



(51) International Patent Classification:

H04N 19/18 (2014.01)

94043 (US). ZHAO, Xin; 1172 Fairford Way, San Jose, CA

95129 (US). LIU, Shan; 1155 Nevada Avenue, San Jose, CA 95125 (US).

(21) International Application Number:

PCT/US2022/014248

(74) Agent: CHEN, Gang; Crowell & Moring LLP, P.O. Box 10087, Chicago, IL 60610 (US).

(22) International Filing Date:

28 January 2022 (28.01.2022)

(25) Filing Language:

English

(26) Publication Language:

English

(30) Priority Data:

63/175,897 16 April 2021 (16.04.2021) US

17/568,275 04 January 2022 (04.01.2022) US

(71) Applicant: TENCENT AMERICA LLC [US/US]; 2747

Park Boulevard, Palo Alto, CA 94306 (US).

(72) Inventors: PERINGASSERY KRISHNAN, Madhu; 100

North Whisman Road, Apt. 3615, Mountain View, CA

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

(54) Title: INTERACTION BETWEEN TRANSFORM PARTITIONING AND PRIMARY/SECONDARY TRANSFORM TYPE SELECTION

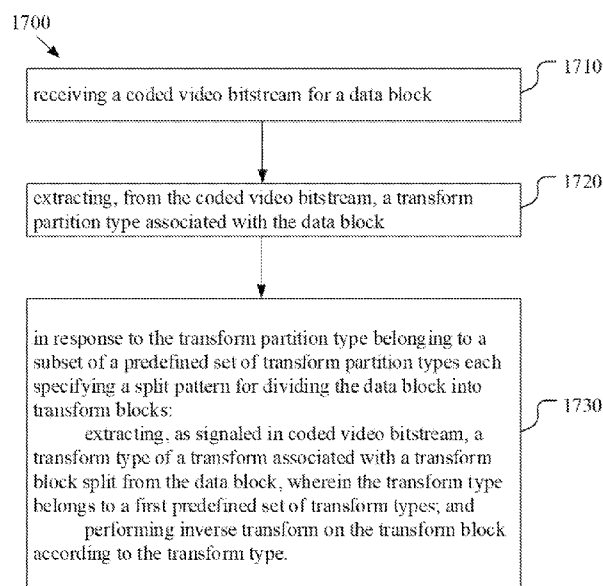


FIG. 17

(57) Abstract: Methods, apparatus, and computer readable storage medium for for encoding/decoding video data in a decoder. The method includes receiving a coded video bitstream for a data block; extracting, from the coded video bitstream, a transform partition type associated with the data block; and in response to the transform partition type belonging to a subset of a predefined set of transform partition types each specifying a split pattern for dividing the data block into transform blocks: extracting, as signaled in coded video bitstream, a transform type of a transform associated with a transform block split from the data block, wherein the transform type belongs to a first predefined set of transform types; and performing inverse transform on the transform block according to the transform type.

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

Published:

— *with international search report (Art. 21(3))*

INTERACTION BETWEEN TRANSFORM PARTITIONING AND PRIMARY/SECONDARY TRANSFORM TYPE SELECTION

INCORPORATION BY REFERENCE

[0001] This application is based on and claims the benefit of priority to U.S. Provisional Application No. 63/175,897 filed on April 16, 2021 and U.S. non-Provisional Application No. 17/568,275 filed on January 4, 2022, both of which are herein incorporated by reference in their entireties.

TECHNICAL FIELD

[0002] This disclosure describes a set of advanced video coding technologies. More specifically, the disclosed technology involves interactions between transform partitioning schemes and primary/secondary transform type selection in video encoding and decoding.

BACKGROUND

[0003] This background description provided herein is for the purpose of generally presenting the context of the disclosure. Work of the presently named inventors, to the extent the work is described in this background section, as well as aspects of the description that may not otherwise qualify as prior art at the time of filing of this application, are neither expressly nor impliedly admitted as prior art against the present disclosure.

