an illustration of angular intra prediction modes in VVC and ECM in accordance with some examples of the present disclosure.

[0025] FIG. 10 is an illustration of the template used for transform block clustering in accordance with some examples of the present disclosure.

[0026] FIG. 11 is an illustration of training data clustering based on prediction mode and neighboring template, in accordance with some examples of the present disclosure.

[0027] FIG. 12A is an illustration of an example that the transform coefficients in the left-above area are retained in accordance with some examples of the present disclosure.

[0028] FIG. 12B is an illustration of an example that the transform coefficients in the above area are retained in accordance with some examples of the present disclosure.

[0029] FIG. 12C is an illustration of an example that the transform coefficients in the left area are retained in accordance with some examples of the present disclosure.

[0030] FIG. 12D is an illustration of an example that the transform coefficients in the left and above areas are retained, where the left and above areas include both the above area and the left area in accordance with some examples of the present disclosure.

[0031] FIG. 13 is an illustration of block transpose and intra mode mapping for KLT matrix sharing in accordance with some examples of the present disclosure.

[0032] FIG. 14 is a flowchart illustrating a method for video decoding in accordance with some examples of the present disclosure.

[0033] FIG. 15 is a flowchart illustrating a method for video encoding corresponding to the method for video decoding illustrated in FIG. 14 and in accordance with some examples of the present disclosure.

[0034] FIG. 16 is a diagram illustrating a computing environment coupled with a user interface, in accordance with some implementations of the present disclosure.

DETAILED DESCRIPTION

[0035] Reference will now be made in detail to specific implementations, embodiments of which are illustrated in the accompanying drawings. In the following detailed description, numerous non-limiting specific details are set forth in order to assist in understanding the subject matter presented herein. But it will be apparent to one of ordinary skill in the art that various alternatives may be used. For example, it will be apparent to one of ordinary skill in the art that the subject matter presented herein can be implemented on many types of electronic devices with digital video capabilities.

[0036] It should be illustrated that the terms "first," "second," and the like used in the description, claims of the present disclosure, and the accompanying drawings are used to distinguish objects, and not used to describe any specific order or sequence. It should be understood that the data used in this way may be interchanged under an appropriate condition, such that the embodiments of the present disclosure described herein may be implemented in orders besides those shown in the accompanying drawings or described in the present disclosure.

[0037] ***Large Block-size Transforms with High-frequency Zeroing***

[0038] In VVC, large block-size transforms, up to 64×64 in size, are enabled, which is primarily useful for higher resolution video, e.g., 1080p and 4K sequences. High frequency transform coefficients are zeroed out for the transform blocks with size (width or height, or both width and height) equal to 64, so that only the lower-frequency coefficients are retained. For example, for an M×N transform block, with M as the block width and N as the block height, when M is equal to 64, only the left 32 columns of transform coefficients are kept. Similarly, when N is equal to 64, only the top 32 rows of transform coefficients are kept. When transform skip mode is used for a large block, the entire block is used without zeroing out any values. In addition, transform shift is removed in transform skip mode. The VTM also supports configurable max transform size in SPS, such that encoder has the flexibility to choose up to 32-length or 64-length transform size depending on the need of specific implementation.

[0039] ***Multiple Transform Selection (MTS) for Core Transform***

[0040] In addition to DCT-II which has been employed in HEVC, a Multiple Transform Selection (MTS) scheme is used for residual coding both inter and intra coded blocks. It uses multiple selected transforms from the DCT8/DST7. The newly introduced transform matrices are DST-VII and DCT-VIII. Table 1 shows the basis functions of the selected DST/DCT.

[0041] In order to keep the orthogonality of the transform matrix, the transform matrices are quantized more accurately than the transform matrices in HEVC. To keep the intermediate values of the transformed coefficients within the 16-bit range, after horizontal and after vertical transform, all the coefficients are to have 10-bit.

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

Table 1 Transform basis functions of DCT-II/ VIII and DSTVII for N-point input.

[0042] In order to control MTS scheme, separate enabling flags are specified at SPS level for intra and inter, respectively. When MTS is enabled at SPS, a CU level flag is signaled to indicate whether MTS is applied or not. Here, MTS is applied only for luma. The MTS signaling is skipped when one of the below conditions is applied. The position of the last significant coefficient for the luma TB is less than 1 (i.e., DC only). The last significant coefficient of the luma TB is located inside the MTS zero-out region. If MTS CU flag is equal to zero, then DCT2 is applied in both directions. However, if MTS CU flag is equal to one, then two other flags are additionally signaled to indicate the transform type for the horizontal and vertical directions, respectively. Transform and signaling mapping table as shown in Table 2. Unified the transform selection for ISP and implicit MTS is used by removing the intra-mode and block-shape dependencies. If current block is ISP mode or if the current block is intra block and both intra and inter explicit MTS is on, then only DST7 is used for both horizontal and vertical transform cores. When it comes to transform matrix precision, 8-bit primary transform cores are used. Therefore, all the transform cores used in HEVC are kept as the same, including 4-point DCT-2 and DST-7, 8-point, 16-point and 32-point DCT-2. Also, other transform cores including 64-point DCT-2, 4-point DCT-8, 8-point, 16-point, 32-point DST-7 and DCT-8, use 8-bit primary transform cores.

MTS_CU_flag	MTS_Hor_flag	MTS_Ver_flag	Intra/inter	
			Horizontal	Vertical

0			DCT2	
1	0	0	DST7	DST7
	0	1	DCT8	DST7
	1	0	DST7	DCT8
	1	1	DCT8	DCT8

Table 2 Transform and signaling mapping table.

[0043] To reduce the complexity of large size DST-7 and DCT-8, High frequency transform coefficients are zeroed out for the DST-7 and DCT-8 blocks with size (width or height, or both width and height) equal to 32. Only the coefficients within the 16x16 lower-frequency region are retained.

[0044] As in HEVC, the residual of a block can be coded with transform skip mode. To avoid the redundancy of syntax coding, the transform skip flag is not signaled when the CU level MTS_CU_flag is not equal to zero. Note that implicit MTS transform is set to DCT2 when LFNST or MIP is activated for the current CU. Also the implicit MTS can be still enabled when MTS is enabled for inter coded blocks.

[0045] ***Low-frequency Non-separable Transform (LFNST)***

[0046] In VVC, LFNST is applied between forward primary transform and quantization (at encoder) and between de-quantization and inverse primary transform (at decoder side) as shown in Figure 4. In LFNST, 4x4 non-separable transform or 8x8 non-separable transform is applied according to block size. For example, 4x4 LFNST is applied for small blocks (i.e., min (width, height) < 8) and 8x8 LFNST is applied for larger blocks (i.e., min (width, height) > 4).

[0047] Application of a non-separable transform, which is being used in LFNST, is described as follows using input as an example. To apply 4x4 LFNST, the 4x4 input block X

$$X = \begin{bmatrix} X_{00} & X_{01} & X_{02} & X_{03} \\ X_{10} & X_{11} & X_{12} & X_{13} \\ X_{20} & X_{21} & X_{22} & X_{23} \\ X_{30} & X_{31} & X_{32} & X_{33} \end{bmatrix} \quad (3\text{-}1)$$

[0048] is first represented as a vector $\vec{X}$:

$$\vec{X} = [X_{00} \quad X_{01} \quad X_{02} \quad X_{03} \quad X_{10} \quad X_{11} \quad X_{12} \quad X_{13} \quad X_{20} \quad X_{21} \quad X_{22} \quad X_{23} \quad X_{30} \quad X_{31} \quad X_{32} \quad X_{33}]^T \quad (3\text{-}2)$$

[0049] The non-separable transform is calculated as $\vec{F} = T \cdot \vec{X}$, where $\vec{F}$ indicates the transform coefficient vector, and T is a 16x16 transform matrix. The 16x1 coefficient vector $\vec{F}$ is subsequently re-organized as 4x4 block using the scanning order for that block (horizontal, vertical or diagonal). The coefficients with smaller index will be placed with the smaller scanning index in the 4x4 coefficient block.

[0050] ***Reduced non-separable transform***

[0051] LFNST (low-frequency non-separable transform) is based on direct matrix multiplication approach to apply non-separable transform so that it is implemented in a single pass without multiple iterations. However, the non-separable transform matrix dimension needs to be reduced to minimize computational complexity and memory space to store the transform coefficients. Hence, reduced non-separable transform (or RST) method is used in LFNST. The main idea of the reduced non-separable transform is to map an N (N is commonly equal to 64 for 8x8 NSST) dimensional vector to an R dimensional vector in a different space, where N/R (R < N) is the reduction factor. Hence, instead of NxN matrix, RST matrix becomes an R×N matrix as follows:

$$T_{RxN} = \begin{bmatrix} t_{11} & t_{12} & t_{13} & \cdots & t_{1N} \\ t_{21} & t_{22} & t_{23} & & t_{2N} \\ & \vdots & & \ddots & \vdots \\ t_{R1} & t_{R2} & t_{R3} & \cdots & t_{RN} \end{bmatrix} \qquad (3\text{-}3)$$

[0052] where the R rows of the transform are R bases of the N dimensional space. The inverse transform matrix for RT is the transpose of its forward transform. For 8x8 LFNST, a reduction factor of 4 is applied, and 64x64 direct matrix, which is conventional 8x8 non-separable transform matrix size, is reduced to16x48 direct matrix. Hence, the 48×16 inverse RST matrix is used at the decoder side to generate core (primary) transform coefficients in 8×8 top-left regions. When16x48 matrices are applied instead of 16x64 with the same transform set configuration, each of which takes 48 input data from three 4x4 blocks in a top-left 8x8 block excluding right-bottom 4x4 block. With the help of the reduced dimension, memory usage for storing all LFNST matrices is reduced from 10KB to 8KB with reasonable performance drop. In order to reduce complexity LFNST is restricted to be applicable only if all coefficients outside the first coefficient sub-group are non-significant. Hence, all primary-only transform coefficients have to be zero when LFNST is applied. This allows a conditioning of the LFNST index signalling on the last-significant position, and

hence avoids the extra coefficient scanning in the current LFNST design, which is needed for checking for significant coefficients at specific positions only.

[0053] The worst-case handling of LFNST (in terms of multiplications per pixel) restricts the non-separable transforms for 4x4 and 8x8 blocks to 8x16 and 8x48 transforms, respectively. In those cases, the last-significant scan position has to be less than 8 when LFNST is applied, for other sizes less than 16. For blocks with a shape of 4xN and Nx4 and N > 8, the proposed restriction implies that the LFNST is now applied only once, and that to the top-left 4x4 region only. As all primary-only coefficients are zero when LFNST is applied, the number of operations needed for the primary transforms is reduced in such cases. From encoder perspective, the quantization of coefficients is remarkably simplified when LFNST transforms are tested. A rate-distortion optimized quantization has to be done at maximum for the first 16 coefficients (in scan order), the remaining coefficients are enforced to be zero.

[0054] ***LFNST transform selection***

[0055] There are totally 4 transform sets and 2 non-separable transform matrices (kernels) per transform set are used in LFNST. The mapping from the intra prediction mode to the transform set can be pre-defined. If one of three CCLM modes (INTRA_LT_CCLM, INTRA_T_CCLM or INTRA_L_CCLM) is used for the current block (81 <= predModeIntra <= 83), transform set 0 is selected for the current chroma block. For each transform set, the selected non-separable secondary transform candidate is further specified by the explicitly signaled LFNST index. The index is signaled in a bit-stream once per Intra CU after transform coefficients.

IntraPredMode	Tr. set index
IntraPredMode < 0	1
0 <= IntraPredMode <= 1	0
2 <= IntraPredMode <= 12	1
13 <= IntraPredMode <= 23	2
24 <= IntraPredMode <= 44	3
45 <= IntraPredMode <= 55	2
56 <= IntraPredMode<= 80	1
81 <= IntraPredMode<= 83	0

Table 3 is a Transform Selection Table

[0056] ***LFNST index signaling and interaction with other tools***

[0057] Since LFNST is restricted to be applicable only if all coefficients outside the first coefficient sub-group are non-significant, LFNST index coding depends on the position of the last significant coefficient. In addition, the LFNST index is context coded but does not depend on intra prediction mode, and only the first bin is context coded. Furthermore, LFNST is applied for intra CU in both intra and inter slices, and for both Luma and Chroma. If a dual tree is enabled, LFNST indices for Luma and Chroma are signaled separately. For inter slice (the dual tree is disabled), a single LFNST index is signaled and used for both Luma and Chroma.

[0058] Considering that a large CU greater than 64x64 is implicitly split (TU tiling) due to the existing maximum transform size restriction (64x64), an LFNST index search could increase data buffering by four times for a certain number of decode pipeline stages. Therefore, the maximum size that LFNST is allowed is restricted to 64x64. Note that LFNST is enabled with DCT2 only. The LFNST index signaling is placed before MTS index signaling.

[0059] The use of scaling matrices for perceptual quantization is not evident that the scaling matrices that are specified for the primary matrices may be useful for LFNST coefficients. Hence, the uses of the scaling matrices for LFNST coefficients are not allowed. For single-tree partition mode, chroma LFNST is not applied.

[0060] ***LFNST training***

[0061] The coding efficiency of LFNST highly depends on the design of LFNST kernels, which are derived by off-line training. The training process can be considered as a clustering problem, where each cluster represents a huge group of transform coefficient blocks retrieved from the actual encoding process, and the 'centroid' of each cluster is the optimal non-separable transform, i.e., KLT, for the associated transform coefficient blocks in the same cluster.

[0062] Enlightened by the classical k-means clustering method, the training of the LFNST is performed in a two-stage iterative manner with an initial state:

[0063] ***Initialization***

[0064] For each transform coefficient block collected from the encoding process, a random label, ranging from 0 to 3, is assigned. Then the low-frequency M×N coefficients are added as one training data in the cluster associated with the assigned label. For each cluster labeled from 1 to 3, the optimal non-separable transform is derived by the solving eigenvectors of a covariance matrix, e.g., singular value decomposition (SVD), which is calculated using the training data in the same cluster. In addition, an identity transform, which means no secondary transform is applied, is assigned as the centroid of the first cluster.

[0065] ***Iteration***

[0066] For each available training data, select the best transform kernel using rate-distortion optimization and relabel the training data using the selected transform kernel. With the updated label of each training data, each cluster is updated, and the 'centroids' (transform kernels) of cluster labeled from 1 to 3 are updated accordingly. The identity transform is always assigned to cluster 0.

[0067] ***Subblock Transform (SBT)***

[0068] In VTM, subblock transform is introduced for an inter-predicted CU. In this transform mode, only a sub-part of the residual block is coded for the CU. When inter-predicted CU with cu_cbf equal to 1, cu_sbt_flag may be signaled to indicate whether the whole residual block or a sub-part of the residual block is coded. In the former case, inter MTS information is further parsed to determine the transform type of the CU. In the latter case, a part of the residual block is coded with inferred adaptive transform and the other part of the residual block is zeroed out.

[0069] When SBT is used for an inter-coded CU, SBT type and SBT position information are signaled in the bitstream. There are two SBT types and two SBT positions, as indicated in FIG. 5. For SBT-V (or SBT-H), the TU width (or height) may equal to half of the CU width (or height) or

1/4 of the CU width (or height), resulting in 2:2 split or 1:3/3:1 split. The 2:2 split is like a binary tree (BT) split while the 1:3/3:1 split is like an asymmetric binary tree (ABT) split. In ABT splitting, only the small region contains the non-zero residual. If one dimension of a CU is 8 in luma samples, the 1:3/3:1 split along that dimension is disallowed. There are at most 8 SBT modes for a CU.

[0070] Position-dependent transform core selection is applied on luma transform blocks in SBT-V and SBT-H (chroma TB always using DCT-2). The two positions of SBT-H and SBT-V are associated with different core transforms. More specifically, the horizontal and vertical transforms for each SBT position is specified in Figure 41. For example, the horizontal and vertical transforms for SBT-V position 0 is DCT-8 and DST-7, respectively. When one side of the residual TU is greater than 32, the transform for both dimensions is set as DCT-2. Therefore, the subblock transform jointly specifies the TU tiling, cbf, and horizontal and vertical core transform type of a residual block. The SBT is not applied to the CU coded with combined inter-intra mode.

[0071] ***Transform Improvement in the ECM***

[0072] Maximum transform size and zeroing-out of transform coefficients. Both CTU size and maximum transform size (i.e., all MTS transform kernels) are extended to 256, where the maximum intra coded block can have a size of 128x128. The maximum CTU size is set to 256 for UHD sequences and it is set to 128, otherwise. In the primary transformation process, there is no normative zeroing out operation applied on transform coefficients. However, if LFNST is applied, the primary transform coefficients outside the LFNST region are normatively zeroed-out.

[0073] ***Enhanced MTS for intra coding***

[0074] In the current VVC design [1], for MTS, only DST7 and DCT8 transform kernels are utilized which are used for intra and inter coding. Additional primary transforms including DCT5, DST4, DST1, and identity transform (IDT) are employed. Also MTS set is made dependent on the TU size and intra mode information. 16 different TU sizes are considered, and for each TU size 5 different classes are considered depending on intra-mode information. For each class, 1, 4 or 6 different transform pairs are considered. Number of intra MTS candidates are adaptively selected (between 1, 4 and 6 MTS candidates) depending on the sum of absolute value of transform coefficients. The sum is compared against the two fixed thresholds to determine the total number of allowed MTS candidates: 1 candidate: sum <= th0; 4 candidates: th0 < sum <= th1; 6 candidates: sum > th1.

[0075] Note, although a total of 80 different classes are considered, some of those different classes often share exactly same transform set. So there are 58 (less than 80) unique entries in the resultant LUT.

[0076] For angular modes, a joint symmetry over TU shape and intra prediction is considered. So, a mode i ($i > 34$) with TU shape AxB will be mapped to the same class corresponding to the mode $j=(68 - i)$ with TU shape BxA. However, for each transform pair the order of the horizontal and vertical transform kernel is swapped. For example, for a 16x4 block with mode 18 (horizontal prediction) and a 4x16 block with mode 50 (vertical prediction) are mapped to the same class. However, the vertical and horizontal transform kernels are swapped. For the wide-angle modes the nearest conventional angular mode is used for the transform set determination. For example, mode 2 is used for all the modes between -2 and -14. Similarly, mode 66 is used for mode 67 to mode 80.

[0077] ***LFNST extension with large kernel***

[0078] The LFNST design in VVC is extended as follows: The number of LFNST sets (S) and candidates (C) are extended to S=35 and C=3, and the LFNST set (lfnstTrSetIdx) for a given intra mode (predModeIntra) is derived according to the following formula: For predModeIntra < 2, lfnstTrSetIdx is equal to 2. lfnstTrSetIdx = predModeIntra, for predModeIntra in [0,34]. lfnstTrSetIdx = 68 – predModeIntra, for predModeIntra in [35,66]. Three different kernels, LFNST4, LFNST8, and LFNST16, are defined to indicate LFNST kernel sets, which are applied to 4xN/Nx4 ($N \geq 4$), 8xN/Nx8 ($N \geq 8$), and MxN ($M, N \geq 16$), respectively. The kernel dimensions are specified by: (LFSNT4, LFNST8*, LFNST16*) = (16x16, 32x64, 32x96). The forward LFNST is applied to top-left low frequency region, which is called Region-Of-Interest (ROI). When LFNST is applied, primary-transformed coefficients that exist in the region other than ROI are zeroed out, which is not changed from the VVC standard.

[0079] The ROI for LFNST16 is depicted in Figure 6. It consists of six 4x4 sub-blocks, which are consecutive in scan order. Since the number of input samples is 96, transform matrix for forward LFNST16 can be Rx96. R is chosen to be 32 in this contribution, 32 coefficients (two 4x4 sub-blocks) are generated from forward LFNST16 accordingly, which are placed following coefficient scan order.

[0080] The ROI for LFNST8 is shown in Figure 7. The forward LFNST8 matrix can be Rx64 and R is chosen to be 32. The generated coefficients are located in the same manner as with LFNST16.

[0081] The mapping from intra prediction modes to these sets is shown in Table 4.

Intra pred. mode	−14	−13	−12	−11	−10	−9	−8	−7	−6	−5	−4	−3	−2	−1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
IFNST set index	2	2	2	2	2	2	2	2	2	2	2	2	2	2	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Intra pred. mode	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49
INFST set index	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19
Intra pred. mode	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	
IFNST set index	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	

Table 4 Mapping of intra prediction modes to LFNST set index

[0082] ***Signal Dependent Transform (SDT) in JEM***

[0083] Considering that there are many similar patches within a frame and across frames, signal dependent transform explores such correlations can enhance coding performance by means of KLT. This trained KLT plays the role of a transform that is intended to compact the energy more efficiently.

[0084] The flowchart in Figure 8 describes this idea. For the current coding block indicated by C, at first, a reference patch R which consists of the reconstructed left-up template t_b and the prediction block p of the coding block is obtained. Then, this reference patch is used to search for N most similar patches over the reconstructed regions. Finally, one-dimensional KLT based on these blocks and prediction block is calculated. The coding block is unknown at the decoder for the collection of similar candidate blocks. The prediction block and the reconstructed template are used to guide the searching of similar blocks instead of using the original block. This tool is used for various block sizes 4×4, 8×8, 16×16 and 32×32.

[0085] It is known that Karhunen-Loéve transform (KLT) is the optimal transform in terms of the energy compaction efficiency. By searching over the reconstructed regions, N blocks $\boldsymbol{x_i}$, i = 1, 2, …, N, which are most similar to the reference patch are obtained. Here, $\boldsymbol{x_i} = (x_{i1}, x_{i2}, \ldots, x_{iD})^T$ and D indicates the vector dimension which is the transform block size. For an example, for 4×4 coding block, N is 16. The prediction p from those blocks is subtracted and obtain the residual blocks as $\boldsymbol{u_i}$, i = 1, 2, …, N, where $\boldsymbol{u_i} = (\boldsymbol{x_i} - \mathrm{p})/\sqrt{N}$. The, these residual blocks are used as the

training samples with zero mean for the KLT derivation. These N training samples can be represented by U = $(\boldsymbol{u_1}, \boldsymbol{u_2},\ldots,\boldsymbol{u_N})$, which is an D×N matrix. Consider the covariance matrix Σ as given by:

[0086] $\Sigma = \boldsymbol{U}\boldsymbol{U}^{\boldsymbol{T}}$ (3-4)

[0087] where the dimension of this covariance matrix is D×D. KLT bases are then the eigenvectors of this covariance matrix. For natural image/video contents, we find the selection of the candidate number N as 100 is enough for the good performance. The computation complexity for the eigenvalue decomposition is $O(D^3)$. For 4×4 block with D being 16, the complexity is $O(16^3)$, which is acceptable. For a large block, the complexity will be very high. For 32×32 block with D being 1024, the time complexity will be 262144 times slower than that for 4×4 block, being intolerable in the coding framework.

[0088] In considering this, a fast algorithm is used to make the large block size KLT feasible. The dimension of Σ is D×D. However, $\boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{U}$ has a much lower dimension as N× N. We calculate the eigenvectors $\boldsymbol{\phi}$ of $\Sigma' = \boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{U}$, which satisfy the equation as

[0089] $\boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{U}\boldsymbol{\phi} = \boldsymbol{\phi}\boldsymbol{\Lambda}$ (3-5),

[0090] where $\boldsymbol{\phi}$ indicates the eigenvector matrix while $\boldsymbol{\Lambda}$ denotes the diagonal matrix with the eigenvalues being the diagonal elements. Let's multiply both sides of Equation 33 by $\boldsymbol{U}$ to get

[0091] $\boldsymbol{U}\boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{U}\boldsymbol{\phi} = \boldsymbol{U}\boldsymbol{\phi}\boldsymbol{\Lambda}$ (3-6).

[0092] Add brackets to this equation and obtain

[0093] $(\boldsymbol{U}\boldsymbol{U}^{\boldsymbol{T}})(\boldsymbol{U}\boldsymbol{\phi}) = (\boldsymbol{U}\boldsymbol{\phi})\boldsymbol{\Lambda}$ (3-7)

[0094] The column vectors of $\boldsymbol{U}\boldsymbol{\phi}$ are the eigenvectors of $\boldsymbol{U}\boldsymbol{U}^{\boldsymbol{T}}$ with their corresponding eigenvalues being the diagonal elements of matrices $\boldsymbol{\Lambda}$. Let $\boldsymbol{\varphi} = \boldsymbol{U}\boldsymbol{\phi}$.

[0095] This indicates the eigenvectors of the high dimensional covariance matrix $\boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{U}$ can be obtained by multiplying $\boldsymbol{U}$ with the eigenvectors $\boldsymbol{\phi}$ which are obtained from the low dimensional covariance matrix $\boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{U}$. The dimensions of $\boldsymbol{\varphi}$ and $\boldsymbol{\Lambda}$ are both D×N. All the other (D−N) eigenvectors of $\boldsymbol{U}\boldsymbol{U}^{\boldsymbol{T}}$ have zero eigenvectors. We can use Schmidt orthogonalization to fill these (D−N) eigenvectors to get D×D eigenvector matrix. To reduce the complexity for matrix multiplication, one can use the obtained N eigenvectors to perform KLT transform, leaving the remaining (D−N) transform coefficients as zeros. This will not attenuate the performance since the first N projections can cover most of the signal energy while the bases are trained from samples being highly correlated with the coding block.

[0096] The described KLT is implemented at the block level on the coding block in the JEM. To have high adaptability to the image/video contents, the proposed scheme supports the proposed KLT on 4×4, 8×8, 16×16 and 32×32 coding blocks. At the JEM encoder side, rate-distortion optimization is used to determine the transform mode among the SDT and the adaptive multiple transform (AMT).

[0097] ***Extended Intra Prediction with Wide-angle Intra Modes***

[0098] Like HEVC, VVC uses a set of reference samples neighboring a current CU (i.e., above the current CU or left to the current CU) to predict samples of the current CU. However, to capture finer edge directions present in natural video (especially for video content in high resolutions, e.g., 4K), a number of angular intra modes is extended from 33 in HEVC to 93 in VVC. FIG. 9 illustrates a diagram of intra modes as defined in VVC and the ECM. As shown in FIG. 9, among the 93 angular intra modes, modes 2 to 66 are conventional angular intra modes, and modes -1 to -14 and modes 67 to 80 are wide-angle intra modes. In addition to the angular intra modes, the planar mode (mode 0 in FIG. 1) and Direct Current (DC) mode (mode 1 in FIG. 9) of HEVC are also applied in VVC.

[0099] Since a quad/binary/ternary tree partition structure is applied in VVC, besides video blocks in square shape, rectangular video blocks also exist for the intra prediction in VVC. Due to unequal width and height of one given video block, various sets of angular intra modes may be selected from the 93 angular intra modes for different block shapes. More specifically, for both square and rectangular video blocks, besides planar and DC modes, 65 angular intra modes among the 93 angular intra modes are also supported for each block shape. When a rectangular block shape of a video block satisfies a certain condition, an index of a wide-angle intra mode of the video block may be adaptively determined by the video decoder 30 according to an index of a conventional angular intra mode received from the video encoder 20 using a mapping relationship as shown in Table 5 below. That is, for non-square blocks, the wide-angle intra modes are signaled by the video encoder 20 using the indexes of the conventional angular intra modes, which are mapped to indexes of the wide-angle intra modes by the video decoder 30 after being parsed, thus ensuring that a total number (i.e., 67) of intra modes (i.e., the planar mode, the DC mode and 65 angular intra modes among the 93 angular intra modes) is unchanged, and the intra mode coding method is unchanged. As a result, a good efficiency of signaling intra modes is achieved while providing a consistent design across different block sizes.

[00100] Table 5 shows a mapping relationship between indexes of conventional angular intra modes and indexes of wide-angle intra modes for the intra prediction of different block shapes in VCC, wherein W represents a width of a video block, and H represents a height of the video block.

Block shape	**Aspect ratio**	**Indexes of conventional angular intra modes**	**Indexes of wide-angle intra modes**
Square, W = H	W/H == 1	None	None
Flat rectangle, W > H	W/H == 2	2,3,4,5,6,7,8,9	67, 68, 69, 70, 71, 72, 73, 74
	W/H == 4	2,3,4,5,6,7,8,9,10,11	67, 68, 69, 70, 71, 72, 73, 74, 75, 76
	W/H == 8	2,3,4,5,6,7,8,9,10,11,12, 13	67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78
	W/H == 16	2,3,4,5,6,7,8,9,10,11,12, 13,14,15	67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80
Tall rectangle, W < H	W/H == 1/2	59,60,61,62,63,64,65,66	-8, -7, -6, -5, -4, -3, -2, -1
	W/H == 1/4	57,58,59,60,61,62,63,64,65,66	-10, -9, -8, -7, -6, -5, -4, -3, -2, -1
	W/H == 1/8	55, 56,57,58,59,60,61,62,63,64,65,66	-12, -11, -10, -9, -8, -7, -6, -5, -4, -3, -2, -1
	W/H == 1/16	53, 54, 55, 56,57,58,59,60,61,62,63,64,65,66	-14, -13, -12, -11, -10, -9, -8, -7, -6, -5, -4, -3, -2, -1

Table 5

[00101] FIG. 1 is a block diagram illustrating an exemplary system 10 for encoding and decoding video blocks in parallel in accordance with some implementations of the present disclosure. As shown in FIG. 1, the system 10 includes a source device 12 that generates and encodes video data to be decoded at a later time by a destination device 14. The source device 12 and the destination device 14 may comprise any of a wide variety of electronic devices, including

cloud servers, server computers, desktop or laptop computers, tablet computers, smart phones, set-top boxes, digital televisions, cameras, display devices, digital media players, video gaming consoles, video streaming device, or the like. In some implementations, the source device 12 and the destination device 14 are equipped with wireless communication capabilities.

[00102] In some implementations, the destination device 14 may receive the encoded video data to be decoded via a link 16. The link 16 may comprise any type of communication medium or device capable of moving the encoded video data from the source device 12 to the destination device 14. In one example, the link 16 may comprise a communication medium to enable the source device 12 to transmit the encoded video data directly to the destination device 14 in real time. The encoded video data may be modulated according to a communication standard, such as a wireless communication protocol, and transmitted to the destination device 14. The communication medium may comprise any 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 any other equipment that may be useful to facilitate communication from the source device 12 to the destination device 14.

[00103] In some other implementations, the encoded video data may be transmitted from an output interface 22 to a storage device 32. Subsequently, the encoded video data in the storage device 32 may be accessed by the destination device 14 via an input interface 28. The storage device 32 may include any of a variety of distributed or locally accessed data storage media such as a hard drive, Blu-ray discs, Digital Versatile Disks (DVDs), Compact Disc Read-Only Memories (CD-ROMs), flash memory, volatile or non-volatile memory, or any other suitable digital storage media for storing the encoded video data. In a further example, the storage device 32 may correspond to a file server or another intermediate storage device that may hold the encoded video data generated by the source device 12. The destination device 14 may access the stored video data from the storage device 32 via streaming or downloading. The file server may be any type of computer capable of storing the encoded video data and transmitting the encoded video data to the destination device 14. Exemplary file servers include a web server (e.g., for a website), a File Transfer Protocol (FTP) server, Network Attached Storage (NAS) devices, or a local disk drive. The destination device 14 may access the encoded video data through any standard

data connection, including a wireless channel (e.g., a Wireless Fidelity (Wi-Fi) connection), a wired connection (e.g., Digital Subscriber Line (DSL), cable modem, etc.), or a combination of both that is suitable for accessing encoded video data stored on a file server. The transmission of the encoded video data from the storage device 32 may be a streaming transmission, a download transmission, or a combination of both.

[00104] As shown in FIG. 1, the source device 12 includes a video source 18, a video encoder 20 and the output interface 22. The video source 18 may include a source such as a video capturing device, e.g., a video camera, a video archive containing previously captured video, a video feeding interface to receive video from a video content provider, and/or a computer graphics system for generating computer graphics data as the source video, or a combination of such sources. As one example, if the video source 18 is a video camera of a security surveillance system, the source device 12 and the destination device 14 may form camera phones or video phones. However, the implementations described in the present application may be applicable to video coding in general, and may be applied to wireless and/or wired applications.

[00105] The captured, pre-captured, or computer-generated video may be encoded by the video encoder 20. The encoded video data may be transmitted directly to the destination device 14 via the output interface 22 of the source device 12. The encoded video data may also (or alternatively) be stored onto the storage device 32 for later access by the destination device 14 or other devices, for decoding and/or playback. The output interface 22 may further include a modem and/or a transmitter.

[00106] The destination device 14 includes the input interface 28, a video decoder 30, and a display device 34. The input interface 28 may include a receiver and/or a modem and receive the encoded video data over the link 16. The encoded video data communicated over the link 16, or provided on the storage device 32, may include a variety of syntax elements generated by the video encoder 20 for use by the video decoder 30 in decoding the video data. Such syntax elements may be included within the encoded video data transmitted on a communication medium, stored on a storage medium, or stored on a file server.

[00107] In some implementations, the destination device 14 may include the display device 34, which can be an integrated display device and an external display device that is configured to communicate with the destination device 14. The display device 34 displays the decoded video data to a user, and may comprise any of a variety of display devices such as a Liquid Crystal

Display (LCD), a plasma display, an Organic Light Emitting Diode (OLED) display, or another type of display device.

[00108] The video encoder 20 and the video decoder 30 may operate according to proprietary or industry standards, such as VVC, HEVC, MPEG-4, Part 10, AVC, or extensions of such standards. It should be understood that the present application is not limited to a specific video encoding/decoding standard and may be applicable to other video encoding/decoding standards. It is generally contemplated that the video encoder 20 of the source device 12 may be configured to encode video data according to any of these current or future standards. Similarly, it is also generally contemplated that the video decoder 30 of the destination device 14 may be configured to decode video data according to any of these current or future standards.

[00109] The video encoder 20 and the video decoder 30 each may be implemented as any of a variety of suitable encoder and/or decoder circuitry, such as one or more microprocessors, Digital Signal Processors (DSPs), Application Specific Integrated Circuits (ASICs), Field Programmable Gate Arrays (FPGAs), discrete logic, software, hardware, firmware or any combinations thereof. When implemented partially in software, an electronic device may store instructions for the software in a suitable, non-transitory computer-readable medium and execute the instructions in hardware using one or more processors to perform the video encoding/decoding operations disclosed in the present disclosure. Each of the video encoder 20 and the 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.

[00110] In some implementations, at least a part of components of the source device 12 (for example, the video source 18, the video encoder 20 or components included in the video encoder 20 as described below with reference to Fig. 2, and the output interface 22) and/or at least a part of components of the destination device 14 (for example, the input interface 28, the video decoder 30 or components included in the video decoder 30 as described below with reference to Fig. 3, and the display device 34) may operate in a cloud computing service network which may provide software, platforms, and/or infrastructure, such as Software as a Service (SaaS), Platform as a Service (PaaS), or Infrastructure as a Service (IaaS). In some implementations, one or more components in the source device 12 and/or the destination device 14 which are not included in the cloud computing service network may be provided in one or more client devices, and the one or more client devices may communicate with server computers in the cloud computing service

network through a wireless communication network (for example, a cellular communication network, a short-range wireless communication network, or a global navigation satellite system (GNSS) communication network) or a wired communication network (e.g., a local area network (LAN) communication network or a power line communication (PLC) network). In an embodiment, at least a part of operations described herein may be implemented as cloud-based services provided by one or more server computers which are implemented by the at least a part of the components of the source device 12 and/or the at least a part of the components of the destination device 14 in the cloud computing service network; and one or more other operations described herein may be implemented by the one or more client devices. In some implementations, the cloud computing service network may be a private cloud, a public cloud, or a hybrid cloud. The terms such as "cloud," "cloud computing," "cloud-based" etc. herein may be used interchangeably as appropriate without departing from the scope of the present disclosure. It should be understood that the present disclosure is not limited to being implemented in the cloud computing service network described above. Instead, the present disclosure may also be implemented in any other type of computing environments currently known or developed in the future.

[00111] FIG. 2 is a block diagram illustrating an exemplary video encoder 20 in accordance with some implementations described in the present application. The video encoder 20 may perform intra and inter predictive coding of video blocks within video frames. Intra predictive coding relies on spatial prediction to reduce or remove spatial redundancy in video data within a given video frame or picture. Inter predictive coding relies on temporal prediction to reduce or remove temporal redundancy in video data within adjacent video frames or pictures of a video sequence. It should be noted that the term "frame" may be used as synonyms for the term "image" or "picture" in the field of video coding.

[00112] As shown in FIG. 2, the video encoder 20 includes a video data memory 40, a prediction processing unit 41, a Decoded Picture Buffer (DPB) 64, a summer 50, a transform processing unit 52, a quantization unit 54, and an entropy encoding unit 56. The prediction processing unit 41 further includes a motion estimation unit 42, a motion compensation unit 44, a partition unit 45, an intra prediction processing unit 46, and an intra Block Copy (BC) unit 48. In some implementations, the video encoder 20 also includes an inverse quantization unit 58, an inverse transform processing unit 60, and a summer 62 for video block reconstruction. An in-loop filter

63, such as a deblocking filter, may be positioned between the summer 62 and the DPB 64 to filter block boundaries to remove blockiness artifacts from reconstructed video. Another in-loop filter, such as Sample Adaptive Offset (SAO) filter, Cross Component Sample Adaptive Offset (CCSAO) filter and/or Adaptive in-Loop Filter (ALF), may also be used in addition to the deblocking filter to filter an output of the summer 62. It should be illustrated that for the CCSAO technique, the present application is not limited to the embodiments described herein, and instead, the application may be applied to a situation where an offset is selected for any of a luma component, a Cb chroma component and a Cr chroma component according to any other of the luma component, the Cb chroma component and the Cr chroma component to modify said any component based on the selected offset. Further, it should also be illustrated that a first component mentioned herein may be any of the luma component, the Cb chroma component and the Cr chroma component, a second component mentioned herein may be any other of the luma component, the Cb chroma component and the Cr chroma component, and a third component mentioned herein may be a remaining one of the luma component, the Cb chroma component and the Cr chroma component. In some examples, the in-loop filters may be omitted, and the decoded video block may be directly provided by the summer 62 to the DPB 64. The video encoder 20 may take the form of a fixed or programmable hardware unit or may be divided among one or more of the illustrated fixed or programmable hardware units.

[00113] The video data memory 40 may store video data to be encoded by the components of the video encoder 20. The video data in the video data memory 40 may be obtained, for example, from the video source 18 as shown in FIG. 1. The DPB 64 is a buffer that stores reference video data (for example, reference frames or pictures) for use in encoding video data by the video encoder 20 (e.g., in intra or inter predictive coding modes). The video data memory 40 and the DPB 64 may be formed by any of a variety of memory devices. In various examples, the video data memory 40 may be on-chip with other components of the video encoder 20, or off-chip relative to those components.

[00114] As shown in FIG. 2, after receiving the video data, the partition unit 45 within the prediction processing unit 41 partitions the video data into video blocks. This partitioning may also include partitioning a video frame into slices, tiles (for example, sets of video blocks), or other larger Coding Units (CUs) according to predefined splitting structures such as a Quad-Tree (QT) structure associated with the video data. The video frame is or may be regarded as a two-

dimensional array or matrix of samples with sample values. A sample in the array may also be referred to as a pixel or a pel. A number of samples in horizontal and vertical directions (or axes) of the array or picture define a size and/or a resolution of the video frame. The video frame may be divided into multiple video blocks by, for example, using QT partitioning.

[00115] The video block again is or may be regarded as a two-dimensional array or matrix of samples with sample values, although of smaller dimension than the video frame. A number of samples in horizontal and vertical directions (or axes) of the video block define a size of the video block. The video block may further be partitioned into one or more block partitions or sub-blocks (which may form again blocks) by, for example, iteratively using QT partitioning, Binary-Tree (BT) partitioning or Triple-Tree (TT) partitioning or any combination thereof. It should be noted that the term "block" or "video block" as used herein may be a portion, in particular a rectangular (square or non- square) portion, of a frame or a picture. With reference, for example, to HEVC and VVC, the block or video block may be or correspond to a Coding Tree Unit (CTU), a CU, a Prediction Unit (PU) or a Transform Unit (TU) and/or may be or correspond to a corresponding block, e.g. a Coding Tree Block (CTB), a Coding Block (CB), a Prediction Block (PB) or a Transform Block (TB) and/or to a sub-block.

[00116] The prediction processing unit 41 may select one of a plurality of possible predictive coding modes, such as one of a plurality of intra predictive coding modes or one of a plurality of inter predictive coding modes, for the current video block based on error results (e.g., coding rate and the level of distortion). The prediction processing unit 41 may provide the resulting intra or inter prediction coded block to the summer 50 to generate a residual block and to the summer 62 to reconstruct the encoded block for use as part of a reference frame subsequently. The prediction processing unit 41 also provides syntax elements, such as motion vectors, intra-mode indicators, partition information, and other such syntax information, to the entropy encoding unit 56.

[00117] In order to select an appropriate intra predictive coding mode for the current video block, the intra prediction processing unit 46 within the prediction processing unit 41 may perform intra predictive coding of the current video block relative to one or more neighbor blocks in the same frame as the current block to be coded to provide spatial prediction. The motion estimation unit 42 and the motion compensation unit 44 within the prediction processing unit 41 perform inter predictive coding of the current video block relative to one or more predictive blocks in one or

more reference frames to provide temporal prediction. The video encoder 20 may perform multiple coding passes, e.g., to select an appropriate coding mode for each block of video data.

[00118] In some implementations, the motion estimation unit 42 determines the inter prediction mode for a current video frame by generating a motion vector, which indicates the displacement of a video block within the current video frame relative to a predictive block within a reference video frame, according to a predetermined pattern within a sequence of video frames. Motion estimation, performed by the motion estimation unit 42, is the process of generating motion vectors, which estimate motion for video blocks.

[00119] A motion vector, for example, may indicate the displacement of a video block within a current video frame or picture relative to a predictive block within a reference frame relative to the current block being coded within the current frame. The predetermined pattern may designate video frames in the sequence as P frames or B frames. The intra BC unit 48 may determine vectors, e.g., block vectors, for intra BC coding in a manner similar to the determination of motion vectors by the motion estimation unit 42 for inter prediction, or may utilize the motion estimation unit 42 to determine the block vector.

[00120] A predictive block for the video block may be or may correspond to a block or a reference block of a reference frame that is deemed as closely matching the video block to be coded in terms of pixel difference, which may be determined by Sum of Absolute Difference (SAD), Sum of Square Difference (SSD), or other difference metrics. In some implementations, the video encoder 20 may calculate values for sub-integer pixel positions of reference frames stored in the DPB 64. For example, the video encoder 20 may interpolate values of one-quarter pixel positions, one-eighth pixel positions, or other fractional pixel positions of the reference frame. Therefore, the motion estimation unit 42 may perform a motion search relative to the full pixel positions and fractional pixel positions and output a motion vector with fractional pixel precision.

[00121] The motion estimation unit 42 calculates a motion vector for a video block in an inter prediction coded frame by comparing the position of the video block to the position of a predictive block of a reference frame selected from a first reference frame list (List 0) or a second reference frame list (List 1), each of which identifies one or more reference frames stored in the DPB 64. The motion estimation unit 42 sends the calculated motion vector to the motion compensation unit 44 and then to the entropy encoding unit 56.

[00122] Motion compensation, performed by the motion compensation unit 44, may involve fetching or generating the predictive block based on the motion vector determined by the motion estimation unit 42. Upon receiving the motion vector for the current video block, the motion compensation unit 44 may locate a predictive block to which the motion vector points in one of the reference frame lists, retrieve the predictive block from the DPB 64, and forward the predictive block to the summer 50. The summer 50 then forms a residual video block of pixel difference values by subtracting pixel values of the predictive block provided by the motion compensation unit 44 from the pixel values of the current video block being coded.

[00123] The pixel difference values forming the residual video block may include luma or chroma component differences or both. The motion compensation unit 44 may also generate syntax elements associated with the video blocks of a video frame for use by the video decoder 30 in decoding the video blocks of the video frame. The syntax elements may include, for example, syntax elements defining the motion vector used to identify the predictive block, any flags indicating the prediction mode, or any other syntax information described herein. Note that the motion estimation unit 42 and the motion compensation unit 44 may be highly integrated, but are illustrated separately for conceptual purposes.

[00124] In some implementations, the intra BC unit 48 may generate vectors and fetch predictive blocks in a manner similar to that described above in connection with the motion estimation unit 42 and the motion compensation unit 44, but with the predictive blocks being in the same frame as the current block being coded and with the vectors being referred to as block vectors as opposed to motion vectors. In particular, the intra BC unit 48 may determine an intra-prediction mode to use to encode a current block. In some examples, the intra BC unit 48 may encode a current block using various intra-prediction modes, e.g., during separate encoding passes, and test their performance through rate-distortion analysis. Next, the intra BC unit 48 may select, among the various tested intra-prediction modes, an appropriate intra-prediction mode to use and generate an intra-mode indicator accordingly. For example, the intra BC unit 48 may calculate rate-distortion values using a rate-distortion analysis for the various tested intra-prediction modes, and select the intra-prediction mode having the best rate-distortion characteristics among the tested modes as the appropriate intra-prediction mode to use.

[00125] Rate-distortion analysis generally determines an amount of distortion (or error) between an encoded block and an original, unencoded block that was encoded to produce the

encoded block, as well as a bitrate (i.e., a number of bits) used to produce the encoded block. Intra BC unit 48 may calculate ratios from the distortions and rates for the various encoded blocks to determine which intra-prediction mode exhibits the best rate-distortion value for the block.

[00126] In other examples, the intra BC unit 48 may use the motion estimation unit 42 and the motion compensation unit 44, in whole or in part, to perform such functions for Intra BC prediction according to the implementations described herein. In either case, for Intra block copy, a predictive block may be a block that is deemed as closely matching the block to be coded, in terms of pixel difference, which may be determined by SAD, SSD, or other difference metrics, and identification of the predictive block may include calculation of values for sub-integer pixel positions.

[00127] Whether the predictive block is from the same frame according to intra prediction, or a different frame according to inter prediction, the video encoder 20 may form a residual video block by subtracting pixel values of the predictive block from the pixel values of the current video block being coded, forming pixel difference values. The pixel difference values forming the residual video block may include both luma and chroma component differences.

[00128] The intra prediction processing unit 46 may intra-predict a current video block, as an alternative to the inter-prediction performed by the motion estimation unit 42 and the motion compensation unit 44, or the intra block copy prediction performed by the intra BC unit 48, as described above. In particular, the intra prediction processing unit 46 may determine an intra prediction mode to use to encode a current block. To do so, the intra prediction processing unit 46 may encode a current block using various intra prediction modes, e.g., during separate encoding passes, and the intra prediction processing unit 46 (or a mode selection unit, in some examples) may select an appropriate intra prediction mode to use from the tested intra prediction modes. The intra prediction processing unit 46 may provide information indicative of the selected intra-prediction mode for the block to the entropy encoding unit 56. The entropy encoding unit 56 may encode the information indicating the selected intra-prediction mode in the bitstream.