[0004] Video coding and decoding can be performed using inter-picture prediction with motion compensation. Uncompressed digital video can include a series of pictures, with each picture having a spatial dimension of, for example, 1920 x 1080 luminance samples and associated full or subsampled chrominance samples. The series of pictures can have a fixed or variable picture rate (alternatively referred to as frame rate) of, for example, 60 pictures per second or 60 frames per second. Uncompressed video has specific bitrate requirements for streaming or data processing. For example, video with a pixel resolution of 1920 x 1080, a frame rate of 60 frames/second, and a chroma subsampling of 4:2:0 at 8 bit per pixel per color channel requires close to 1.5 Gbit/s bandwidth. An hour of such video requires more than 600 GBytes of storage space.

[0005] One purpose of video coding and decoding can be the reduction of redundancy in the uncompressed input video signal, through compression. Compression can help reduce the aforementioned bandwidth and/or storage space requirements, in some cases, by two orders of magnitude or more. Both lossless compression and lossy compression, as well as a

combination thereof can be employed. Lossless compression refers to techniques where an exact copy of the original signal can be reconstructed from the compressed original signal via a decoding process. Lossy compression refers to coding/decoding process where original video information is not fully retained during coding and not fully recoverable during decoding. When using lossy compression, the reconstructed signal may not be identical to the original signal, but the distortion between original and reconstructed signals is made small enough to render the reconstructed signal useful for the intended application albeit some information loss. In the case of video, lossy compression is widely employed in many applications. The amount of tolerable distortion depends on the application. For example, users of certain consumer video streaming applications may tolerate higher distortion than users of cinematic or television broadcasting applications. The compression ratio achievable by a particular coding algorithm can be selected or adjusted to reflect various distortion tolerance: higher tolerable distortion generally allows for coding algorithms that yield higher losses and higher compression ratios.

[0006] A video encoder and decoder can utilize techniques from several broad categories and steps, including, for example, motion compensation, Fourier transform, quantization, and entropy coding.

[0007] Video codec technologies can include techniques known as intra coding. In intra coding, sample values are represented without reference to samples or other data from previously reconstructed reference pictures. In some video codecs, a picture is spatially subdivided into blocks of samples. When all blocks of samples are coded in intra mode, that picture can be referred to as an intra picture. Intra pictures and their derivatives such as independent decoder refresh pictures, can be used to reset the decoder state and can, therefore, be used as the first picture in a coded video bitstream and a video session, or as a still image. The samples of a block after intra prediction can then be subject to a transform into frequency domain, and the transform coefficients so generated can be quantized before entropy coding. Intra prediction represents a technique that minimizes sample values in the pre-transform domain. In some cases, the smaller the DC value after a transform is, and the smaller the AC coefficients are, the fewer the bits that are required at a given quantization step size to represent the block after entropy coding.

[0008] Traditional intra coding such as that known from, for example, MPEG-2 generation coding technologies, does not use intra prediction. However, some newer video compression technologies include techniques that attempt coding/decoding of blocks based on, for example, surrounding sample data and/or metadata that are obtained during the

encoding and/or decoding of spatially neighboring, and that precede in decoding order the blocks of data being intra coded or decoded. Such techniques are henceforth called “intra prediction” techniques. Note that in at least some cases, intra prediction uses reference data only from the current picture under reconstruction and not from other reference pictures.

[0009] There can be many different forms of intra prediction. When more than one of such techniques are available in a given video coding technology, the technique in use can be referred to as an intra prediction mode. One or more intra prediction modes may be provided in a particular codec. In certain cases, modes can have submodes and/or may be associated with various parameters, and mode/submode information and intra coding parameters for blocks of video can be coded individually or collectively included in mode codewords. Which codeword to use for a given mode, submode, and/or parameter combination can have an impact in the coding efficiency gain through intra prediction, and so can the entropy coding technology used to translate the codewords into a bitstream.