[00129] After the prediction processing unit 41 determines the predictive block for the current video block via either inter prediction or intra prediction, the summer 50 forms a residual video block by subtracting the predictive block from the current video block. The residual video data in the residual block may be included in one or more TUs and is provided to the transform processing unit 52. The transform processing unit 52 transforms the residual video data into residual transform

coefficients using a transform, such as a Discrete Cosine Transform (DCT) or a conceptually similar transform.

[00130] The transform processing unit 52 may send the resulting transform coefficients to the quantization unit 54. The quantization unit 54 quantizes the transform coefficients to further reduce the bit rate. The quantization process may also reduce the bit depth associated with some or all of the coefficients. The degree of quantization may be modified by adjusting a quantization parameter. In some examples, the quantization unit 54 may then perform a scan of a matrix including the quantized transform coefficients. Alternatively, the entropy encoding unit 56 may perform the scan.

[00131] Following quantization, the entropy encoding unit 56 entropy encodes the quantized transform coefficients into a video bitstream using, e.g., Context Adaptive Variable Length Coding (CAVLC), Context Adaptive Binary Arithmetic Coding (CABAC), Syntax-based context-adaptive Binary Arithmetic Coding (SBAC), Probability Interval Partitioning Entropy (PIPE) coding or another entropy encoding methodology or technique. The encoded bitstream may then be transmitted to the video decoder 30 as shown in FIG. 1, or archived in the storage device 32 as shown in FIG. 1 for later transmission to or retrieval by the video decoder 30. The entropy encoding unit 56 may also entropy encode the motion vectors and the other syntax elements for the current video frame being coded.

[00132] The inverse quantization unit 58 and the inverse transform processing unit 60 apply inverse quantization and inverse transformation, respectively, to reconstruct the residual video block in the pixel domain for generating a reference block for prediction of other video blocks. As noted above, the motion compensation unit 44 may generate a motion compensated predictive block from one or more reference blocks of the frames stored in the DPB 64. The motion compensation unit 44 may also apply one or more interpolation filters to the predictive block to calculate sub-integer pixel values for use in motion estimation.

[00133] The summer 62 adds the reconstructed residual block to the motion compensated predictive block produced by the motion compensation unit 44 to produce a reference block for storage in the DPB 64. The reference block may then be used by the intra BC unit 48, the motion estimation unit 42 and the motion compensation unit 44 as a predictive block to inter predict another video block in a subsequent video frame.

[00134] FIG. 3A is a block diagram illustrating an exemplary video decoder 30 in accordance with some implementations of the present application. The video decoder 30 includes a video data memory 79, an entropy decoding unit 80, a prediction processing unit 81, an inverse quantization unit 86, an inverse transform processing unit 88, a summer 90, and a DPB 92. The prediction processing unit 81 further includes a motion compensation unit 82, an intra prediction unit 84, and an intra BC unit 85. The video decoder 30 may perform a decoding process generally reciprocal to the encoding process described above with respect to the video encoder 20 in connection with FIG. 2. For example, the motion compensation unit 82 may generate prediction data based on motion vectors received from the entropy decoding unit 80, while the intra-prediction unit 84 may generate prediction data based on intra-prediction mode indicators received from the entropy decoding unit 80. FIG. 3B is a block diagram illustrating a general video decoder for an AVS3.

[00135] In some examples, a unit of the video decoder 30 may be tasked to perform the implementations of the present application. Also, in some examples, the implementations of the present disclosure may be divided among one or more of the units of the video decoder 30. For example, the intra BC unit 85 may perform the implementations of the present application, alone, or in combination with other units of the video decoder 30, such as the motion compensation unit 82, the intra prediction unit 84, and the entropy decoding unit 80. In some examples, the video decoder 30 may not include the intra BC unit 85 and the functionality of intra BC unit 85 may be performed by other components of the prediction processing unit 81, such as the motion compensation unit 82.

[00136] The video data memory 79 may store video data, such as an encoded video bitstream, to be decoded by the other components of the video decoder 30. The video data stored in the video data memory 79 may be obtained, for example, from the storage device 32, from a local video source, such as a camera, via wired or wireless network communication of video data, or by accessing physical data storage media (e.g., a flash drive or hard disk). The video data memory 79 may include a Coded Picture Buffer (CPB) that stores encoded video data from an encoded video bitstream. The DPB 92 of the video decoder 30 stores reference video data for use in decoding video data by the video decoder 30 (e.g., in intra or inter predictive coding modes). The video data memory 79 and the DPB 92 may be formed by any of a variety of memory devices, such as dynamic random access memory (DRAM), including Synchronous DRAM (SDRAM), Magneto-resistive RAM (MRAM), Resistive RAM (RRAM), or other types of memory devices. For

illustrative purpose, the video data memory 79 and the DPB 92 are depicted as two distinct components of the video decoder 30 in FIG. 3A or 3B. But it will be apparent to one skilled in the art that the video data memory 79 and the DPB 92 may be provided by the same memory device or separate memory devices. In some examples, the video data memory 79 may be on-chip with other components of the video decoder 30, or off-chip relative to those components.

[00137] During the decoding process, the video decoder 30 receives an encoded video bitstream that represents video blocks of an encoded video frame and associated syntax elements. The video decoder 30 may receive the syntax elements at the video frame level and/or the video block level. The entropy decoding unit 80 of the video decoder 30 entropy decodes the bitstream to generate quantized coefficients, motion vectors or intra-prediction mode indicators, and other syntax elements. The entropy decoding unit 80 then forwards the motion vectors or intra-prediction mode indicators and other syntax elements to the prediction processing unit 81.

[00138] When the video frame is coded as an intra predictive coded (I) frame or for intra coded predictive blocks in other types of frames, the intra prediction unit 84 of the prediction processing unit 81 may generate prediction data for a video block of the current video frame based on a signaled intra prediction mode and reference data from previously decoded blocks of the current frame.

[00139] When the video frame is coded as an inter-predictive coded (i.e., B or P) frame, the motion compensation unit 82 of the prediction processing unit 81 produces one or more predictive blocks for a video block of the current video frame based on the motion vectors and other syntax elements received from the entropy decoding unit 80. Each of the predictive blocks may be produced from a reference frame within one of the reference frame lists. The video decoder 30 may construct the reference frame lists, List 0 and List 1, using default construction techniques based on reference frames stored in the DPB 92.

[00140] In some examples, when the video block is coded according to the intra BC mode described herein, the intra BC unit 85 of the prediction processing unit 81 produces predictive blocks for the current video block based on block vectors and other syntax elements received from the entropy decoding unit 80. The predictive blocks may be within a reconstructed region of the same picture as the current video block defined by the video encoder 20.

[00141] The motion compensation unit 82 and/or the intra BC unit 85 determines prediction information for a video block of the current video frame by parsing the motion vectors and other

syntax elements, and then uses the prediction information to produce the predictive blocks for the current video block being decoded. For example, the motion compensation unit 82 uses some of the received syntax elements to determine a prediction mode (e.g., intra or inter prediction) used to code video blocks of the video frame, an inter prediction frame type (e.g., B or P), construction information for one or more of the reference frame lists for the frame, motion vectors for each inter predictive encoded video block of the frame, inter prediction status for each inter predictive coded video block of the frame, and other information to decode the video blocks in the current video frame.

[00142] Similarly, the intra BC unit 85 may use some of the received syntax elements, e.g., a flag, to determine that the current video block was predicted using the intra BC mode, construction information of which video blocks of the frame are within the reconstructed region and should be stored in the DPB 92, block vectors for each intra BC predicted video block of the frame, intra BC prediction status for each intra BC predicted video block of the frame, and other information to decode the video blocks in the current video frame.

[00143] The motion compensation unit 82 may also perform interpolation using the interpolation filters as used by the video encoder 20 during encoding of the video blocks to calculate interpolated values for sub-integer pixels of reference blocks. In this case, the motion compensation unit 82 may determine the interpolation filters used by the video encoder 20 from the received syntax elements and use the interpolation filters to produce predictive blocks.

[00144] The inverse quantization unit 86 inverse quantizes the quantized transform coefficients provided in the bitstream and entropy decoded by the entropy decoding unit 80 using the same quantization parameter calculated by the video encoder 20 for each video block in the video frame to determine a degree of quantization. The inverse transform processing unit 88 applies an inverse transform, e.g., an inverse DCT, an inverse integer transform, or a conceptually similar inverse transform process, to the transform coefficients in order to reconstruct the residual blocks in the pixel domain.

[00145] After the motion compensation unit 82 or the intra BC unit 85 generates the predictive block for the current video block based on the vectors and other syntax elements, the summer 90 reconstructs decoded video block for the current video block by summing the residual block from the inverse transform processing unit 88 and a corresponding predictive block generated by the motion compensation unit 82 and the intra BC unit 85. An in-loop filter 91 such as deblocking

filter, SAO filter, CCSAO filter and/or ALF may be positioned between the summer 90 and the DPB 92 to further process the decoded video block. In some examples, the in-loop filter 91 may be omitted, and the decoded video block may be directly provided by the summer 90 to the DPB 92. The decoded video blocks in a given frame are then stored in the DPB 92, which stores reference frames used for subsequent motion compensation of next video blocks. The DPB 92, or a memory device separate from the DPB 92, may also store decoded video for later presentation on a display device, such as the display device 34 of FIG. 1.

[00146] In a typical video coding process, a video sequence typically includes an ordered set of frames or pictures. Each frame may include three sample arrays, denoted SL, SCb, and SCr. SL is a two-dimensional array of luma samples. SCb is a two-dimensional array of Cb chroma samples. SCr is a two-dimensional array of Cr chroma samples. In other instances, a frame may be monochrome and therefore includes only one two-dimensional array of luma samples. FIG. 3C is an illustration of an LFNST process according to one example of the present disclosure.

[00147] As shown in FIG. 4A, the video encoder 20 (or more specifically the partition unit 45) generates an encoded representation of a frame by first partitioning the frame into a set of CTUs. A video frame may include an integer number of CTUs ordered consecutively in a raster scan order from left to right and from top to bottom. Each CTU is a largest logical coding unit and the width and height of the CTU are signaled by the video encoder 20 in a sequence parameter set, such that all the CTUs in a video sequence have the same size being one of 128×128, 64×64, 32×32, and 16×16. But it should be noted that the present application is not necessarily limited to a particular size.

[00148] As shown in FIG. 4B, each CTU may comprise one CTB of luma samples, two corresponding coding tree blocks of chroma samples, and syntax elements used to code the samples of the coding tree blocks. The syntax elements describe properties of different types of units of a coded block of pixels and how the video sequence can be reconstructed at the video decoder 30, including inter or intra prediction, intra prediction mode, motion vectors, and other parameters. In monochrome pictures or pictures having three separate color planes, a CTU may comprise a single coding tree block and syntax elements used to code the samples of the coding tree block. A coding tree block may be an NxN block of samples.

[00149] To achieve a better performance, the video encoder 20 may recursively perform tree partitioning such as binary-tree partitioning, ternary-tree partitioning, quad-tree partitioning or a

combination thereof on the coding tree blocks of the CTU and divide the CTU into smaller CUs. As depicted in FIG. 4C, the 64x64 CTU 400 is first divided into four smaller CUs, each having a block size of 32x32. Among the four smaller CUs, CU 410 and CU 420 are each divided into four CUs of 16x16 by block size. The two 16x16 CUs 430 and 440 are each further divided into four CUs of 8x8 by block size.

[00150] FIG. 4D depicts a quad-tree data structure illustrating the end result of the partition process of the CTU 400 as depicted in FIG. 4C, each leaf node of the quad-tree corresponding to one CU of a respective size ranging from 32x32 to 8x8. Like the CTU depicted in FIG. 4B, each CU may comprise a CB of luma samples and two corresponding coding blocks of chroma samples of a frame of the same size, and syntax elements used to code the samples of the coding blocks. In monochrome pictures or pictures having three separate color planes, a CU may comprise a single coding block and syntax structures used to code the samples of the coding block. It should be noted that the quad-tree partitioning depicted in FIGS. 4C and 4D is only for illustrative purposes and one CTU can be split into CUs to adapt to varying local characteristics based on quad/ternary/binary-tree partitions.

[00151] In the multi-type tree structure, one CTU is partitioned by a quad-tree structure and each quad-tree leaf CU can be further partitioned by a binary and ternary tree structure. As shown in FIG. 4E, there are five possible partitioning types of a coding block having a width W and a height H, i.e., quaternary partitioning, horizontal binary partitioning, vertical binary partitioning, horizontal ternary partitioning, and vertical ternary partitioning.

[00152] In some implementations, the video encoder 20 may further partition a coding block of a CU into one or more MxN PBs. A PB is a rectangular (square or non-square) block of samples on which the same prediction, inter or intra, is applied. A PU of a CU may comprise a PB of luma samples, two corresponding PBs of chroma samples, and syntax elements used to predict the PBs. In monochrome pictures or pictures having three separate color planes, a PU may comprise a single PB and syntax structures used to predict the PB. The video encoder 20 may generate predictive luma, Cb, and Cr blocks for luma, Cb, and Cr PBs of each PU of the CU.

[00153] The video encoder 20 may use intra prediction or inter prediction to generate the predictive blocks for a PU. If the video encoder 20 uses intra prediction to generate the predictive blocks of a PU, the video encoder 20 may generate the predictive blocks of the PU based on decoded samples of the frame associated with the PU. If the video encoder 20 uses inter prediction

to generate the predictive blocks of a PU, the video encoder 20 may generate the predictive blocks of the PU based on decoded samples of one or more frames other than the frame associated with the PU.

[00154] After the video encoder 20 generates predictive luma, Cb, and Cr blocks for one or more PUs of a CU, the video encoder 20 may generate a luma residual block for the CU by subtracting the CU's predictive luma blocks from its original luma coding block such that each sample in the CU's luma residual block indicates a difference between a luma sample in one of the CU's predictive luma blocks and a corresponding sample in the CU's original luma coding block. Similarly, the video encoder 20 may generate a Cb residual block and a Cr residual block for the CU, respectively, such that each sample in the CU's Cb residual block indicates a difference between a Cb sample in one of the CU's predictive Cb blocks and a corresponding sample in the CU's original Cb coding block and each sample in the CU's Cr residual block may indicate a difference between a Cr sample in one of the CU's predictive Cr blocks and a corresponding sample in the CU's original Cr coding block.

[00155] Furthermore, as illustrated in FIG. 4C, the video encoder 20 may use quad-tree partitioning to decompose the luma, Cb, and Cr residual blocks of a CU into one or more luma, Cb, and Cr transform blocks respectively. A transform block is a rectangular (square or non-square) block of samples on which the same transform is applied. A TU of a CU may comprise a transform block of luma samples, two corresponding transform blocks of chroma samples, and syntax elements used to transform the transform block samples. Thus, each TU of a CU may be associated with a luma transform block, a Cb transform block, and a Cr transform block. In some examples, the luma transform block associated with the TU may be a sub-block of the CU's luma residual block. The Cb transform block may be a sub-block of the CU's Cb residual block. The Cr transform block may be a sub-block of the CU's Cr residual block. In monochrome pictures or pictures having three separate color planes, a TU may comprise a single transform block and syntax structures used to transform the samples of the transform block.

[00156] The video encoder 20 may apply one or more transforms to a luma transform block of a TU to generate a luma coefficient block for the TU. A coefficient block may be a two-dimensional array of transform coefficients. A transform coefficient may be a scalar quantity. The video encoder 20 may apply one or more transforms to a Cb transform block of a TU to generate

a Cb coefficient block for the TU. The video encoder 20 may apply one or more transforms to a Cr transform block of a TU to generate a Cr coefficient block for the TU.

[00157] After generating a coefficient block (e.g., a luma coefficient block, a Cb coefficient block or a Cr coefficient block), the video encoder 20 may quantize the coefficient block. Quantization generally refers to a process in which transform coefficients are quantized to possibly reduce the amount of data used to represent the transform coefficients, providing further compression. After the video encoder 20 quantizes a coefficient block, the video encoder 20 may entropy encode syntax elements indicating the quantized transform coefficients. For example, the video encoder 20 may perform CABAC on the syntax elements indicating the quantized transform coefficients. Finally, the video encoder 20 may output a bitstream that includes a sequence of bits that forms a representation of coded frames and associated data, which is either saved in the storage device 32 or transmitted to the destination device 14.

[00158] After receiving a bitstream generated by the video encoder 20, the video decoder 30 may parse the bitstream to obtain syntax elements from the bitstream. The video decoder 30 may reconstruct the frames of the video data based at least in part on the syntax elements obtained from the bitstream. The process of reconstructing the video data is generally reciprocal to the encoding process performed by the video encoder 20. For example, the video decoder 30 may perform inverse transforms on the coefficient blocks associated with TUs of a current CU to reconstruct residual blocks associated with the TUs of the current CU. The video decoder 30 also reconstructs the coding blocks of the current CU by adding the samples of the predictive blocks for PUs of the current CU to corresponding samples of the transform blocks of the TUs of the current CU. After reconstructing the coding blocks for each CU of a frame, video decoder 30 may reconstruct the frame.

[00159] As noted above, video coding achieves video compression using primarily two modes, i.e., intra-frame prediction (or intra-prediction) and inter-frame prediction (or inter-prediction). It is noted that IBC could be regarded as either intra-frame prediction or a third mode. Between the two modes, inter-frame prediction contributes more to the coding efficiency than intra-frame prediction because of the use of motion vectors for predicting a current video block from a reference video block.

[00160] But with the ever improving video data capturing technology and more refined video block size for preserving details in the video data, the amount of data required for representing

motion vectors for a current frame also increases substantially. One way of overcoming this challenge is to benefit from the fact that not only a group of neighboring CUs in both the spatial and temporal domains have similar video data for predicting purpose but the motion vectors between these neighboring CUs are also similar. Therefore, it is possible to use the motion information of spatially neighboring CUs and/or temporally co-located CUs as an approximation of the motion information (e.g., motion vector) of a current CU by exploring their spatial and temporal correlation, which is also referred to as "Motion Vector Predictor (MVP)" of the current CU.

[00161] Instead of encoding, into the video bitstream, an actual motion vector of the current CU determined by the motion estimation unit 42 as described above in connection with FIG. 2, the motion vector predictor of the current CU is subtracted from the actual motion vector of the current CU to produce a Motion Vector Difference (MVD) for the current CU. By doing so, there is no need to encode the motion vector determined by the motion estimation unit 42 for each CU of a frame into the video bitstream and the amount of data used for representing motion information in the video bitstream can be significantly decreased.

[00162] Like the process of choosing a predictive block in a reference frame during inter-frame prediction of a code block, a set of rules need to be adopted by both the video encoder 20 and the video decoder 30 for constructing a motion vector candidate list (also known as a "merge list") for a current CU using those potential candidate motion vectors associated with spatially neighboring CUs and/or temporally co-located CUs of the current CU and then selecting one member from the motion vector candidate list as a motion vector predictor for the current CU. By doing so, there is no need to transmit the motion vector candidate list itself from the video encoder 20 to the video decoder 30 and an index of the selected motion vector predictor within the motion vector candidate list is sufficient for the video encoder 20 and the video decoder 30 to use the same motion vector predictor within the motion vector candidate list for encoding and decoding the current CU.

[00163] The transform in the VVC and ECM has been designed more efficiently than that in the HEVC, both in the transform cores and signaling methods. More methods which aim to further improve the performance of transform coding have been proposed in the past decade, including MTS, LFNST, SDT and so on. However, there are still several drawbacks in these methods.

[00164] In the current VVC and ECM, the primary transform cores are derived from DCT and DST formulas. However, the derived transform cores may not adapt to all the video contents with different characteristics. Therefore, more efficient transform cores remain to be investigated.

[00165] In the current VVC and ECM, the transform matrices in LFNST are actually trained KLT. However, LFNST is only applied in the secondary transform. The KLT in the primary transform is not considered.

[00166] The signal dependent transform (SDT) in the JEM can achieve significant coding gain. However, the KLT matrices in SDT need to be online derived at both the encoder and decoder side by template matching. The complexity of template matching and eigenvalue decomposition is not acceptable especially for the decoder.

[00167] To train the KLT, training data should be collected firstly. In the SDT (online KLT training), training data is collected by template matching at both encoder and decoder side, which is too complex. In LFNST, all the transform blocks with different block sizes are firstly collected and these blocks are then divided into K groups using clustering method. Then one KLT matrix is derived for each group. In video coding, one index is signaled for each CU to indicate which transform matrix is used, i.e., which group the current transform block belongs to. This method may suffer from two drawbacks. Firstly, the training data classification is to divide the training data into different groups to guarantee that the training samples in one group conform to the same statistical characteristics. However, the training data classification in LFNST maybe not accurate enough. Secondly, the singled index at CU level also brings additional overhead bits.

[00168] In the current VVC and ECM, the transform cores are fixed which cannot adapt to the non-stationary characteristics of video contents.

[00169] In the current video coding standard, the forward and inverse transform share the same transform matrix to guarantee the perfect reconstruction. However, due to the quantization of transform coefficients, the perfect reconstruction cannot be guaranteed and the transform matrix sharing may not lead to the optimal compression performance.

[00170] In the current VVC and ECM, secondary transform termed as LFNST is applied to the transform block in which MTS is not applied, i.e., DCT-II is applied to both horizontal and vertical transforms. However, LFNST is neglected for the transform block coded with the MTS.

[00171] To tackle the above-mentioned problems of KLT in video coding, several methods are proposed to improve the performance of KLT in video coding: (i) This disclosure proposes an

offline trained KLT method for the primary transform; (ii) To improve the performance of the trained KLT, more efficient transform block (including the training data) classification methods are proposed; (iii) To deal with the diverse video statistical characteristics, the KLT matrices adaptive signaling methods are also proposed; (iv) To handle the quantization effect of transform coefficients, asymmetric KLT is proposed in which the forward and inverse transform may use different transform matrices; and (v) In this disclosure, it is proposed to enable LFNST for MTS.

[00172] In this section, KLT is firstly introduced briefly. Then the proposed KLT training and signaling methods are further proposed.

[00173] ***A Brief Introduction of KLT***

[00174] KLT is a kind of orthogonal transform based on the statistical property of the signal, which can achieve the optimal transform gain in terms of mean square error (MSE) metric. Denote the input vector as x, the output vector of KLT as X. The correlation between any two elements in X is zero, i.e.,

[00175] $$E[(X[u_1] - \bar{X}[u_1])(X[u_2] - \bar{X}[u_2])] = 0 \qquad (5\text{-}1)$$

[00176] Equation (4) is further converted as the following form:

[00177] $$E[X[u_1]X[u_2]] - \bar{X}[u_1]\bar{X}[u_2] = 0 \qquad (5\text{-}2)$$

[00178] If it is assumed that the input vector x is a zero-mean variable, then $\bar{X}[u_1]\bar{X}[u_2] = 0$ $(u_1 \neq u_2)$. Equation (5) is converted as:

[00179] $$E[X[u_1]X[u_2]] = 0 \qquad (5\text{-}3)$$

[00180] In this disclosure, the KLT matrix is denoted as $\boldsymbol{K}$, the transform can be described as follow:

[00181] $$X = \boldsymbol{K}x \qquad (5\text{-}4)$$

[00182] The covariance matrix of the X can be described as:

[00183] $$\mathrm{E}[XX^T] = \mathrm{E}[\boldsymbol{K}xx^T\boldsymbol{K}^T] = \boldsymbol{K}\mathrm{E}[xx^T]\boldsymbol{K}^T = \boldsymbol{K}C\boldsymbol{K}^T \qquad (5\text{-}5)$$

[00184] Where C is the covariance matrix of the input vector. To ensure that $E[X[u_1]X[u_2]] = 0$, $\mathrm{E}[XX^T]$ must be a diagonal matrix. In the other words, matrix $\boldsymbol{K}$ must transform matrix C into a diagonal matrix. That is to say the row vectors of matrix $\boldsymbol{K}$ are eigenvectors of matrix C.

[00185] ***KLT Training***

[00186] To derive the KLT for the signal, the statistical properties should be known, i.e., the covariance matrix of the signal. However, in practice it is difficult to obtain the covariance matrix

of the signal since the probability distribution is not available. Therefore, in the practical application, the KLT derivation is usually based on training from data. There are several issues in the training-based KLT derivation. The first issue is that given the training set how to derive the corresponding KLT matrix. The second issue is how to collect and classify training samples into different groups, in each group the training samples share the same statistical distribution.

[00187] ***KLT training method***

[00188] In this disclosure, KLT can be trained using singular value decomposition (SVD) or eigenvalue decomposition given the training data. Firstly, it is assumed that we have collected a set of training data with zero mean for the KLT, which consists of N blocks, x_i, $i = 1,2,\dots,N$ where $x_i = (x_{i1},\dots,x_{iD})^T$ and D represents the dimension of the vector which is just the transform block size. The training samples are normalized as $u_i = x_i/\sqrt{N}$. Then, we take these residual blocks as the training samples with zero means for deriving KLT. These N training samples can be represented by $\mathrm{U} = (u_1,\dots,u_N)$, which is a $D \times N$ matrix. We denote the covariance matrix Σ of those training blocks as:

[00189] $$\Sigma = \mathrm{U}\mathrm{U}^T \qquad (5\text{-}6)$$

[00190] Where the dimension of Σ is $D \times D$. The relationship between this covariance matrix and its eigenvectors is as:

[00191] $$\Sigma\Psi = \Psi\Lambda \qquad (5\text{-}7)$$

[00192] Where Ψ is the transform matrix with the column vectors being the eigenvectors (KLT bases) and Λ is a diagonal matrix with the diagonal elements being the eigenvalues. The eigenvectors can be obtained by performing eigenvalue decomposition or SVD on the covariance matrix. Note that we have sorted the eigenvalues together with the eigenvectors in descending order of the eigenvalues. This is to let the energy of the transform coefficients have descending order corresponding to the training samples.

[00193] ***Training data clustering***

[00194] In the proposed KLT training method, the samples are transform blocks retrieved from the encoding process. Then these samples are clustered into different groups and each KLT matrix is derived for each group.

[00195] In an embodiment, the training samples are clustered based on the transform block size. That it for each block size, one KLT matrix is derived.

[00196] In an embodiment, for intra prediction, the training samples are clustered based on transform block size and intra mode. That is for each transform block size, a set of KLT matrices are derived, in which one intra mode corresponds to one KLT matrix or several intra modes share one KLT matrix.

[00197] In an embodiment, for inter prediction the training samples are clustered based on transform block size and AMVR mode. That is for each transform block size, a set of KLT matrices are derived, in which one AMVR mode corresponds to one KLT matrix.

[00198] In an embodiment, for inter prediction the training samples are clustered based on transform block size and the absolute value of motion vector (MV). Several thresholds $\mathrm{th}_1, \mathrm{th}_2, \ldots, \mathrm{th}_M$ are predefined to classify the motion vector. That is for each transform block size, a set of KLT matrices are derived, in which one absolute value range of the motion vector corresponds to one KLT matrix. Here, the motion vector used for classification is the maximum value of horizontal MV and vertical MV values for both uni-prediction and bi-prediction.

[00199] In an embodiment, for inter prediction the training samples are clustered based on transform block size, prediction mode and neighboring template. It should be noted that for intra prediction, the prediction mode here refers to intra direction. While for inter prediction, the prediction mode refers to AMVR mode or motion vector value or inter prediction direction and so on. FIG. 10 is provided to show the template used in this embodiment for a certain transform block size, which consists of N row and N columns of reconstructed pixels.

[00200] The training process is illustrated as follow. In the training set of each transform block size and prediction mode, the neighboring templates are divided into K clusters using classical k-means clustering based on a certain distance metric, including but not limited to sum of square error (SSE) and sum of absolute difference (SAD). Then for each transform block size and prediction mode, K template centers are derived and fixed at both the encoder and decoder, as illustrated in FIG. 11. For each (transform block size, prediction mode, template center), one training set is obtained and the corresponding KLT matrix is derived.

[00201] In encoding and decoding process, for each transform block, the KLT transform set is firstly selected based on its block size and prediction mode. Then the distances between the template of the current block and the K template centers are calculated. The template center which leads to the minimum distance is selected and the corresponding KLT matrix is used for the current transform block.

[00202] In an embodiment, the proposed method in embodiment 4 is extended. In embodiment 4, for each (transform block size, prediction mode, template center), one KLT matrix is derived. In this embodiment, for each (transform block size, prediction mode, template center), the training set is further divided into several groups using classical k-means clustering as LFNST. In the encoding and decoding process, for each (transform block size, prediction mode, template center) pair, a set of transform matrices are selected and an index used to further indicate which transform matrix in the set is used for the current block.

[00203] ***QP-dependent Primary Transform***

[00204] In this disclosure, the QP-dependent primary transform is proposed. In the current VVC, ECM and the KLT transform methods, the primary transforms are QP independent, i.e., blocks with different QPs share the same transform matrices. In this disclosure, it is proposed to use different KLT transforms for different QPs. The KLT training process is described as follows. Firstly, the training data is collected using different QPs from the encoding process. Secondly, for each QP, the corresponding training set is utilized to derive the KLT matrices using the methods introduced above.

[00205] In the encoding and decoding process, for each transform block, in addition to the transform block size, prediction mode and neighboring template, its QP value is also utilized by the encoder or decoder to select the corresponding transform matrix.

[00206] ***KLT for Wide-Angle Intra Modes***

[00207] In this disclosure, different KLT matrices are trained for different intra prediction modes, i.e., for each intra prediction mode or intra prediction mode set, one KLT matrices set is trained. Here, the intra prediction mode set may consist of more than one intra prediction modes. As introduced above, wide-angle intra prediction modes are utilized for rectangular block in VVC and the ECM. Therefore, in this disclosure, KLT for wide-angle intra modes is considered. In one embodiment, only non-wide angular intra modes are utilized for KLT, i.e., 67 KLT matrices are trained for each block size in which each KLT matrix corresponds to one intra prediction mode ranging from 0 to 66. If the intra mode is wide angle for one block, the corresponding non-wide angle intra mode is utilized to select the corresponding KLT matrix.

[00208] In one embodiment, both wide and non-wide angular intra modes are utilized for KLT, i.e., each KLT matrix is trained for the intra mode ranging from -14~80. Therefore, if a block is

coded with wide-angle intra mode, the corresponding KLT matrix is identified and utilized for forward and inverse KLT.

[00209] ***Asymmetric KLT***

[00210] In the ordinary KLT, the forward transform and inverse transform share the same transform matrix to guarantee the perfect reconstruction property of the signal. However, in video/image coding, the transform coefficients are quantized after transform using a certain quantization parameter (QP). Therefore, the original signal cannot be reconstructed perfectly. To handle this problem, asymmetric KLT is proposed in this disclosure, in which the forward transform and inverse transform may use different transform matrices.

[00211] In the first embodiment, the transform units with the same size but coded using different QPs share the same forward KLT matrix while use different inverse KLT matrices. One example method to derive the forward and inverse KLT matrices is introduced as follows: (i) In the first step, for each TU size, the KLT matrix are derived for all the QP values according to the method described above. In this step, the derived forward KLT matrix and inverse KLT matrix share the same parameters; (ii) In the second step, the KLT matrices derived in the first step are integrated into video coder and the information of the TU using KLT is collected as the training data used in the following steps, including QP values, the original residues and the quantized transform blocks; and (iii) In the third step, the inverse KLT matrix are refined for each QP value or QP value range. The original residues and the quantized transform blocks for a certain QP value or QP value range are denoted as $\{r_1, r_2, \ldots, r_N\}$ and $\{\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_N\}$, respectively. The refinement of the inverse KLT matrix InvT can be described as follow.

[00212] $\text{InvT}^{opt} = \arg\min \sum_{i=1}^{N} D\{r_i, \text{InvT}(\hat{c}_i)\}$

[00213] In the second embodiment, the transform units with the same size but coded using different QPs share the same inverse KLT matrix while using different forward KLT matrices.

[00214] In the third embodiment, the transform units with the same size may use separate forward and inverse KLT matrices pair for different QPs, in which the forward and inverse matrices in each pair may have different matrix elements.

[00215] According to the one or more embodiments of the asymmetric KLT, for each TU size, several QP thresholds can be set: QP_1, QP_2,..., QP_K. These QP thresholds divide the QP values into K+1 ranges, and for each QP range, one set of forward/inverse KLT matrices are derived and applied.

[00216] ***KLT Signaling in Video Coding***

[00217] In this disclosure, some embodiments on KLT signaling are proposed as follows.

[00218] In the first embodiment, it is proposed to apply high level on/off control of KLT. A flag is signaled at the SPS or PPS or slice header to indicate whether KLT is applied to the sequence or picture or slice.

[00219] In the second embodiment, it is proposed to use KLT as additional transform candidate of MTS. Firstly, MTS CU flag is signaled to indicate whether MTS is used for the current CU. If MTS CU flag is true, the KLT CU flag is signaled to indicate whether KLT is used as the primary transform of the CU. If MTS CU flag is true and KLT CU flag is false, the transform pairs in the current VVC or ECM are utilized. If more than one KLT matrices are used for the CU, an KLT index should be further signaled.

[00220] In the third embodiment, it is proposed to replace DCT2 and LFNST with KLT. Firstly, MTS CU flag is signaled to indicate whether MTS is used for the current CU. If MTS CU flag is false, KLT is applied to the CU, otherwise MTS is applied.

[00221] In the fourth embodiment, it is proposed to replace DCT2 (and LFNST) and MTS. If only one KLT is enabled for the CU, then the KLT is directly applied to the CU. If more than one KLT is enabled for the CU, one KLT index is signaled to indicate which KLT matrix is applied. In the fifth embodiment, it is proposed to apply KLT as LFNST, i.e., secondary transform of MTS. If MTS CU flag is true, then LFNST flag is signaled. If both MTS CU flag and LFNST flag is true and more than one LFNST is enabled for the CU, then one LFNST index is signaled to indicate which KLT matrix is applied.

[00222] ***High-level KLT Matrices Signaling***

[00223] In the above sections, the training and block level signaling methods are proposed and described. More specifically, the trained KLT matrices are fixed at both encoder and decoder. In practical applications, the statistical characteristics of video contents are usually time-variant, i.e., the statistical characteristics may vary for different frames or video sequences. In these cases, it is difficult for the fixed KLT matrices to achieve the optimal compression efficiency. To tackle this problem of fixed KLT matrices, the high level adaptive KLT matrices signaling method is proposed in this disclosure: (i) In the first aspect, it is proposed to adaptively signal the KLT matrices in the sequence parameter set (SPS). At the encoder side, the KLT matrices are derived for the sequence. Then the KLT matrices are signaled in the SPS header; (2) In the second aspect,

it is proposed to adaptively signal the KLT matrices in the picture parameter set (PPS). At the encoder side, the KLT matrices are derived for each picture. Then the KLT matrices are signaled in the PPS header; and (3) In the third aspect, it is proposed to adaptively signal the KLT matrices in the slice header. At the encoder side, the KLT matrices are derived for each slice. Then the KLT matrices are signaled in the slice header.

[00224] It should be noted that two methods are proposed in this disclosure to code the elements in the KLT matrices. In the first method, the elements in the KLT matrices are directly coded using typical codes, i.e., Exponential-Golomb (EG) code. In the second method, the adaptive KLT matrices are predicted using the fixed KLT matrices, then the residues of the KLT matrices are further signaled using typical codes.

[00225] ***LFNST for MTS***

[00226] In the current VVC and ECM, LFNST is applied between forward primary transform and quantization (at encoder) and between de-quantization and inverse primary transform (at decoder side). It also should be noted that LFNST is only enabled when the MTS is not used in the primary transform, i.e., DCT-II is applied in both horizontal and vertical transform. When MTS is used in the primary transform, LFNST is not used. To further improve the coding performance of LFNST, it is proposed to apply LFNST to transform blocks coded with MTS. The proposed LFNST kernels may be trained or offline derived, and then fixed at both encoder and decoder side.

[00227] In one or more embodiments, there are totally 4 transform sets and 2 non-separable transform matrices (kernels) per transform set are used in the proposed LFNST for MTS. The mapping from the intra prediction mode to the transform set is pre-defined as shown in Table 3. If one of three CCLM modes (INTRA_LT_CCLM, INTRA_T_CCLM or INTRA_L_CCLM) is used for the current block (81 <= predModeIntra <= 83), transform set 0 is selected for the current chroma block. For each transform set, the selected non-separable secondary transform candidate is further specified by the explicitly signalled LFNST index. The index is signalled in a bit-stream once per Intra CU after transform coefficients.

[00228] In some other embodiments, there are 67 transform sets, i.e., for each intra prediction mode, a corresponding transform set is used and one non-separable transform matrix for each transform set is used in the proposed LFNST. If the CU is coded using TIMD mode, the intra mode is mapped to the transform index with the following method:

[00229] $\mathrm{trIdx} = \left(\mathrm{intraMode} < 2 ? \mathrm{intraMode} : \left((\mathrm{intraMode} \gg 1) + 1\right)\right)$

where intramode is the intra mode and trIdx is the index for the LFNST transform set.

[00230] It should be noted the proposed LFNST for MTS may be used but not limited to the above described two schemes. Since LFNST is restricted to be applicable only if all coefficients outside the first coefficient sub-group are non-significant, LFNST index coding depends on the position of the last significant coefficient. If more than one transform matrices are used in each transform set, Ithe LFNST index needs to be signaled, which can be context coded (not) depend on intra prediction mode. Furthermore, LFNST is applied for intra CU in both intra and inter slices, and for both Luma and Chroma. If a dual tree is enabled, LFNST indices for Luma and Chroma are signaled separately. For inter slice (the dual tree is disabled), a single LFNST index is signaled and used for both Luma and Chroma.

[00231] ***KLT Zeroing-out***

[00232] To reduce the computational complexity and the memory space of coefficients of the proposed KLT, zeroing-out of KLT is proposed in this disclosure, including the spatial domain zeroing-out and frequency domain zeroing-out.

[00233] In the spatial domain zeroing-out, the vector with dimension of N is mapped to a vector with dimension of K, where K is smaller than N. For a vector with dimension of N, the typical non-separable transform is the matrix multiplication between the KLT matrix and the vector, where the KLT matrix has the shape of NxN. In the proposed spatial domain zeroing-out method, the dimension of the forward KLT matrix is reduced from NxN to KxN. Accordingly, the inverse KLT matrix is the transpose of the forward KLT matrix with the dimension of NxK. Embodiments are proposed in this disclosure to derive the KLT for spatial domain zeroing-out. It should be noted that the method to derive the reduced KLT is not limited to the following embodiments.

[00234] In one embodiment, the KLT for the spatial domain zeroing-out is derived by discarding the eigenvectors in equation $(5-7)$. Denote the KLT matrix as $\Psi = [\varphi_1, \varphi_2, \ldots, \varphi_N]$ with φ_i being the eigenvectors (KLT bases). The eigenvectors can be obtained by performing eigenvalue decomposition or SVD on the covariance matrix. Note that we have sorted the eigenvalues together with the eigenvectors in descending order of the eigenvalues to make the energy of the transform coefficients have descending order corresponding to the training samples. In this embodiment, the eigenvectors with smaller eigenvalues are discarded. The reduced KLT matrix can be represented as follow:

[00235] $\Psi^* = [\varphi_1, \varphi_2, \ldots, \varphi_K], \quad K < N$

[00236] The proposed frequency domain zeroing-out is applied when KLT is used for the secondary transform of MTS. After the MTS is conducted on the residue block, the primary transform block is obtained. Only some primary transform coefficients are kept while other coefficients are discarded. FIG. 12A-D provide some example zeroing-out methods of the primary transform block, in which the blue part represents the area that transform coefficients are retained.

[00237] FIG. 12A provides an example that the transform coefficients in the left-above area are retained. FIG. 12B provides an example that the transform coefficients in the above area are retained. FIG. 12C provides an example that the transform coefficients in the left area are retained. FIG. 12D provides an example that the transform coefficients in the left and above areas are retained, where the left and above areas include both the above area and the left area.

[00238] In some embodiments, how to apply the zeroing-out methods may be illustrated in FIG. 12A-D. In one embodiment, only one of the zeroing-out method in FIG. 12A-D is applied for the primary transform block. In yet another embodiment, zeroing-out methods switch scheme is proposed. For each primary transform block, an index is signaled in the bitstream to indicate which zeroing-out method is applied to the block.

[00239] ***KLT Matrix Sharing***

[00240] In the proposed KLT, the residue block or primary transform block is firstly converted from a rectangular block to a vector. Take the following 4x2 block as an example.

[00241] $X = \begin{bmatrix} X_{00} & X_{01} & X_{02} & X_{03} \\ X_{10} & X_{11} & X_{12} & X_{13} \end{bmatrix}$

[00242] X is converted to a vector of eight dimensions using raster scan order or vertical scan order:

[00243] $\vec{X} = [X_{00}\ X_{01}\ X_{02}\ X_{03}\ X_{10}\ X_{11}\ X_{12}\ X_{13}]$ or $\vec{X} = [X_{00}\ X_{10}\ X_{01}\ X_{11}\ X_{02}\ X_{12}\ X_{03}\ X_{13}]$

[00244] KLT operation can be described as $\vec{Y} = \Psi\vec{X}$, where Ψ is the KLT matrix.

[00245] In the typical implementation, the training and derivation of KLT matrix is block shape dependent, i.e., different KLT matrices are trained for different block shapes. For example, different KLT matrices are used for the MxN block and NxM block when M is not equal to N.

[00246] To reduce the memory to store the coefficients of the KLT matrices, KLT matrix sharing methods are proposed in this disclosure. The basic idea of KLT kernel sharing is to share the same KLT matrix among the blocks with the same area. Several embodiments are proposed for the KLT matrix sharing.

[00247] In one embodiment, the blocks with the same area (number of samples in the block) share the same KLT matrix or KLT matrix set. For example, 16x4 block, 8x8 block and 4x16 block have different block shape, but the same KLT matrix is applied to these blocks.

[00248] In yet another embodiment, if the width of one block A is the same with the height of another block B, and the height of block A is the same with the width of block B, then block A and block B share the same KLT matrix or KLT matrix set. For example, HxW block and WxH block can share the same KLT matrix or KLT matrix set. In the embodiment, to use the shared KLT matrix, both the HxW block and WxH block are firstly converted to the vectors with the same dimension of WxH using certain scan orders.

[00249] In the first method, raster scan order, vertical scan order or any other scan order is applied to both the HxW block and WxH block to convert the blocks to the vectors.

[00250] In the second method, when converting the blocks to the vectors, raster scan order is applied to the block with the shape of WxH, while vertical scan order is applied to the block with the shape of HxW. Or vertical scan order is applied to the block with the shape of WxH, while raster scan order is applied to the block with the shape of HxW.

[00251] In the third method, any certain scan order (including but not limited to raster scan and vertical scan order) is applied to the block with the shape of WxH. The block with the shape of HxW is firstly transposed to the block with the shape of WxH. Then the scan order for the shape of WxH is applied to the transposed block to convert it into a vector.

[00252] In embodiments of the present disclosure, the block is firstly converted to a vector with raster scan or vertical scan. In one example embodiment of KLT matrix sharing, for a block with size of WxH, if the height H is larger than its width W, the block is firstly transposed to a block with size of HxW as illustrated in FIG.13 and then converted to a vector using raster scan or vertical scan. The corresponding KLT matrix is identified with the width, height and intra mode of the transposed block. As shown in FIG.13, the WxH block before transposing is coded with intra mode of m1, i.e., the prediction direction is from bottom-left to above-right. However, after transposing the block, in addition to the change of the block shape, the prediction direction also changes, i.e., the intra mode is changed to m2. Therefore, the corresponding KLT matrix of the transposed block is identified with intra mode m2 instead of m1. From the perspective of the decoder, when the block with shape of WxH and intra mode m1 is a rectangular block and its

height is larger than width, the shape of the transposed block and the mapped intra mode m2 is used to identify the KLT matrix. The mapped intra mode m2 is derived as follows.

[00253] In one embodiment, if wide-angle intra modes are not considered for KLT as described in the Section "KLT for Wide-Angle Intra Modes," the mapped intra mode m2 is derived as follows:

If m1 is DC or planar mode, then m2 = m1;

Otherwise, m2 = 68 – m1

[00254] In yet another embodiment, if wide-angle intra modes are considered for KLT as described in the Section "KLT for Wide-Angle Intra Modes," the mapped intra mode m2 is derived as follows:

If m1 is DC or planar mode, then m2 = m1;

Else if -14 <= m1 <= -1, then m2 = 66-m1;

Otherwise, m2 = 68 – m1

[00255] It should be noted that in the above example implementation, the block is transposed if its height is larger than its width. However, the proposed KLT matrix sharing method is not limited to this kind of implementation, other implementation methods are also allowed. For example, we may set that a block needs to be transposed if its width is larger than its height.

[00256] ***Multiple KLT Kernels Selection***

[00257] In the previous sections, for each block size and intra mode, only one KLT transform kernel is derived and applied. However, due to the diversity of the residue signal characteristics, one single transform kernel cannot capture all the signal distributions. To further improve the proposed KLT, in this disclosure multiple KLT kernel selection method is provided. In the multiple KLT kernel selection, for each block size and intra mode, more than one KLT transform kernels are derived and utilized. For signaling, the KLT flag is firstly signaled to indicate whether KLT is enabled. If KLT flag is true, the KLT kernel index is further signaled to indicate which KLT kernel is used for the current block.

[00258] In some examples of the present disclosure, the number of KLT kernels for each block size and intra mode is not limited. Also, the number of KLT kernels of different block size may be different. For example, three KLT kernels are derived for blocks with size of 4x4 while 4 KLT kernels are derived for blocks with size of 16x16.

[00259] Several embodiments are provided to signal the syntax elements for KLT with multiple kernels selection.

[00260] In one embodiment, the case of two KLT kernels is provided, that is for the blocks with certain size, two KLT kernels are derived. As shown in Table 6 below, in this embodiment, the first bin is context coded with context C0. If KLT is enabled, the second bin is then coded with context C1 to indicate which KLT kernel is utilized.

Binarized value	Context	Explanation
0	C_0	KLT is not enabled
10	C_0C_1	KLT kernel 1 is used
11	C_0C_1	KLT kernel 2 is used

Table 6

[00261] In another embodiment, the case of two KLT kernels is provided, that is for the blocks with certain size, two KLT kernels are derived. As shown in Table 7 below, in this embodiment, the first bin is context coded with context C0. If KLT is enabled, the second bin is then coded with bypass mode, i.e., equal probability to indicate which KLT kernel is utilized.