[0010] A certain mode of intra prediction was introduced with H.264, refined in H.265, and further refined in newer coding technologies such as joint exploration model (JEM), versatile video coding (VVC), and benchmark set (BMS). Generally, for intra prediction, a predictor block can be formed using neighboring sample values that have become available. For example, available values of particular set of neighboring samples along certain direction and/or lines may be copied into the predictor block. A reference to the direction in use can be coded in the bitstream or may itself be predicted.

[0011] Referring to FIG. 1A, depicted in the lower right is a subset of nine predictor directions specified in H.265’s 33 possible intra predictor directions (corresponding to the 33 angular modes of the 35 intra modes specified in H.265). The point where the arrows converge (101) represents the sample being predicted. The arrows represent the direction from which neighboring samples are used to predict the sample at 101. For example, arrow (102) indicates that sample (101) is predicted from a neighboring sample or samples to the upper right, at a 45 degree angle from the horizontal direction. Similarly, arrow (103) indicates that sample (101) is predicted from a neighboring sample or samples to the lower left of sample (101), in a 22.5 degree angle from the horizontal direction.

[0012] Still referring to FIG. 1A, on the top left there is depicted a square block (104) of 4 x 4 samples (indicated by a dashed, boldface line). The square block (104) includes 16 samples, each labelled with an “S”, its position in the Y dimension (e.g., row index) and its position in the X dimension (e.g., column index). For example, sample S21 is the second sample in the Y dimension (from the top) and the first (from the left) sample in the X

dimension. Similarly, sample S44 is the fourth sample in block (104) in both the Y and X dimensions. As the block is 4 x 4 samples in size, S44 is at the bottom right. Further shown are example reference samples that follow a similar numbering scheme. A reference sample is labelled with an R, its Y position (e.g., row index) and X position (column index) relative to block (104). In both H.264 and H.265, prediction samples adjacently neighboring the block under reconstruction are used.

[0013] Intra picture prediction of block 104 may begin by copying reference sample values from the neighboring samples according to a signaled prediction direction. For example, assuming that the coded video bitstream includes signaling that, for this block 104, indicates a prediction direction of arrow (102)—that is, samples are predicted from a prediction sample or samples to the upper right, at a 45-degree angle from the horizontal direction. In such a case, samples S41, S32, S23, and S14 are predicted from the same reference sample R05. Sample S44 is then predicted from reference sample R08.

[0014] In certain cases, the values of multiple reference samples may be combined, for example through interpolation, in order to calculate a reference sample; especially when the directions are not evenly divisible by 45 degrees.

[0015] The number of possible directions has increased as video coding technology has continued to develop. In H.264 (year 2003), for example, nine different direction are available for intra prediction. That increased to 33 in H.265 (year 2013), and JEM/VVC/BMS, at the time of this disclosure, can support up to 65 directions. Experimental studies have been conducted to help identify the most suitable intra prediction directions, and certain techniques in the entropy coding may be used to encode those most suitable directions in a small number of bits, accepting a certain bit penalty for directions. Further, the directions themselves can sometimes be predicted from neighboring directions used in the intra prediction of the neighboring blocks that have been decoded.

[0016] FIG. 1B shows a schematic (180) that depicts 65 intra prediction directions according to JEM to illustrate the increasing number of prediction directions in various encoding technologies developed over time.

[0017] The manner for mapping of bits representing intra prediction directions to the prediction directions in the coded video bitstream may vary from video coding technology to video coding technology; and can range, for example, from simple direct mappings of prediction direction to intra prediction mode, to codewords, to complex adaptive schemes involving most probable modes, and similar techniques. In all cases, however, there can be certain directions for intra prediction that are statistically less likely to occur in video content

than certain other directions. As the goal of video compression is the reduction of redundancy, those less likely directions will, in a well-designed video coding technology, may be represented by a larger number of bits than more likely directions.

[0018] Inter picture prediction, or inter prediction may be based on motion compensation. In motion compensation, sample data from a previously reconstructed picture or part thereof (reference picture), after being spatially shifted in a direction indicated by a motion vector (MV henceforth), may be used for a prediction of a newly reconstructed picture or picture part (e.g., a block). In some cases, the reference picture can be the same as the picture currently under reconstruction. MVs may have two dimensions X and Y, or three dimensions, with the third dimension being an indication of the reference picture in use (akin to a time dimension).

[0019] In some video compression techniques, a current MV applicable to a certain area of sample data can be predicted from other MVs, for example from those other MVs that are related to other areas of the sample data that are spatially adjacent to the area under reconstruction and precede the current MV in decoding order. Doing so can substantially reduce the overall amount of data required for coding the MVs by relying on removing redundancy in correlated MVs, thereby increasing compression efficiency. MV prediction can work effectively, for example, because when coding an input video signal derived from a camera (known as natural video) there is a statistical likelihood that areas larger than the area to which a single MV is applicable move in a similar direction in the video sequence and, therefore, can in some cases be predicted using a similar motion vector derived from MVs of neighboring area. That results in the actual MV for a given area to be similar or identical to the MV predicted from the surrounding MVs. Such an MV in turn may be represented, after entropy coding, in a smaller number of bits than what would be used if the MV is coded directly rather than predicted from the neighboring MV(s). In some cases, MV prediction can be an example of lossless compression of a signal (namely: the MVs) derived from the original signal (namely: the sample stream). In other cases, MV prediction itself can be lossy, for example because of rounding errors when calculating a predictor from several surrounding MVs.

[0020] Various MV prediction mechanisms are described in H.265/HEVC (ITU-T Rec. H.265, "High Efficiency Video Coding", December 2016). Out of the many MV prediction mechanisms that H.265 specifies, described below is a technique henceforth referred to as "spatial merge".

[0021] Specifically, referring to FIG. 2, a current block (201) comprises samples that have been found by the encoder during the motion search process to be predictable from a previous block of the same size that has been spatially shifted. Instead of coding that MV directly, the MV can be derived from metadata associated with one or more reference pictures, for example from the most recent (in decoding order) reference picture, using the MV associated with either one of five surrounding samples, denoted A0, A1, and B0, B1, B2 (202 through 206, respectively). In H.265, the MV prediction can use predictors from the same reference picture that the neighboring block uses.

SUMMARY

[0022] The present disclosure describes various embodiments of methods, apparatus, and computer-readable storage medium for video encoding and/or decoding.

[0023] According to one aspect, an embodiment of the present disclosure provides a method for encoding/decoding video data in a decoder. The method includes receiving a coded video bitstream for a data block; extracting, from the coded video bitstream, a transform partition type associated with the data block; and in response to the transform partition type belonging to a subset of a predefined set of transform partition types each specifying a split pattern for dividing the data block into transform blocks: extracting, as signaled in coded video bitstream, a transform type of a transform associated with a transform block split from the data block, wherein the transform type belongs to a first predefined set of transform types; and performing inverse transform on the transform block according to the transform type.

[0024] According to another aspect, an embodiment of the present disclosure provides a method for encoding/decoding video data. The method includes receiving a coded video bitstream for a data block; extracting, from the coded video bitstream, a transform partition type associated with a data block of the video data; in response to the transform partition type belonging to a subset of a predefined set of transform partition types, extracting, from the coded video bitstream, a transform type associated with a transform block of the data block; and in response to the transform partition type not belonging to the predefined set of transform partition types, identifying the transform type for the data block by default.

[0025] According to another aspect, an embodiment of the present disclosure provides a method for encoding/decoding video data. The method includes receiving a coded video bitstream for a data block; extracting, from the coded video bitstream, a transform type

of a transform associated with a transform block of the data block; and in response to the transform type belonging to a predefined set of transform types: extracting, from the coded video bitstream, a transform partition type associated with the data block.