Binarized value	Context	Explanation
0	C_0	KLT is not enabled
10	C_0E	KLT kernel 1 is used
11	C_0E	KLT kernel 2 is used

Table 7

[00262] In yet another embodiment, the case of three KLT kernels is provided, that is for the blocks with certain size, three KLT kernels are derived. As shown in Table 8 below, in this embodiment, the first bin is context coded with context C_0. If KLT is enabled, the second and third bins are then coded with context C_1 and C_2, respectively to indicate which KLT kernel is utilized.

Binarized value	Context	Explanation
0	C_0	KLT is not enabled
11	C_0C_1	KLT kernel 1 is used
100	$C_0C_1C_2$	KLT kernel 2 is used
101	$C_0C_1C_2$	KLT kernel 3 is used

Table 8

[00263] In yet another embodiment, the case of three KLT kernels is provided, that is for the blocks with certain size, three KLT kernels are derived. As shown in Table 9 below, in this embodiment, the first bin is context coded with context C_0. If KLT is enabled, the second bin is then coded with context C_1. The third bin is coded with bypass mode, i.e., equal probability.

Binarized value	Context	Explanation
0	C_0	KLT is not enabled
11	C_0C_1	KLT kernel 1 is used
100	C_0C_1E	KLT kernel 2 is used
101	C_0C_1E	KLT kernel 3 is used

Table 9

[00264] In yet another embodiment, the case of four KLT kernels is provided, that is for the blocks with certain size, four KLT kernels are derived. As shown in Table 10 below, in this embodiment, the first bin is context coded with context C_0. If KLT is enabled, the second and third bins are then coded with context C_1 and C_2, respectively.

Binarized value	Context	Explanation
0	C_0	KLT is not enabled
111	$C_0C_1C_2$	KLT kernel 1 is used
110	$C_0C_1C_2$	KLT kernel 2 is used
100	$C_0C_1C_2$	KLT kernel 3 is used
101	$C_0C_1C_2$	KLT kernel 4 is used

Table 10

[00265] In yet another embodiment, the case of four KLT kernels is provided, that is for the blocks with certain size, four KLT kernels are derived. As shown in Table 11 below, in this embodiment, the first bin is context coded with context C_0. If KLT is enabled, the second bin is then coded with context C_1. The third bin is coded with bypass mode, i.e., equal probability.

Binarized value	Context	Explanation
0	C_0	KLT is not enabled
111	C_0C_1E	KLT kernel 1 is used
110	C_0C_1E	KLT kernel 2 is used
100	C_0C_1E	KLT kernel 3 is used
101	C_0C_1E	KLT kernel 4 is used

Table 11

[00266] In embodiments of the present disclosure, the following methods and apparatus are proposed to derive KLT matrices and apply these new matrices in video coding. The algorithms of the KLT training are introduced. The improved KLT training data clustering methods are proposed. The signaling methods of KLT in video coding are proposed. Asymmetric KLT method is proposed to handle the quantization effect of the transform coefficients. LFNST for MTS is also proposed to further improve the coding efficiency of LFNST. Finally, the high-level KLT matrices signaling methods are proposed. KLT zeroing-out methods are proposed to reduce the

computational complexity and memory to store the KLT matrices. KLT matrix sharing methods are proposed to reduce the memory to store the KLT matrices. It is expected to further improve the coding efficiency of transform coding in the VVC and ECM.

[00267] FIG. 14 is a flowchart illustrating a method 1400 for video decoding in accordance with some examples of the present disclosure. At step 1401, the method 1400 includes obtaining, by a decoder, a plurality of Karhunen-Loève Transform (KLT) transform kernels for a current block in an intra mode. At step 1402, the method includes selecting, by the decoder, a KLT transform kernel for the current block in the intra mode from the plurality of KLT transform kernels.

[00268] In some examples, the method 1400 further includes: receiving, by the decoder and from an encoder, a KLT flag that indicates whether KLT is enabled; and performing, by the decoder, one of following acts: in response to determining that the KLT flag is true, determining, by the decoder, that the KLT is enabled, and receiving, by the decoder and from the encoder, a KLT kernel index; or in response to determining that the KLT flag is false, determining, by the decoder, that the KLT is not enabled. In some examples, step 1402 includes: in response to determining that the KLT flag is true, selecting, by the decoder, the KLT transform kernel for the current block in the intra mode based on the KLT kernel index.

[00269] In some examples, the KLT kernel index is determined using a Rate-Distortion Optimization (RDO) technique.

[00270] In some examples, a number of the plurality of KLT transform kernels for each block size and intra mode is not limited. In some examples, the plurality of KLT transform kernels correspond to one or more first blocks of a first size and one or more second blocks of a second size, the first size is different from the second size, and a first number of the one or more first blocks is different from a second number of the one or more second blocks. Particularly, in some examples, the first size is 4×4, the first number is 3, the second size is 16×16, and the second number is 4.

[00271] In some examples, the KLT kernel index is a binarized value including one or more bins, and the one or more bins correspond to one or more contexts.

[00272] In some examples, the plurality of KLT transform kernels include a first KLT transform kernel and a second KLT transform kernel; the binarized value includes a first bin and a second bin; the one or more contexts include a first context and a second context; the first bin is context coded using the first context; and the second bin is context coded with the second context, or coded

in a bypass mode. In some examples, being coded in a bypass mode means equal probability to indicate which KLT kernel is utilized.

[00273] In some examples, step 1402 includes one of following steps: in response to determining that the first bin is 0, determining that the KLT is not enabled; in response to determining that the first bin is 1 and the second bin is 0, selecting the first KLT transform kernel as the KLT transform kernel; or in response to determining that the first bin is 1 and the second bin is 1, selecting the second KLT transform kernel as the KLT transform kernel.

[00274] In some examples, the plurality of KLT transform kernels comprise a first KLT transform kernel, a second KLT transform kernel, and a third KLT transform kernel; the binarized value comprises a first bin, a second bin, and a third bin; the one or more contexts comprise a first context, a second context, and a third context; the first bin is context coded using the first context; the second bin is context coded using the second context; and the third bin is context coded with the third context, or coded in a bypass mode. In some examples, being coded in a bypass mode means equal probability to indicate which KLT kernel is utilized,

[00275] In some examples, step 1402 includes one of following steps: in response to determining that the first bin is 0, determining that the KLT is not enabled; in response to determining that the first bin is 1 and the second bin is 1, selecting the first KLT transform kernel as the KLT transform kernel; in response to determining that the first bin is 1, the second bin is 0, and the third bin is 0, selecting the second KLT transform kernel as the KLT transform kernel; or in response to determining that the first bin is 1, the second bin is 0, and the third bin is 1, selecting the third KLT transform kernel as the KLT transform kernel.

[00276] In some examples, the plurality of KLT transform kernels comprise a first KLT transform kernel, a second KLT transform kernel, a third KLT transform kernel, and a fourth KLT transform kernel; the binarized value comprises a first bin, a second bin, and a third bin; the one or more contexts comprise a first context, a second context, and a third context; the first bin is context coded using the first context; the second bin is context coded using the second context; and the third bin is context coded with the third context, or coded in a bypass mode. In some examples, being coded in a bypass mode means equal probability to indicate which KLT kernel is utilized.

[00277] In some examples, step 1402 includes one of following steps: in response to determining that the first bin is 0, determining that the KLT is not enabled; in response to

determining that the first bin is 1, the second bin is 1, and the third bin is 1, selecting the first KLT transform kernel as the KLT transform kernel; in response to determining that the first bin is 1, the second bin is 1, and the third bin is 0, selecting the second KLT transform kernel as the KLT transform kernel; in response to determining that the first bin is 1, the second bin is 0, and the third bin is 0, selecting the third KLT transform kernel as the KLT transform kernel; or in response to determining that the first bin is 1, the second bin is 0, and the third bin is 1, selecting the fourth KLT transform kernel as the KLT transform kernel.

[00278] FIG. 15 is a flowchart illustrating a method 1500 for video encoding in accordance with some examples of the present disclosure and in correspondence with the method for video decoding illustrated in FIG. 14. At step 1501, the method 1500 includes obtaining, by an encoder, a plurality of Karhunen-Loève Transform (KLT) transform kernels for a current block in an intra mode. At step 1502, the method includes determining, by the encoder, that KLT is enabled for the current block. At step 1503, the method includes selecting, by the encoder, a KLT transform kernel for the current block in the intra mode from the plurality of KLT transform kernels.

[00279] In some examples, the method 1500 further includes: signaling, by the encoder and to a decoder, a KLT flag that indicates whether the KLT is enabled; and performing, by the encoder, one of following acts: in response to determining that the KLT is enabled, setting, by the encoder, the KLT flag as true, and signaling, by the encoder and to the decoder, a KLT kernel index; or in response to determining that the KLT is not enabled, setting, by the encoder, the KLT flag as false. In some examples, step 1503 includes: in response to determining that the KLT is enabled, selecting, by the encoder, the KLT transform kernel for the current block in the intra mode based on the KLT kernel index.

[00280] In some examples, the KLT kernel index is determined using a Rate-Distortion Optimization (RDO) technique. The RDO technique generally minimizes the amount of distortion (loss of video quality) against the amount of data required to encode the video, the rate.

[00281] In some examples, a number of the plurality of KLT transform kernels for each block size and intra mode is not limited. In some examples, the plurality of KLT transform kernels correspond to one or more first blocks of a first size and one or more second blocks of a second size, the first size is different from the second size, and a first number of the one or more first blocks is different from a second number of the one or more second blocks. Particularly, in some

examples, the first size is 4 × 4, the first number is 3, the second size is 16 × 16, and the second number is 4.

[00282] In some examples, the KLT kernel index is a binarized value including one or more bins, and the one or more bins correspond to one or more contexts.

[00283] In some examples, the plurality of KLT transform kernels include a first KLT transform kernel and a second KLT transform kernel; the binarized value includes a first bin and a second bin; the one or more contexts include a first context and a second context; the first bin is context coded using the first context; and the second bin is context coded with the second context, or coded in a bypass mode. In some examples, being coded in a bypass mode means equal probability to indicate which KLT kernel is utilized.

[00284] In some examples, step 1503 includes one of following steps: in response to determining that the KLT is not enabled, setting the first bin as 0; in response to selecting the first KLT transform kernel as the KLT transform kernel, setting the first bin as 1 and the second bin as 0,; or in response to selecting the second KLT transform kernel as the KLT transform kernel, setting the first bin as 1 and the second bin as 1.

[00285] In some examples, the plurality of KLT transform kernels comprise a first KLT transform kernel, a second KLT transform kernel, and a third KLT transform kernel; the binarized value comprises a first bin, a second bin, and a third bin; the one or more contexts comprise a first context, a second context, and a third context; the first bin is context coded using the first context; the second bin is context coded using the second context; and the third bin is context coded with the third context, or coded in a bypass mode. In some examples, being coded in a bypass mode means equal probability to indicate which KLT kernel is utilized.

[00286] In some examples, step 1503 includes one of following steps: in response to determining that the KLT is not enabled, setting the first bin as 0; in response to selecting the first KLT transform kernel as the KLT transform kernel, setting the first bin as 1 and the second bin as 1; in response to selecting the second KLT transform kernel as the KLT transform kernel, setting the first bin as 1, the second bin as 0, and the third bin as 0; or in response to selecting the third KLT transform kernel as the KLT transform kernel, setting the first bin as 1, the second bin as 0, and the third bin as 1.

[00287] In some examples, the plurality of KLT transform kernels comprise a first KLT transform kernel, a second KLT transform kernel, a third KLT transform kernel, and a fourth KLT

transform kernel; the binarized value comprises a first bin, a second bin, and a third bin; the one or more contexts comprise a first context, a second context, and a third context; the first bin is context coded using the first context; the second bin is context coded using the second context; and the third bin is context coded with the third context, or coded in a bypass mode. In some examples, being coded in a bypass mode means equal probability to indicate which KLT kernel is utilized.

[00288] In some examples, step 1503 includes one of following steps: in response to determining that the KLT is not enabled, setting the first bin as 0; in response to selecting the first KLT transform kernel as the KLT transform kernel, setting the first bin as 1, the second bin as 1, and the third bin as 1; in response to selecting the second KLT transform kernel as the KLT transform kernel, setting the first bin as 1, the second bin as 1, and the third bin as 0; in response to selecting the third KLT transform kernel as the KLT transform kernel, setting the first bin as 1, the second bin as 0, and the third bin as 0; or in response to selecting the fourth KLT transform kernel as the KLT transform kernel, setting the first bin as 1, the second bin as 0, and the third bin as 1.

[00289] FIG. 16 shows a computing environment 1610 coupled with a user interface 1650. The computing environment 1610 can be part of a data processing server. The computing environment 1610 includes a processor 1620, a memory 1630, and an Input/Output (I/O) interface 1640.

[00290] The processor 1620 typically controls overall operations of the computing environment 1610, such as the operations associated with display, data acquisition, data communications, and image processing. The processor 1620 may include one or more processors to execute instructions to perform all or some of the steps in the above-described methods. Moreover, the processor 1620 may include one or more modules that facilitate the interaction between the processor 1620 and other components. The processor may be a Central Processing Unit (CPU), a microprocessor, a single chip machine, a Graphical Processing Unit (GPU), or the like.

[00291] The memory 1630 is configured to store various types of data to support the operation of the computing environment 1610. The memory 1630 may include predetermined software 1632. Embodiments of such data includes instructions for any applications or methods operated on the computing environment 1610, video datasets, image data, etc. The memory 1630 may be implemented by using any type of volatile or non-volatile memory devices, or a combination thereof, such as a Static Random Access Memory (SRAM), an Electrically Erasable

Programmable Read-Only Memory (EEPROM), an Erasable Programmable Read-Only Memory (EPROM), a Programmable Read-Only Memory (PROM), a Read-Only Memory (ROM), a magnetic memory, a flash memory, a magnetic or optical disk.

[00292] In one example, the memory 1630 is configured to store instructions executable by the processor; where the processor, upon execution of the instructions, is configured to perform any method as illustrated in FIGS. 14-15.

[00293] The I/O interface 1640 provides an interface between the processor 1620 and peripheral interface modules, such as a keyboard, a click wheel, buttons, and the like. The buttons may include but are not limited to, a home button, a start scan button, and a stop scan button. The I/O interface 1640 can be coupled with an encoder and decoder.

[00294] In an embodiment, there is also provided a non-transitory computer-readable storage medium comprising a plurality of programs, for example, in the memory 1630, executable by the processor 1620 in the computing environment 1610, for performing the above-described methods and/or storing a bitstream generated by the encoding method described above or a bitstream to be decoded by the decoding method described above. In one example, the plurality of programs may be executed by the processor 1620 in the computing environment 1610 to receive (for example, from the video encoder 20 in FIG. 2) a bitstream or data stream including encoded video information (for example, video blocks representing encoded video frames, and/or associated one or more syntax elements, etc.), and may also be executed by the processor 1620 in the computing environment 1610 to perform the decoding method described above according to the received bitstream or data stream. In another example, the plurality of programs may be executed by the processor 1620 in the computing environment 1610 to perform the encoding method described above to encode video information (for example, video blocks representing video frames, and/or associated one or more syntax elements, etc.) into a bitstream or data stream, and may also be executed by the processor 1620 in the computing environment 1610 to transmit the bitstream or data stream (for example, to the video decoder 30 in FIG. 3A). Alternatively, the non-transitory computer-readable storage medium may have stored therein a bitstream or a data stream comprising encoded video information (for example, video blocks representing encoded video frames, and/or associated one or more syntax elements) generated by an encoder (for example, the video encoder 20 in FIG. 2) using, for example, the encoding method described above for use by a decoder (for example, the video decoder 30 in Fig. 3A or Fig. 3B) in decoding video data. The

non-transitory computer-readable storage medium may be, for example, a ROM, a Random Access Memory (RAM), a CD-ROM, a magnetic tape, a floppy disc, an optical data storage device or the like.

[00295] In an embodiment, there is provided a bitstream generated by the encoding method described above or a bitstream to be decoded by the decoding method described above. In an embodiment, there is provided a bitstream comprising encoded video information generated by the encoding method described above or encoded video information to be decoded by the decoding method described above.

[00296] In an embodiment, the is also provided a computing device comprising one or more processors (for example, the processor 1620); and the non-transitory computer-readable storage medium or the memory 1630 having stored therein a plurality of programs executable by the one or more processors, wherein the one or more processors, upon execution of the plurality of programs, are configured to perform the above-described methods.

[00297] In an embodiment, there is also provided a computer program product having instructions for storage or transmission of a bitstream comprising encoded video information generated by the encoding method described above or encoded video information to be decoded by the decoding method described above. In an embodiment, there is also provided a computer program product comprising a plurality of programs, for example, in the memory 1630, executable by the processor 1620 in the computing environment 1610, for performing the above-described methods. For example, the computer program product may include the non-transitory computer-readable storage medium.

[00298] In an embodiment, the computing environment 1610 may be implemented with one or more ASICs, DSPs, Digital Signal Processing Devices (DSPDs), Programmable Logic Devices (PLDs), FPGAs, GPUs, controllers, micro-controllers, microprocessors, or other electronic components, for performing the above methods.

[00299] In an embodiment, there is also provided a method of storing a bitstream, comprising storing the bitstream on a digital storage medium, wherein the bitstream comprises encoded video information generated by the encoding method described above or encoded video information to be decoded by the decoding method described above.

[00300] In an embodiment, there is also provided a method for transmitting a bitstream generated by the encoder described above. In an embodiment, there is also provided a method for receiving a bitstream to be decoded by the decoder described above.

[00301] The description of the present disclosure has been presented for purposes of illustration and is not intended to be exhaustive or limited to the present disclosure. Many modifications, variations, and alternative implementations will be apparent to those of ordinary skill in the art having the benefit of the teachings presented in the foregoing descriptions and the associated drawings.

[00302] Unless specifically stated otherwise, an order of steps of the method according to the present disclosure is only intended to be illustrative, and the steps of the method according to the present disclosure are not limited to the order specifically described above, but may be changed according to practical conditions. In addition, at least one of the steps of the method according to the present disclosure may be adjusted, combined or deleted according to practical requirements.

[00303] The embodiments were chosen and described in order to explain the principles of the disclosure and to enable others skilled in the art to understand the disclosure for various implementations and to best utilize the underlying principles and various implementations with various modifications as are suited to the particular use contemplated. Therefore, it is to be understood that the scope of the disclosure is not to be limited to the specific embodiments of the implementations disclosed and that modifications and other implementations are intended to be included within the scope of the present disclosure.

[00304] The above methods may be implemented using an apparatus that includes one or more circuitries, which include application specific integrated circuits (ASICs), digital signal processors (DSPs), digital signal processing devices (DSPDs), programmable logic devices (PLDs), field programmable gate arrays (FPGAs), controllers, micro-controllers, microprocessors, or other electronic components. The apparatus may use the circuitries in combination with the other hardware or software components for performing the above described methods. Each module, sub-module, unit, or sub-unit disclosed above may be implemented at least partially using the one or more circuitries.

[00305] Other embodiments of the disclosure will be apparent to those skilled in the art from consideration of the specification and practice of the disclosure disclosed here. This application is intended to cover any variations, uses, or adaptations of the disclosure following the general

principles thereof and including such departures from the present disclosure as come within known or customary practice in the art. It is intended that the specification and embodiments be considered as exemplary only. The specification and embodiments are considered as exemplary. The application is intended to cover any variations, uses, or adaptations of the disclosure.

[00306] It will be appreciated that the present disclosure is not limited to the exact embodiments described above and illustrated in the accompanying drawings, and that various modifications and changes can be made without departing from the scope thereof.

WHAT IS CLAIMED IS:

1. A method for video decoding, comprising:

obtaining, by a decoder, a plurality of Karhunen-Loève Transform (KLT) transform kernels for a current block in an intra mode; and

selecting, by the decoder, a KLT transform kernel for the current block in the intra mode from the plurality of KLT transform kernels.

2. The method of claim 1, further comprising:

receiving, by the decoder and from an encoder, a KLT flag that indicates whether KLT is enabled; and

performing, by the decoder, one of following acts:

in response to determining that the KLT flag is true, determining, by the decoder, that the KLT is enabled, and receiving, by the decoder and from the encoder, a KLT kernel index; or

in response to determining that the KLT flag is false, determining, by the decoder, that the KLT is not enabled.

3. The method of claim 2, wherein selecting, by the decoder, the KLT transform kernel for the current block in the intra mode from the plurality of KLT transform kernels comprises:

in response to determining that the KLT flag is true, selecting, by the decoder, the KLT transform kernel for the current block in the intra mode based on the KLT kernel index.

4. The method of claim 1, wherein the plurality of KLT transform kernels correspond to one or more first blocks of a first size and one or more second blocks of a second size, the first size is different from the second size, and a first number of the one or more first blocks is different from a second number of the one or more second blocks.

5. The method of claim 4, wherein the first size is 4×4, the first number is 3, the second size is 16×16, and the second number is 4.

6. The method of claim 2, wherein the KLT kernel index is a binarized value comprising one or more bins, and the one or more bins correspond to one or more contexts.

7. The method of claim 6, wherein:

the plurality of KLT transform kernels comprise a first KLT transform kernel and a second KLT transform kernel;

the binarized value comprises a first bin and a second bin;

the one or more contexts comprise a first context and a second context;

the first bin is context coded using the first context; and

the second bin is coded with the second context, or coded in a bypass mode.

8. The method of claim 7, wherein selecting, by the decoder, the KLT transform kernel for the current block in the intra mode from the plurality of KLT transform kernels comprises one of following steps:

in response to determining that the first bin is 0, determining that the KLT is not enabled;

in response to determining that the first bin is 1 and the second bin is 0, selecting the first KLT transform kernel as the KLT transform kernel; or

in response to determining that the first bin is 1 and the second bin is 1, selecting the second KLT transform kernel as the KLT transform kernel.

9. The method of claim 6, wherein:

the plurality of KLT transform kernels comprise a first KLT transform kernel, a second KLT transform kernel, and a third KLT transform kernel;

the binarized value comprises a first bin, a second bin, and a third bin;

the one or more contexts comprise a first context, a second context, and a third context;

the first bin is context coded using the first context;

the second bin is context coded using the second context; and

the third bin is context coded with the third context, or coded with in a bypass mode.

10. The method of claim 9, wherein selecting, by the decoder, the KLT transform kernel for the current block in the intra mode from the plurality of KLT transform kernels comprises one of following steps:

in response to determining that the first bin is 0, determining that the KLT is not enabled;

in response to determining that the first bin is 1 and the second bin is 1, selecting the first KLT transform kernel as the KLT transform kernel;

in response to determining that the first bin is 1, the second bin is 0, and the third bin is 0, selecting the second KLT transform kernel as the KLT transform kernel; or

in response to determining that the first bin is 1, the second bin is 0, and the third bin is 1, selecting the third KLT transform kernel as the KLT transform kernel.

11. The method of claim 6, wherein:

the plurality of KLT transform kernels comprise a first KLT transform kernel, a second KLT transform kernel, a third KLT transform kernel, and a fourth KLT transform kernel;

the binarized value comprises a first bin, a second bin, and a third bin;

the one or more contexts comprise a first context, a second context, and a third context;

the first bin is context coded using the first context;

the second bin is context coded using the second context; and

the third bin is context coded with the third context, or coded in a bypass mode.