[0026] According to another aspect, an embodiment of the present disclosure provides an apparatus for video encoding and/or decoding. The apparatus includes a memory storing instructions; and a processor in communication with the memory. When the processor executes the instructions, the processor is configured to cause the apparatus to perform the above methods for video decoding and/or encoding.

[0027] According to yet another aspect, an embodiment of the present disclosure provides non-transitory computer-readable mediums storing instructions which when executed by a computer for video decoding and/or encoding cause the computer to perform the above methods for video decoding and/or encoding.

[0028] The above and other aspects and their implementations are described in greater detail in the drawings, the descriptions, and the claims.

BRIEF DESCRIPTION OF THE DRAWINGS

[0029] Further features, the nature, and various advantages of the disclosed subject matter will be more apparent from the following detailed description and the accompanying drawings in which:

[0030] FIG. 1A shows a schematic illustration of an exemplary subset of intra prediction directional modes.

[0031] FIG. 1B shows an illustration of exemplary intra prediction directions.

[0032] FIG. 2 shows a schematic illustration of a current block and its surrounding spatial merge candidates for motion vector prediction in one example.

[0033] FIG. 3 shows a schematic illustration of a simplified block diagram of a communication system (300) in accordance with an example embodiment.

[0034] FIG. 4 shows a schematic illustration of a simplified block diagram of a communication system (400) in accordance with an example embodiment.

[0035] FIG. 5 shows a schematic illustration of a simplified block diagram of a video decoder in accordance with an example embodiment.

[0036] FIG. 6 shows a schematic illustration of a simplified block diagram of a video encoder in accordance with an example embodiment.

[0037] FIG. 7 shows a block diagram of a video encoder in accordance with another example embodiment.

[0038] FIG. 8 shows a block diagram of a video decoder in accordance with another example embodiment.

[0039] FIG. 9 shows directional intra prediction modes according to example embodiments of the disclosure.

[0040] FIG. 10 shows non-directional intra prediction modes according to example embodiments of the disclosure.

[0041] FIG. 11 shows recursive intra prediction modes according to example embodiments of the disclosure.

[0042] FIG. 12 shows transform block partitioning and scan of an intra prediction block according to example embodiments of the disclosure.

[0043] FIG. 13 shows transform block partitioning and scan of an inter prediction block according to example embodiments of the disclosure.

[0044] FIG. 14 shows low frequency non-separable transform process according to example embodiments of the disclosure.

[0045] FIG. 15 shows an intra prediction scheme based on various reference lines according to example embodiments of the disclosure.

[0046] FIG. 16 shows a scheme of non-recursive block partitioning according to example embodiments of the disclosure.

[0047] FIG. 17 shows a flow chart according to example embodiments of the disclosure.

[0048] FIG. 18 shows a schematic illustration of a computer system in accordance with example embodiments of the disclosure.

DETAILED DESCRIPTION OF EMBODIMENTS

[0049] FIG. 3 illustrates a simplified block diagram of a communication system (300) according to an embodiment of the present disclosure. The communication system (300) includes a plurality of terminal devices that can communicate with each other, via, for example, a network (350). For example, the communication system (300) includes a first pair of terminal devices (310) and (320) interconnected via the network (350). In the example of FIG. 3, the first pair of terminal devices (310) and (320) may perform unidirectional transmission of data. For example, the terminal device (310) may code video data (e.g., of a stream of video pictures that are captured by the terminal device (310)) for transmission to the other terminal device (320) via the network (350). The encoded video data can be transmitted in the form of one or more coded video bitstreams. The terminal

device (320) may receive the coded video data from the network (350), decode the coded video data to recover the video pictures and display the video pictures according to the recovered video data. Unidirectional data transmission may be implemented in media serving applications and the like.