12. The method of claim 11, wherein selecting, by the decoder, the KLT transform kernel for the current block in the intra mode from the plurality of KLT transform kernels comprises one of following steps:

in response to determining that the first bin is 0, determining that the KLT is not enabled;

in response to determining that the first bin is 1, the second bin is 1, and the third bin is 1, selecting the first KLT transform kernel as the KLT transform kernel;

in response to determining that the first bin is 1, the second bin is 1, and the third bin is 0, selecting the second KLT transform kernel as the KLT transform kernel;

in response to determining that the first bin is 1, the second bin is 0, and the third bin is 0, selecting the third KLT transform kernel as the KLT transform kernel; or

in response to determining that the first bin is 1, the second bin is 0, and the third bin is 1, selecting the fourth KLT transform kernel as the KLT transform kernel.

13. A method for video encoding, comprising:

obtaining, by an encoder, a plurality of Karhunen-Loève Transform (KLT) transform kernels for a current block in an intra mode;

determining, by the encoder, that KLT is enabled for the current block; and

selecting, by the encoder, a KLT transform kernel for the current block in the intra mode from the plurality of KLT transform kernels.

14. The method of claim 13, further comprising:

signaling, by the encoder and to a decoder, a KLT flag that indicates whether the KLT is enabled; and

performing, by the encoder, one of following acts:

in response to determining that the KLT is enabled, setting, by the encoder, the KLT flag as true, and signaling, by the encoder and to the decoder, a KLT kernel index; or

in response to determining that the KLT is not enabled, setting, by the encoder, the KLT flag as false.

15. The method of claim 14, wherein selecting, by the encoder, the KLT transform kernel for the current block in the intra mode from the plurality of KLT transform kernels comprises:

in response to determining that the KLT is enabled, selecting, by the encoder, the KLT transform kernel for the current block in the intra mode based on the KLT kernel index.

16. The method of claim 13, wherein the plurality of KLT transform kernels correspond to one or more first blocks of a first size and one or more second blocks of a second size, the first size is different from the second size, and a first number of the one or more first blocks is different from a second number of the one or more second blocks.

17. The method of claim 16, wherein the first size is 4 × 4, the first number is 3, the second size is 16 × 16, and the second number is 4.

18. The method of claim 14, wherein the KLT kernel index is a binarized value comprising one or more bins, and the one or more bins correspond to one or more contexts.

19 The method of claim 18, wherein:

the plurality of KLT transform kernels comprise a first KLT transform kernel and a second KLT transform kernel;

the binarized value comprises a first bin and a second bin;

the one or more contexts comprise a first context and a second context;

the first bin is context coded using the first context; and

the second bin is context coded with the second context, or coded in a bypass mode.

20 The method of claim 19, wherein selecting, by the encoder, the KLT transform kernel for the current block in the intra mode from the plurality of KLT transform kernels comprises one of following steps:

in response to determining that the KLT is not enabled, setting the first bin as 0;

in response to selecting the first KLT transform kernel as the KLT transform kernel, setting the first bin as 1 and the second bin as 0; or

in response to selecting the second KLT transform kernel as the KLT transform kernel, setting the first bin as 1 and the second bin as 1.

21. The method of claim 18, wherein:

the plurality of KLT transform kernels comprise a first KLT transform kernel, a second KLT transform kernel, and a third KLT transform kernel;

the binarized value comprises a first bin, a second bin, and a third bin;

the one or more contexts comprise a first context, a second context, and a third context;

the first bin is context coded using the first context;

the second bin is context coded using the second context; and

the third bin is context coded with the third context, or coded in a bypass mode.

22. The method of claim 21, wherein selecting, by the encoder, the KLT transform kernel for the current block in the intra mode from the plurality of KLT transform kernels comprises one of following steps:

in response to determining that the KLT is not enabled, setting the first bin as 0;

in response to selecting the first KLT transform kernel as the KLT transform kernel, setting the first bin as 1 and the second bin as 1;

in response to selecting the second KLT transform kernel as the KLT transform kernel, setting the first bin as 1, the second bin as 0, and the third bin as 0; or

in response to selecting the third KLT transform kernel as the KLT transform kernel, setting the first bin as 1, the second bin as 0, and the third bin as 1.

23. The method of claim 18, wherein:

the plurality of KLT transform kernels comprise a first KLT transform kernel, a second KLT transform kernel, a third KLT transform kernel, and a fourth KLT transform kernel;

the binarized value comprises a first bin, a second bin, and a third bin;

the one or more contexts comprise a first context, a second context, and a third context;

the first bin is context coded using the first context;

the second bin is context coded using the second context; and

the third bin is context coded with the third context, or coded in a bypass mode.

24. The method of claim 23, wherein selecting, by the encoder, the KLT transform kernel for the current block in the intra mode from the plurality of KLT transform kernels comprises one of following steps:

in response to determining that the KLT is not enabled, setting the first bin as 0;

in response to selecting the first KLT transform kernel as the KLT transform kernel, setting the first bin as 1, the second bin as 1, and the third bin as 1;

in response to selecting the second KLT transform kernel as the KLT transform kernel, setting the first bin as 1, the second bin as 1, and the third bin as 0;

in response to selecting the third KLT transform kernel as the KLT transform kernel, setting the first bin as 1, the second bin as 0, and the third bin as 0; or

in response to selecting the fourth KLT transform kernel as the KLT transform kernel, setting the first bin as 1, the second bin as 0, and the third bin as 1.

25. An apparatus for video decoding, comprising:

one or more processors; and

a memory coupled to the one or more processors and configured to store instructions executable by the one or more processors, wherein the one or more processors, upon execution of the instructions are configured to perform the method in any one of claims 1-12.

26. An apparatus for video encoding, comprising:

one or more processors; and

a memory coupled to the one or more processors and configured to store instructions executable by the one or more processors, wherein the one or more processors, upon execution of the instructions are configured to perform the method in any one of claims 13-24.

27. A non-transitory computer readable storage medium for storing computer-executable instructions that, when executed by one or more computer processors, cause the one or more computer processors to perform the method in any one of claims 1-12.

28. A non-transitory computer readable storage medium for storing computer-executable instructions that, when executed by one or more computer processors, cause the one or more computer processors to perform the method in any one of claims 13-24.

29. A non-transitory computer-readable storage medium for storing a bitstream to be decoded by the method in any of claims 1-12.

30. A non-transitory computer-readable storage medium for storing a bitstream generated by the method in any of claims 13-24.

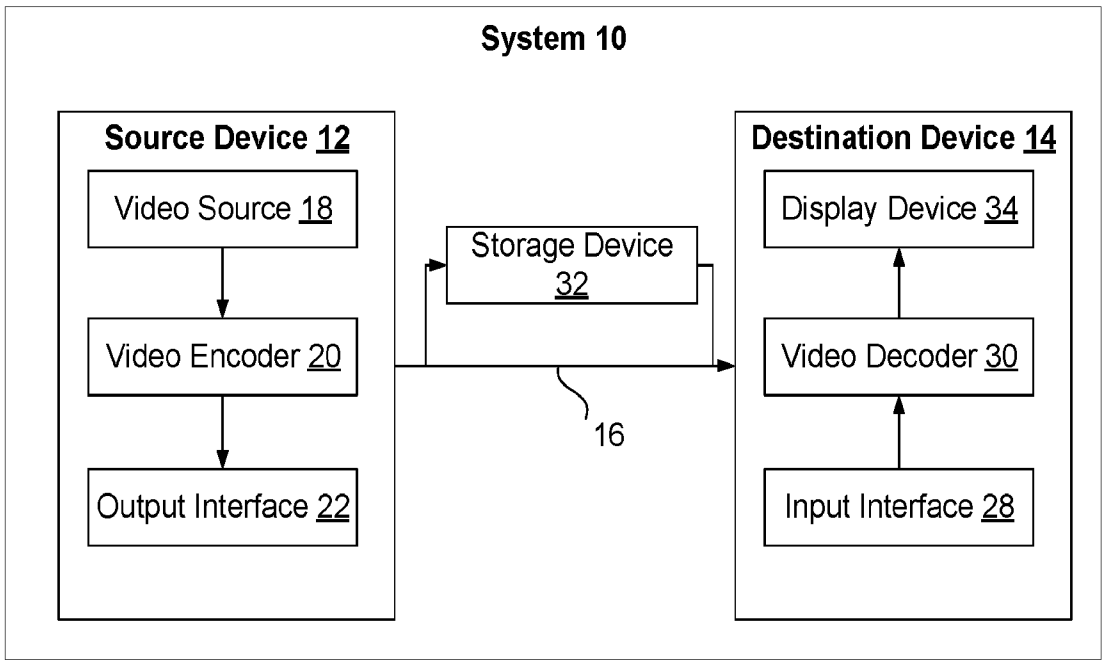
System 10
Source Device 12
Video Source 18
Video Encoder 20
Output Interface 22
Storage Device 32
16
Destination Device 14
Display Device 34
Video Decoder 30
Input Interface 28

FIG. 1

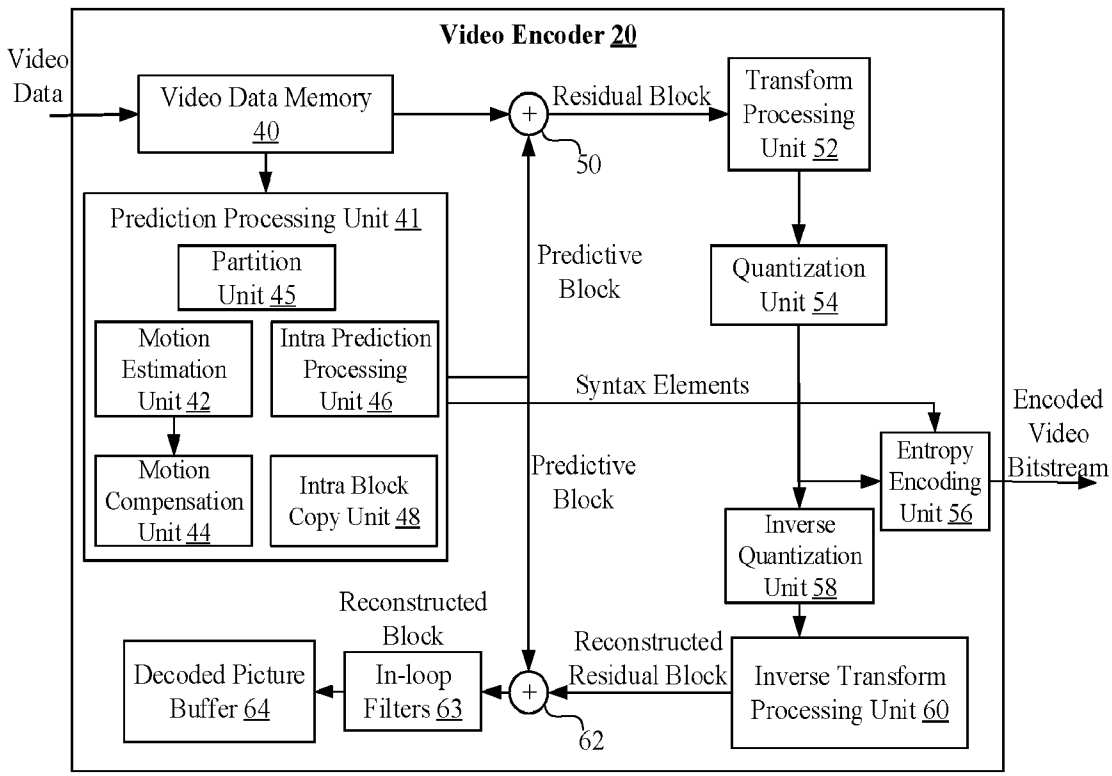
Video Encoder 20
Video Data
Video Data Memory 40
Residual Block
50
Transform Processing Unit 52
Prediction Processing Unit 41
Partition Unit 45
Motion Estimation Unit 42
Intra Prediction Processing Unit 46
Motion Compensation Unit 44
Intra Block Copy Unit 48
Predictive Block
Quantization Unit 54
Syntax Elements
Entropy Encoding Unit 56
Encoded Video Bitstream
Predictive Block
Inverse Quantization Unit 58
Reconstructed Block
Reconstructed Residual Block
Inverse Transform Processing Unit 60
Decoded Picture Buffer 64
In-loop Filters 63
62

FIG. 2

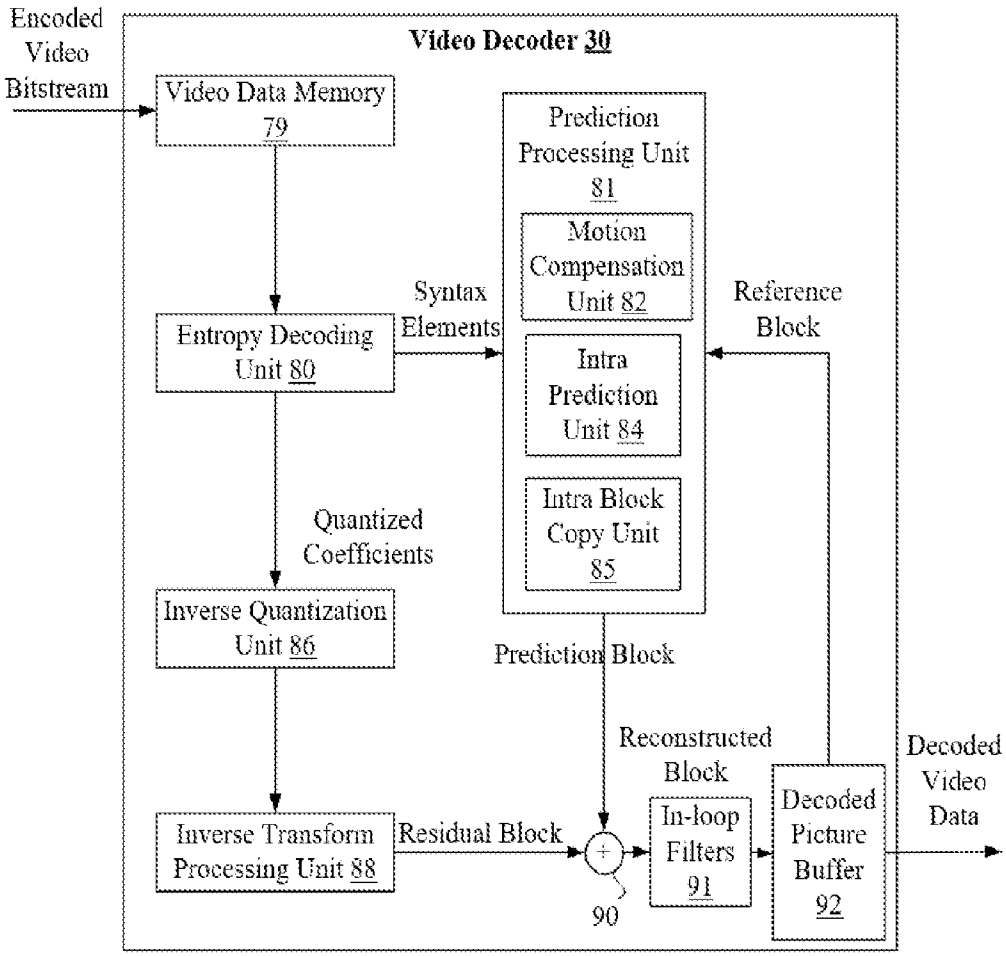
Encoded Video Bitstream
Video Decoder 30
Video Data Memory 79
Prediction Processing Unit 81
Motion Compensation Unit 82
Intra Prediction Unit 84
Intra Block Copy Unit 85
Entropy Decoding Unit 80
Syntax Elements
Reference Block
Quantized Coefficients
Inverse Quantization Unit 86
Prediction Block
Reconstructed Block
Inverse Transform Processing Unit 88
Residual Block
90
In-loop Filters 91
Decoded Picture Buffer 92
Decoded Video Data

FIG. 3A

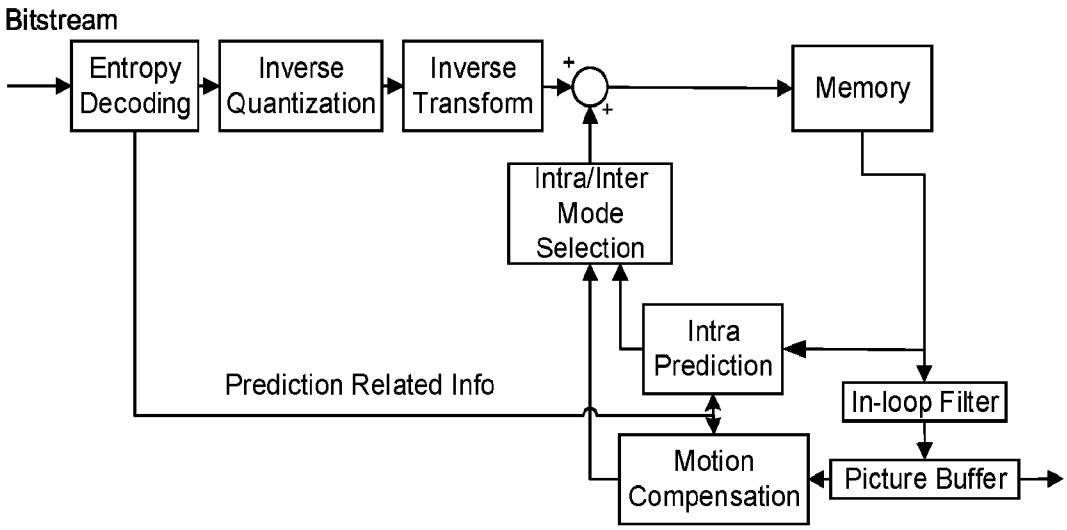
Bitstream
Entropy Decoding
Inverse Quantization
Inverse Transform
+
+
Memory
Intra/Inter Mode Selection
Intra Prediction
Prediction Related Info
In-loop Filter
Motion Compensation
Picture Buffer

FIG. 3B

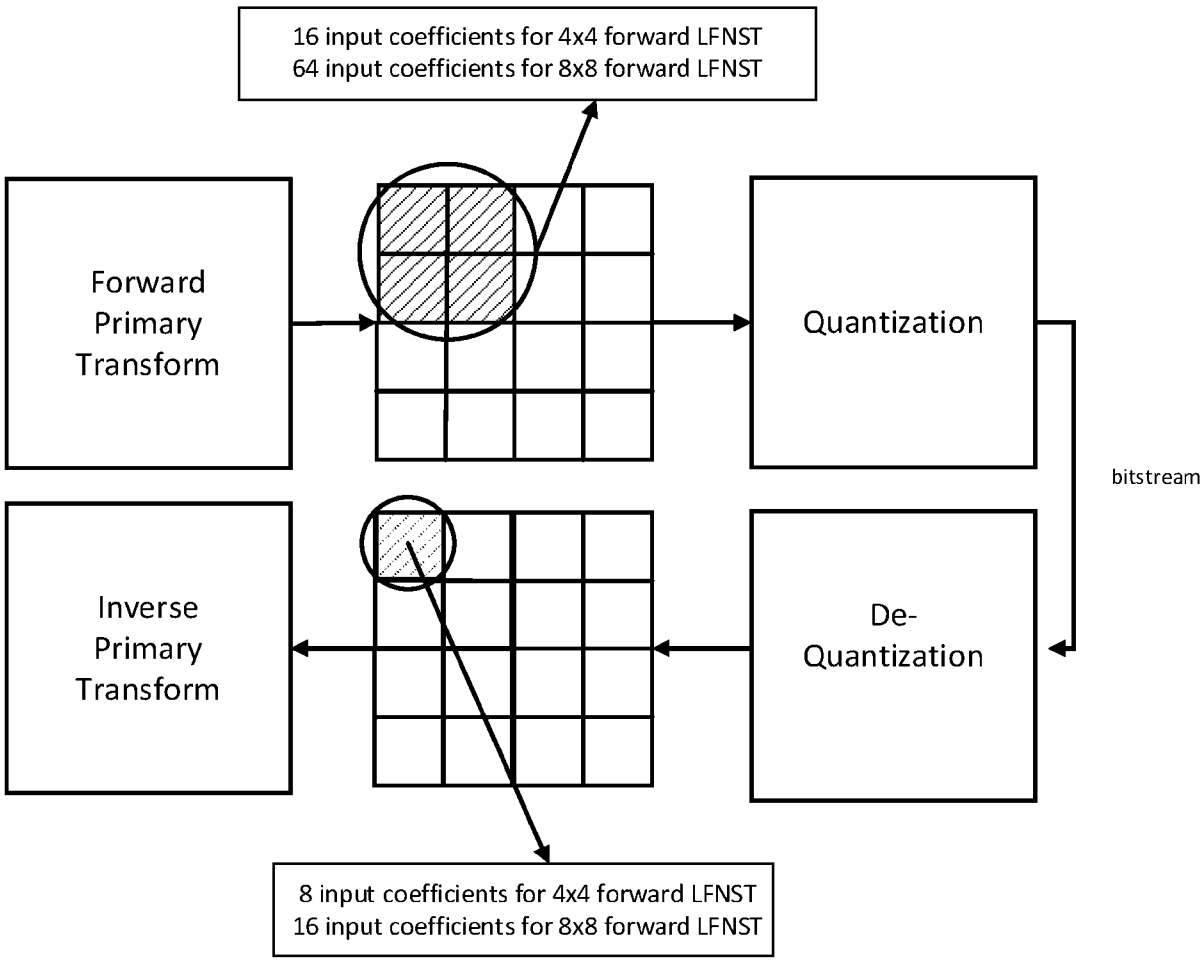
16 input coefficients for 4x4 forward LFNST
64 input coefficients for 8x8 forward LFNST
Forward Primary Transform
Quantization
bitstream
Inverse Primary Transform
De-Quantization
8 input coefficients for 4x4 forward LFNST
16 input coefficients for 8x8 forward LFNST

FIG. 3C

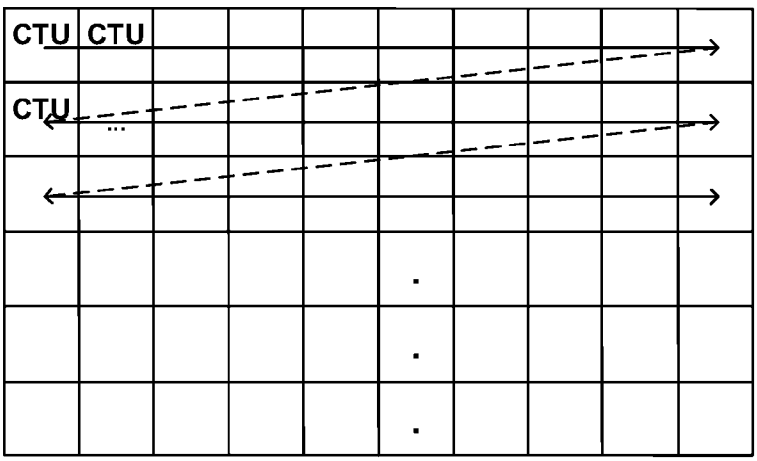
CTU
CTU
CTU
...