[0050] In another example, the communication system (300) includes a second pair of terminal devices (330) and (340) that perform bidirectional transmission of coded video data that may be implemented, for example, during a videoconferencing application. For bidirectional transmission of data, in an example, each terminal device of the terminal devices (330) and (340) may code video data (e.g., of a stream of video pictures that are captured by the terminal device) for transmission to the other terminal device of the terminal devices (330) and (340) via the network (350). Each terminal device of the terminal devices (330) and (340) also may receive the coded video data transmitted by the other terminal device of the terminal devices (330) and (340), and may decode the coded video data to recover the video pictures and may display the video pictures at an accessible display device according to the recovered video data.

[0051] In the example of FIG. 3, the terminal devices (310), (320), (330) and (340) may be implemented as servers, personal computers and smart phones but the applicability of the underlying principles of the present disclosure may not be so limited. Embodiments of the present disclosure may be implemented in desktop computers, laptop computers, tablet computers, media players, wearable computers, dedicated video conferencing equipment, and/or the like. The network (350) represents any number or types of networks that convey coded video data among the terminal devices (310), (320), (330) and (340), including for example wireline (wired) and/or wireless communication networks. The communication network (350) may exchange data in circuit-switched, packet-switched, and/or other types of channels. Representative networks include telecommunications networks, local area networks, wide area networks and/or the Internet. For the purposes of the present discussion, the architecture and topology of the network (350) may be immaterial to the operation of the present disclosure unless explicitly explained herein.

[0052] FIG. 4 illustrates, as an example for an application for the disclosed subject matter, a placement of a video encoder and a video decoder in a video streaming environment. The disclosed subject matter may be equally applicable to other video applications, including, for example, video conferencing, digital TV broadcasting, gaming, virtual reality, storage of compressed video on digital media including CD, DVD, memory stick and the like, and so on.

[0053] A video streaming system may include a video capture subsystem (413) that can include a video source (401), e.g., a digital camera, for creating a stream of video pictures or images (402) that are uncompressed. In an example, the stream of video pictures (402) includes samples that are recorded by a digital camera of the video source 401. The stream of video pictures (402), depicted as a bold line to emphasize a high data volume when compared to encoded video data (404) (or coded video bitstreams), can be processed by an electronic device (420) that includes a video encoder (403) coupled to the video source (401). The video encoder (403) can include hardware, software, or a combination thereof to enable or implement aspects of the disclosed subject matter as described in more detail below. The encoded video data (404) (or encoded video bitstream (404)), depicted as a thin line to emphasize a lower data volume when compared to the stream of uncompressed video pictures (402), can be stored on a streaming server (405) for future use or directly to downstream video devices (not shown). One or more streaming client subsystems, such as client subsystems (406) and (408) in FIG. 4 can access the streaming server (405) to retrieve copies (407) and (409) of the encoded video data (404). A client subsystem (406) can include a video decoder (410), for example, in an electronic device (430). The video decoder (410) decodes the incoming copy (407) of the encoded video data and creates an outgoing stream of video pictures (411) that are uncompressed and that can be rendered on a display (412) (e.g., a display screen) or other rendering devices (not depicted). The video decoder 410 may be configured to perform some or all of the various functions described in this disclosure. In some streaming systems, the encoded video data (404), (407), and (409) (e.g., video bitstreams) can be encoded according to certain video coding/compression standards. Examples of those standards include ITU-T Recommendation H.265. In an example, a video coding standard under development is informally known as Versatile Video Coding (VVC). The disclosed subject matter may be used in the context of VVC, and other video coding standards.

[0054] It is noted that the electronic devices (420) and (430) can include other components (not shown). For example, the electronic device (420) can include a video decoder (not shown) and the electronic device (430) can include a video encoder (not shown) as well.

[0055] FIG. 5 shows a block diagram of a video decoder (510) according to any embodiment of the present disclosure below. The video decoder (510) can be included in an electronic device (530). The electronic device (530) can include a receiver (531) (e.g.,

receiving circuitry). The video decoder (510) can be used in place of the video decoder (410) in the example of FIG. 4.