FIG. 4A

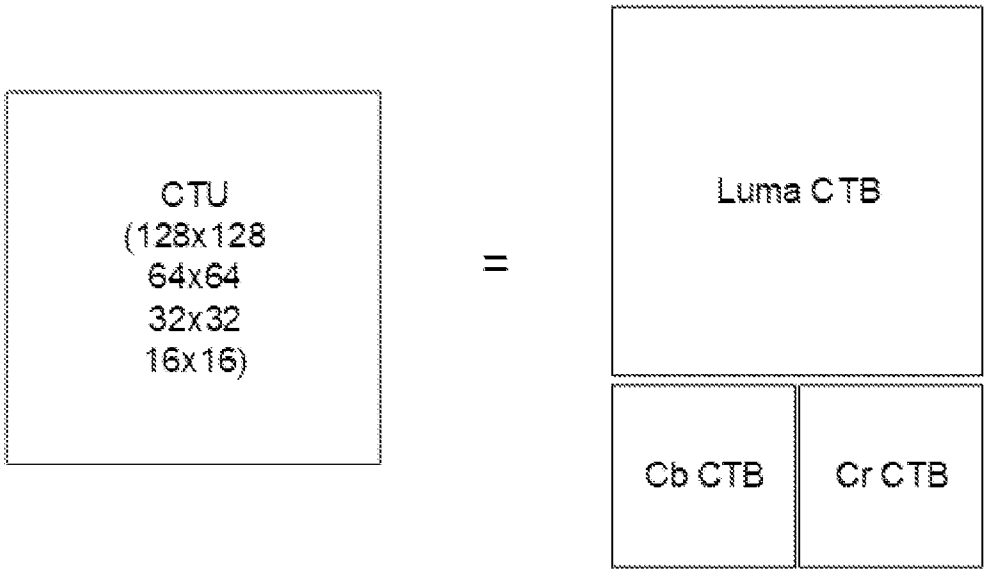

FIG. 4B

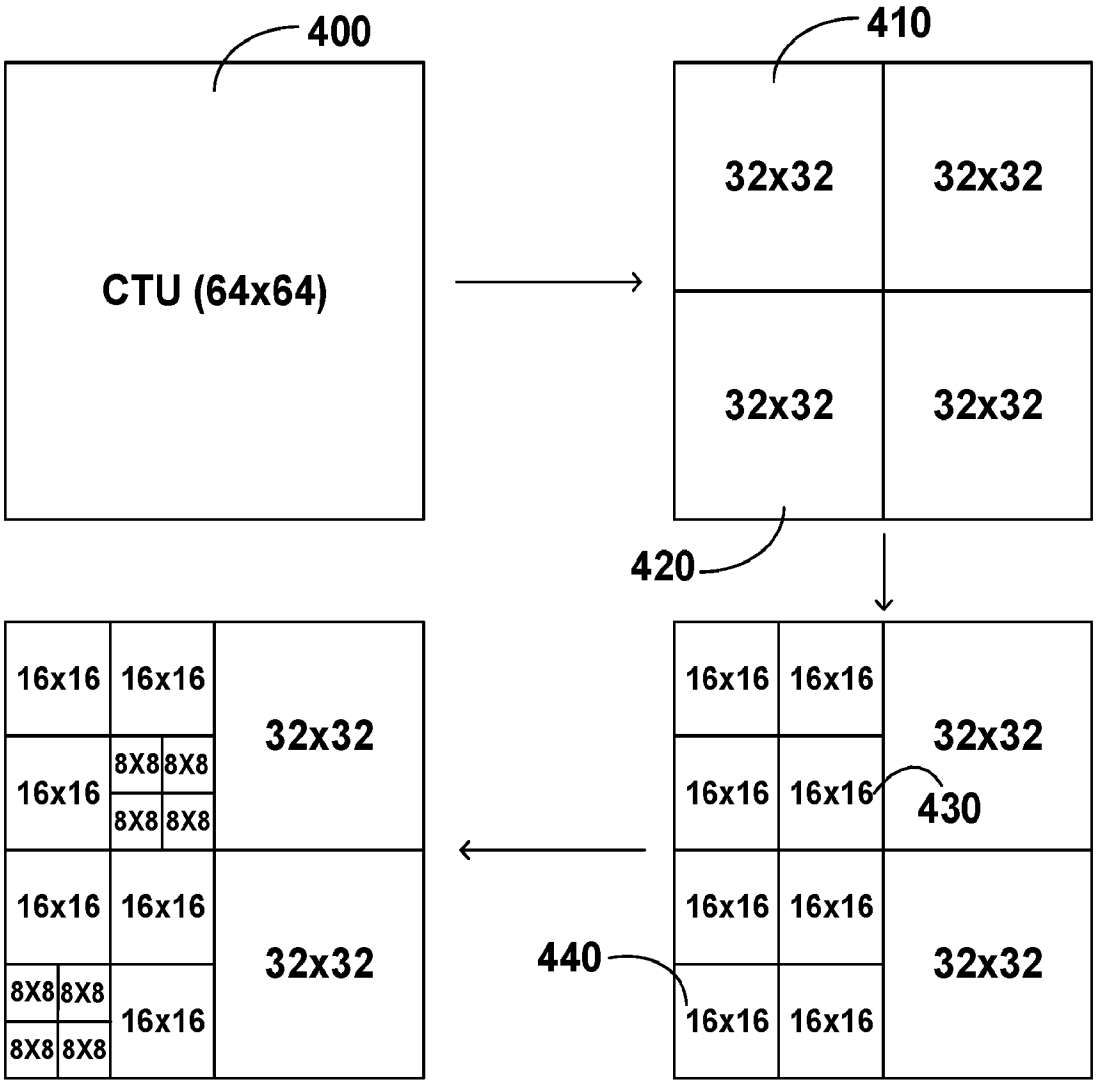
400
CTU (64x64)
410
32x32
32x32
32x32
32x32
420
16x16
16x16
32x32
16x16
8X8
8X8
8X8
8X8
16x16
16x16
32x32
8X8
8X8
8X8
8X8
16x16
16x16
16x16
32x32
16x16
16x16
16x16
16x16
32x32
16x16
16x16
430
440
16x16
16x16

FIG. 4C

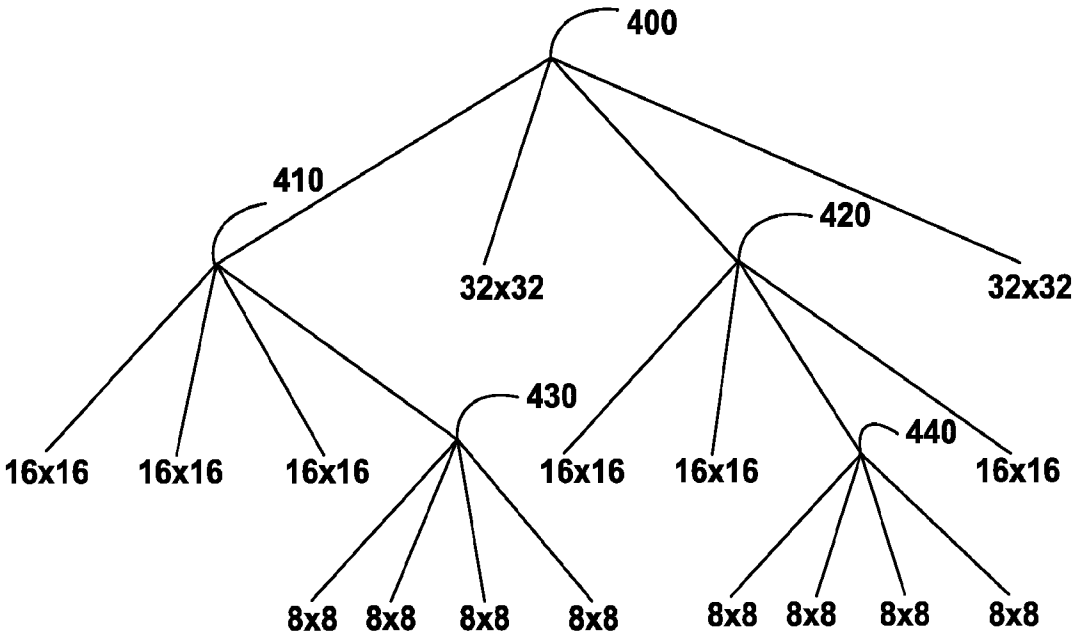
400
410
420
32x32
32x32
430
440
16x16
16x16
16x16
16x16
16x16
16x16
8x8
8x8
8x8
8x8
8x8
8x8
8x8
8x8

FIG. 4D

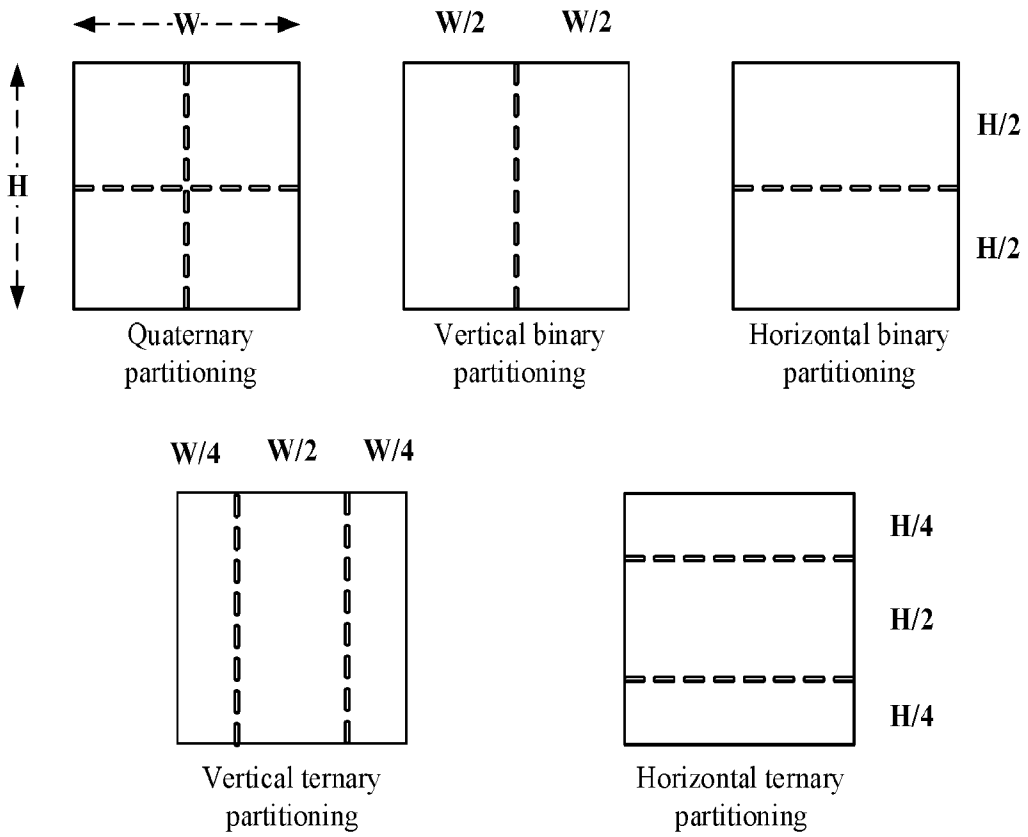
W
H
W/2
W/2
H/2
H/2
Quaternary partitioning
Vertical binary partitioning
Horizontal binary partitioning
W/4
W/2
W/4
H/4
H/2
H/4
Vertical ternary partitioning
Horizontal ternary partitioning

FIG. 4E

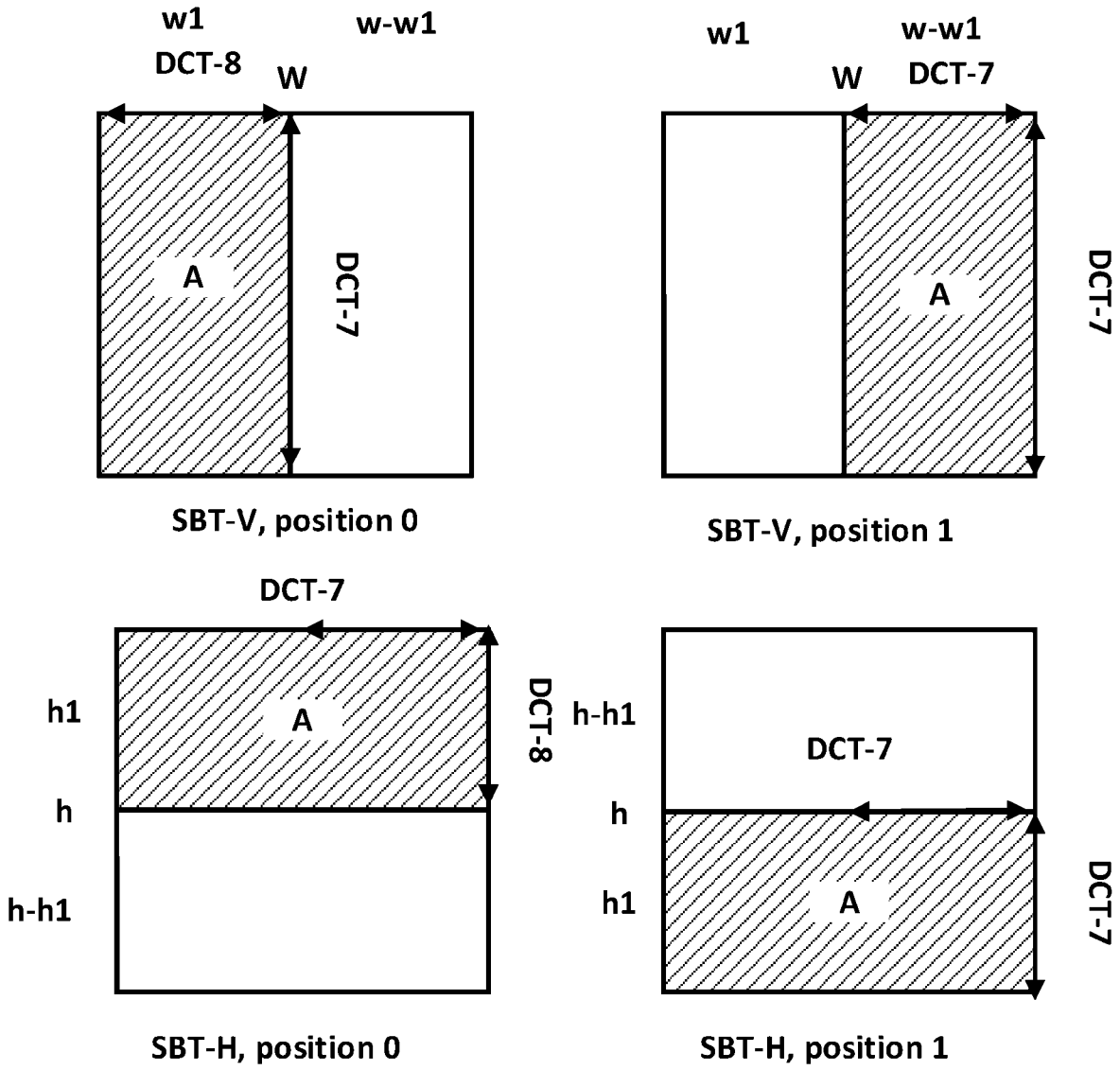
w1
w-w1
DCT-8
W
A
DCT-7
SBT-V, position 0
w1
w-w1
W
DCT-7
A
DCT-7
SBT-V, position 1
DCT-7
h1
A
DCT-8
h
h-h1
SBT-H, position 0
h-h1
DCT-7
h
h1
A
DCT-7
SBT-H, position 1

FIG. 5

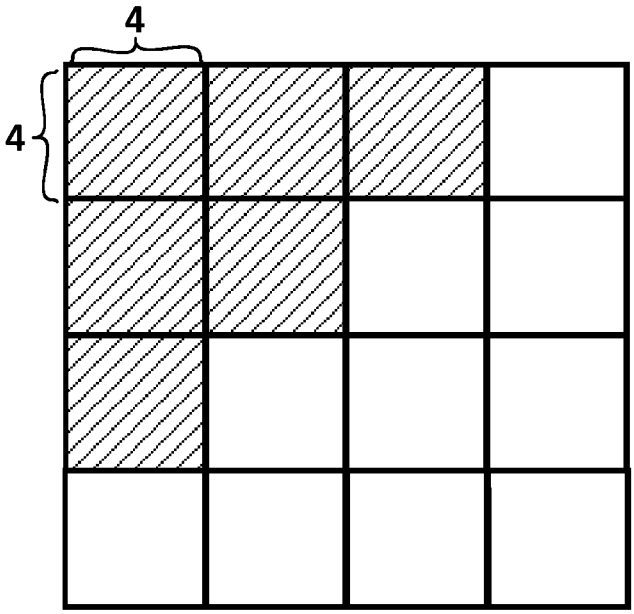

FIG. 6

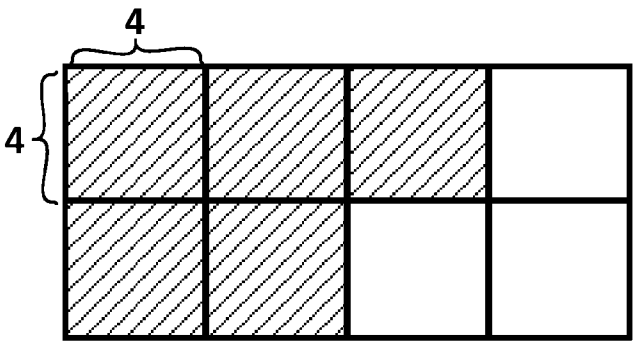

FIG. 7

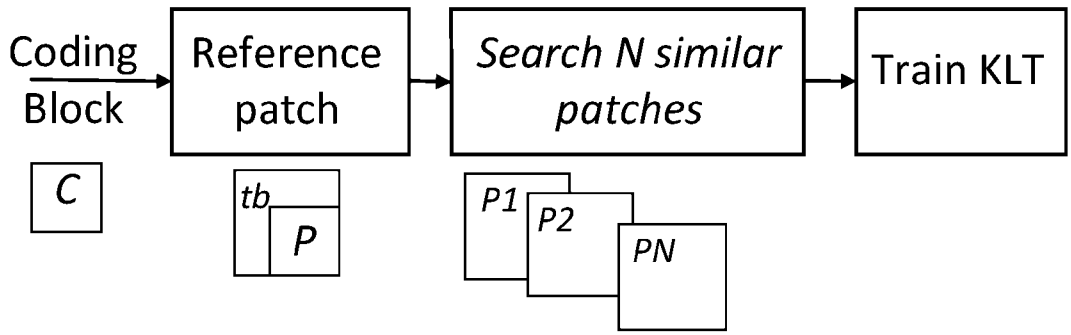

FIG. 8

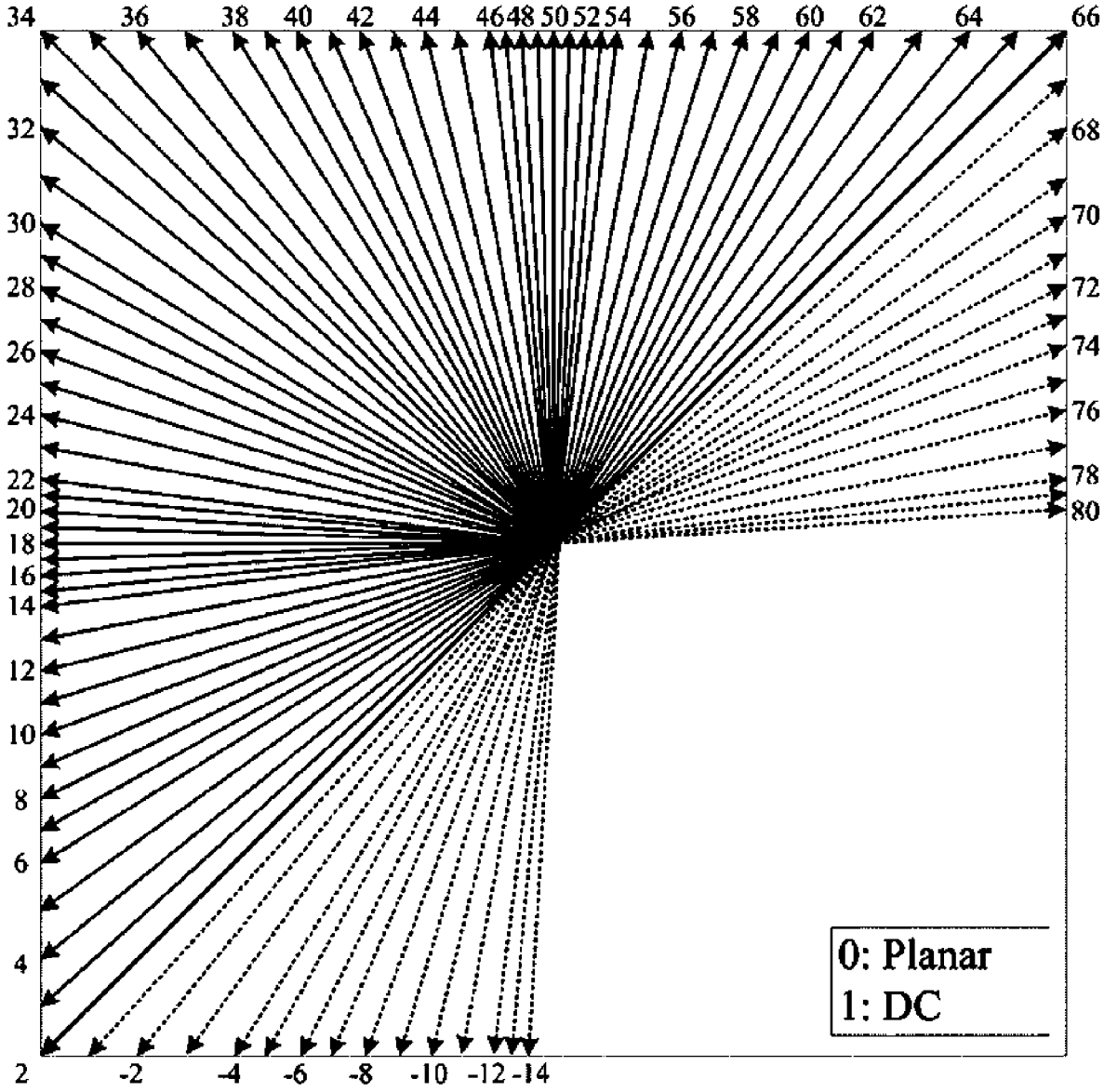
34
36
38
40
42
44
46
48
50
52
54
56
58
60
62
64
66
32
30
28
26
24
22
20
18
16
14
12
10
8
6
4
2
68
70
72
74
76
78
80
-2
-4
-6
-8
-10
-12
-14
0: Planar
1: DC

FIG. 9

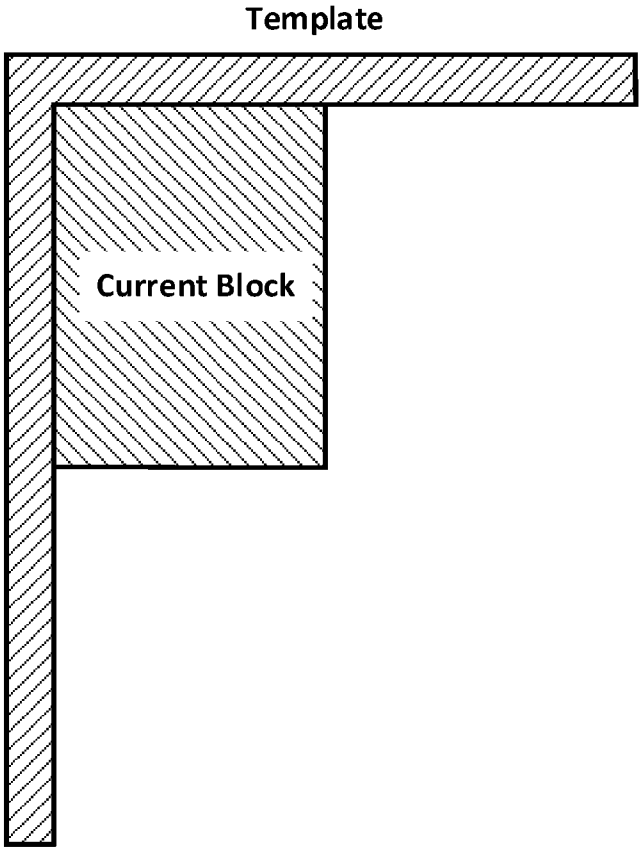

FIG. 10

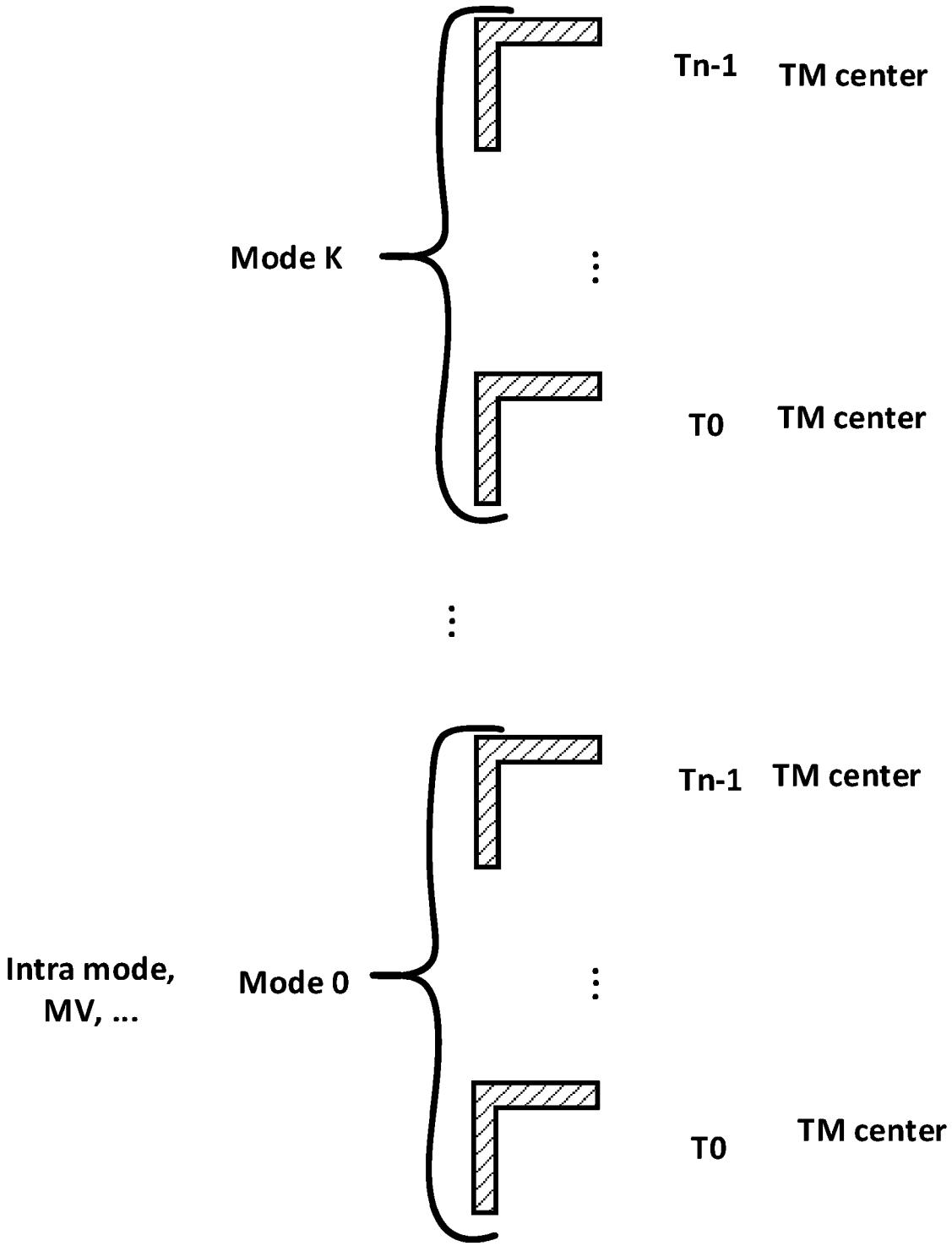
Tn-1
TM center
Mode K
T0
TM center
Tn-1
TM center
Intra mode,
MV, ...
Mode 0
T0
TM center

FIG. 11

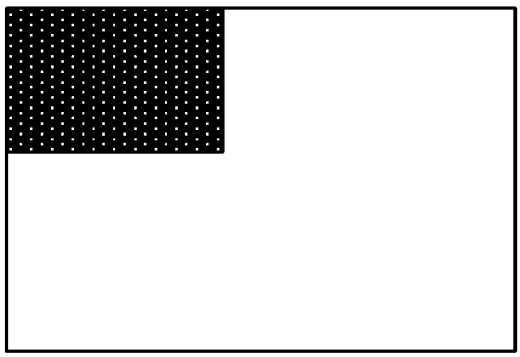

FIG. 12A

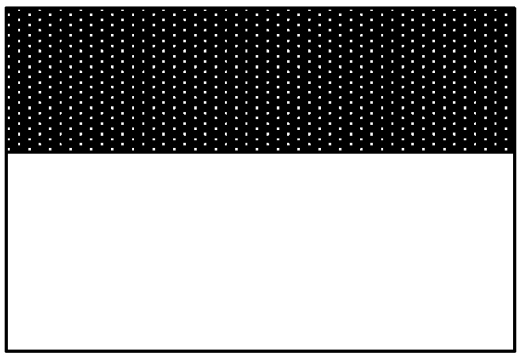

FIG. 12B

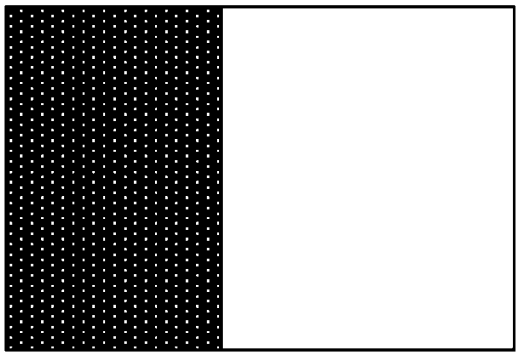

FIG. 12C

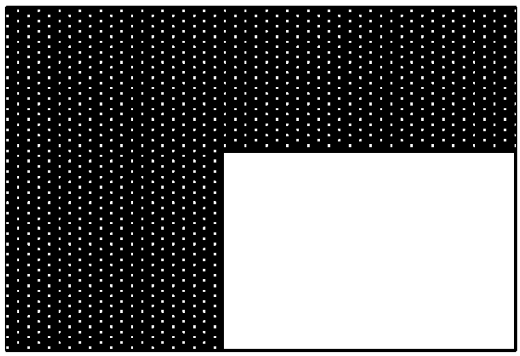

FIG. 12D

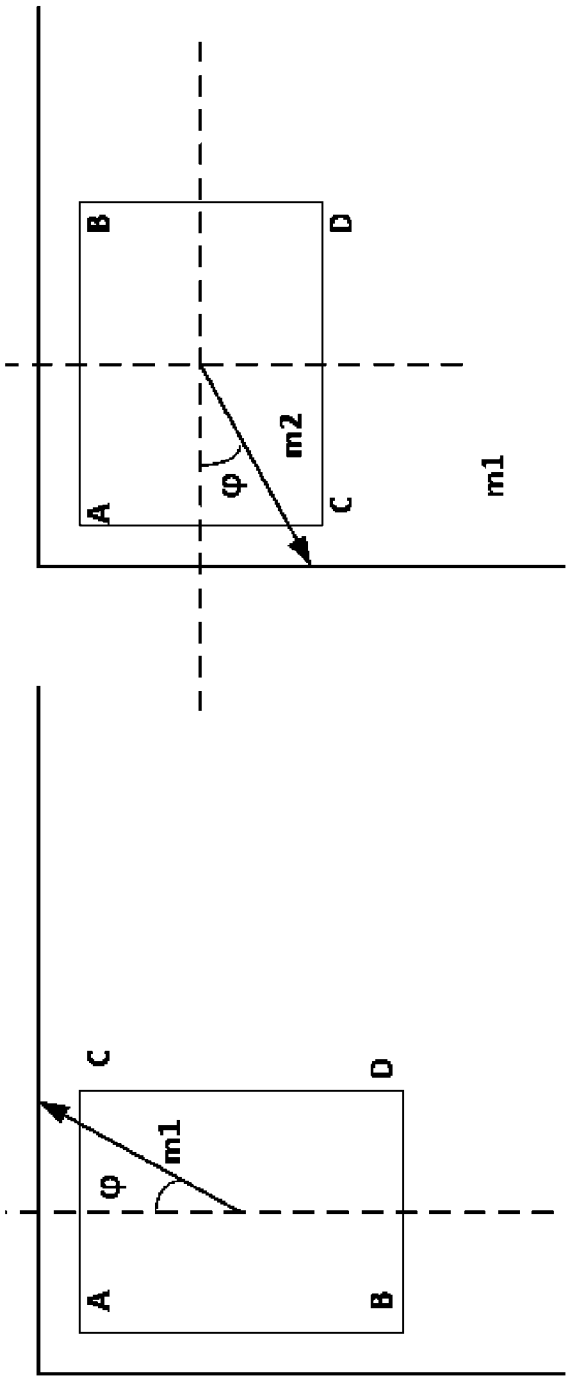

FIG. 13

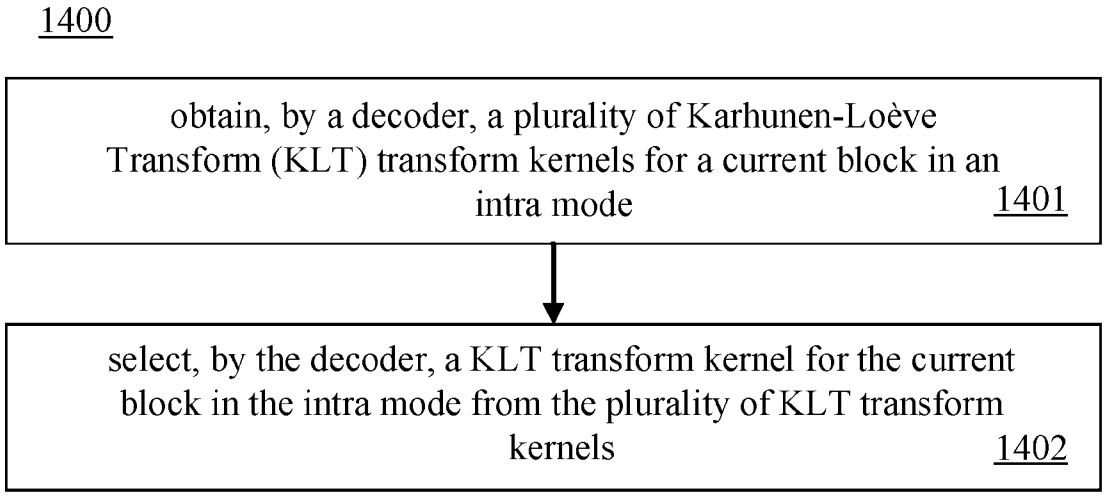
1400
obtain, by a decoder, a plurality of Karhunen-Loève Transform (KLT) transform kernels for a current block in an intra mode
1401
select, by the decoder, a KLT transform kernel for the current block in the intra mode from the plurality of KLT transform kernels
1402

FIG. 14

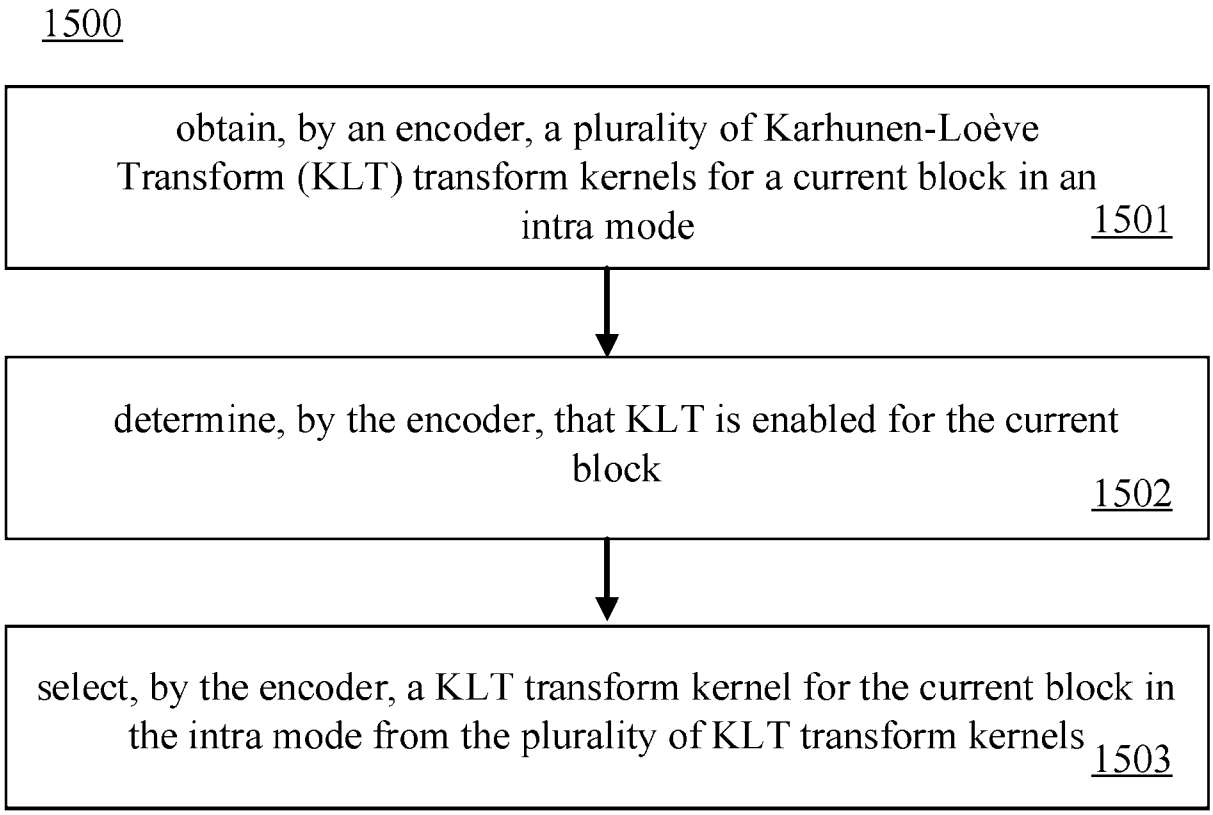
1500
obtain, by an encoder, a plurality of Karhunen-Loève Transform (KLT) transform kernels for a current block in an intra mode
1501
determine, by the encoder, that KLT is enabled for the current block
1502
select, by the encoder, a KLT transform kernel for the current block in the intra mode from the plurality of KLT transform kernels
1503

FIG. 15

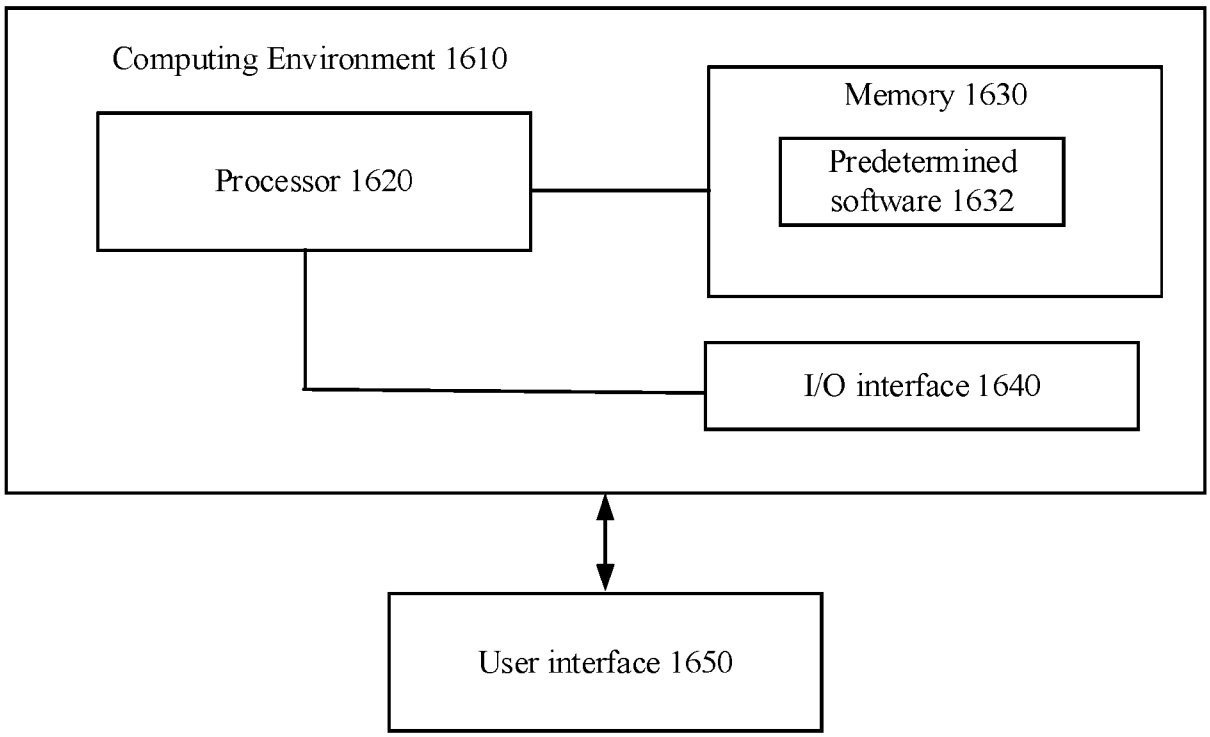
Computing Environment 1610
Memory 1630
Processor 1620
Predetermined software 1632
I/O interface 1640
User interface 1650

FIG. 16

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2024/011479

A. CLASSIFICATION OF SUBJECT MATTER

H04N 19/122(2014.01)i; **H04N 19/61**(2014.01)i; **H04N 19/157**(2014.01)i; **H04N 19/70**(2014.01)i; **H04N 19/176**(2014.01)i

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 19/122(2014.01); H04N 19/12(2014.01); H04N 19/176(2014.01); H04N 19/61(2014.01)

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

Korean utility models and applications for utility models
Japanese utility models and applications for utility models

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

eKOMPASS(KIPO internal) & Keywords: intra prediction, Karhunen-Loeve Transform, transform kernel, flag, index, bin

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	MOONMO KOO et al., "Non-EE2: Separable KLT for intra coding", Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29, [Document: JVET-AB0100-v2, (version 2)], 28th Meeting: Mainz, DE, pp. 1-2, 21 October 2022 [retrieved on: 2024-04-18], Retrieved from: <https://jvet-experts.org/>	1-3,6-8,13-15, 18-20,25-30
A	pages 1-2	4-5,9-12,16-17,21-24
A	KR 10-2020-0035388 A (LG ELECTRONICS INC.) 03 April 2020 (2020-04-03) claims 2-3	1-30
A	KR 10-2010-0131934 A (LG ELECTRONICS INC.) 16 December 2010 (2010-12-16) claims 6-7	1-30
A	WO 2022-069331 A1 (INTERDIGITAL VC HOLDINGS FRANCE, SAS) 07 April 2022 (2022-04-07) paragraphs [0017]-[0064]; and figures 1-9	1-30

☑ 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
"D" document cited by the applicant in the international application
"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	Date of mailing of the international search report
08 May 2024	**27 May 2024**

Name and mailing address of the ISA/KR	Authorized officer
Korean Intellectual Property Office **189 Cheongsa-ro, Seo-gu, Daejeon** **35208, Republic of Korea**	**KIM, Sung Hoon**
Facsimile No. **+82-42-481-8578**	Telephone No. **+82-42-481-8710**

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2024/011479

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	HAOMING CHEN et al., "New Transforms Tightly Bounded by DCT and KLT," IEEE Signal Processing Letters, vol. 19, no. 6, pp. 344-347, 18 April 2012 [retrieved on: 2024-04-18], Retrieved from: <https://ieeexplore.ieee.org/document/6186780> pages 344-347	1-30

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/US2024/011479

Patent document cited in search report	Publication date (day/month/year)	Patent family member(s)	Publication date (day/month/year)
KR 10-2020-0035388 A	03 April 2020	EP 2154898 A2	17 February 2010
		EP 2154898 A3	02 June 2010
		EP 2373031 A1	05 October 2011
		KR 10-1805531 B1	10 January 2018
		KR 10-1994216 B1	28 June 2019
		KR 10-2010-0020441 A	22 February 2010
		KR 10-2010-0131934 A	16 December 2010
		KR 10-2017-0134300 A	06 December 2017
		KR 10-2019-0075031 A	28 June 2019
		KR 10-2096407 B1	02 April 2020
		KR 10-2213876 B1	05 February 2021
		US 10015519 B2	03 July 2018
		US 10405001 B2	03 September 2019
		US 10986372 B2	20 April 2021
		US 2010-0061454 A1	11 March 2010
		US 2012-0163469 A1	28 June 2012
		US 2015-0249840 A1	03 September 2015
		US 2017-0188050 A1	29 June 2017
		US 2018-0278961 A1	27 September 2018
		US 2019-0349605 A1	14 November 2019
		US 2021-0211726 A1	08 July 2021
		US 8687692 B2	01 April 2014
		US 9100648 B2	04 August 2015
		US 9635368 B2	25 April 2017
		WO 2010-018992 A2	18 February 2010
		WO 2010-018992 A3	03 June 2010
		WO 2010-143853 A2	16 December 2010
		WO 2010-143853 A3	24 February 2011
KR 10-2010-0131934 A	16 December 2010	EP 2154898 A2	17 February 2010
		EP 2154898 A3	02 June 2010
		EP 2373031 A1	05 October 2011
		KR 10-1805531 B1	10 January 2018
		KR 10-1994216 B1	28 June 2019
		KR 10-2010-0020441 A	22 February 2010
		KR 10-2017-0134300 A	06 December 2017
		KR 10-2019-0075031 A	28 June 2019
		KR 10-2020-0035388 A	03 April 2020
		KR 10-2096407 B1	02 April 2020
		KR 10-2213876 B1	05 February 2021
		US 10015519 B2	03 July 2018
		US 10405001 B2	03 September 2019
		US 10986372 B2	20 April 2021
		US 2010-0061454 A1	11 March 2010
		US 2012-0163469 A1	28 June 2012
		US 2015-0249840 A1	03 September 2015
		US 2017-0188050 A1	29 June 2017
		US 2018-0278961 A1	27 September 2018
		US 2019-0349605 A1	14 November 2019
		US 2021-0211726 A1	08 July 2021
		US 8687692 B2	01 April 2014

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/US2024/011479

Patent document cited in search report	Publication date (day/month/year)	Patent family member(s)	Publication date (day/month/year)
		US 9100648 B2	04 August 2015
		US 9635368 B2	25 April 2017
		WO 2010-018992 A2	18 February 2010
		WO 2010-018992 A3	03 June 2010
		WO 2010-143853 A2	16 December 2010
		WO 2010-143853 A3	24 February 2011
WO 2022-069331 A1	07 April 2022	CN 116018805 A	25 April 2023
		EP 4222955 A1	09 August 2023
		US 2024-0031606 A1	25 January 2024