[0056] The receiver (531) may receive one or more coded video sequences to be decoded by the video decoder (510). In the same or another embodiment, one coded video sequence may be decoded at a time, where the decoding of each coded video sequence is independent from other coded video sequences. Each video sequence may be associated with multiple video frames or images. The coded video sequence may be received from a channel (501), which may be a hardware/software link to a storage device which stores the encoded video data or a streaming source which transmits the encoded video data. The receiver (531) may receive the encoded video data with other data such as coded audio data and/or ancillary data streams, that may be forwarded to their respective processing circuitry (not depicted). The receiver (531) may separate the coded video sequence from the other data. To combat network jitter, a buffer memory (515) may be disposed in between the receiver (531) and an entropy decoder / parser (520) ("parser (520)" henceforth). In certain applications, the buffer memory (515) may be implemented as part of the video decoder (510). In other applications, it can be outside of and separate from the video decoder (510) (not depicted). In still other applications, there can be a buffer memory (not depicted) outside of the video decoder (510) for the purpose of, for example, combating network jitter, and there may be another additional buffer memory (515) inside the video decoder (510), for example to handle playback timing. When the receiver (531) is receiving data from a store/forward device of sufficient bandwidth and controllability, or from an isosynchronous network, the buffer memory (515) may not be needed, or can be small. For use on best-effort packet networks such as the Internet, the buffer memory (515) of sufficient size may be required, and its size can be comparatively large. Such buffer memory may be implemented with an adaptive size, and may at least partially be implemented in an operating system or similar elements (not depicted) outside of the video decoder (510).

[0057] The video decoder (510) may include the parser (520) to reconstruct symbols (521) from the coded video sequence. Categories of those symbols include information used to manage operation of the video decoder (510), and potentially information to control a rendering device such as display (512) (e.g., a display screen) that may or may not be an integral part of the electronic device (530) but can be coupled to the electronic device (530), as is shown in FIG. 5. The control information for the rendering device(s) may be in the form of Supplemental Enhancement Information (SEI messages) or Video Usability Information (VUI) parameter set fragments (not depicted). The parser (520) may parse/entropy-decode

the coded video sequence that is received by the parser (520). The entropy coding of the coded video sequence can be in accordance with a video coding technology or standard, and can follow various principles, including variable length coding, Huffman coding, arithmetic coding with or without context sensitivity, and so forth. The parser (520) may extract from the coded video sequence, a set of subgroup parameters for at least one of the subgroups of pixels in the video decoder, based upon at least one parameter corresponding to the subgroups. The subgroups can include Groups of Pictures (GOPs), pictures, tiles, slices, macroblocks, Coding Units (CUs), blocks, Transform Units (TUs), Prediction Units (PUs) and so forth. The parser (520) may also extract from the coded video sequence information such as transform coefficients (e.g., Fourier transform coefficients), quantizer parameter values, motion vectors, and so forth.

[0058] The parser (520) may perform an entropy decoding / parsing operation on the video sequence received from the buffer memory (515), so as to create symbols (521).

[0059] Reconstruction of the symbols (521) can involve multiple different processing or functional units depending on the type of the coded video picture or parts thereof (such as: inter and intra picture, inter and intra block), and other factors. The units that are involved and how they are involved may be controlled by the subgroup control information that was parsed from the coded video sequence by the parser (520). The flow of such subgroup control information between the parser (520) and the multiple processing or functional units below is not depicted for simplicity.

[0060] Beyond the functional blocks already mentioned, the video decoder (510) can be conceptually subdivided into a number of functional units as described below. In a practical implementation operating under commercial constraints, many of these functional units interact closely with each other and can, at least partly, be integrated with one another. However, for the purpose of describing the various functions of the disclosed subject matter with clarity, the conceptual subdivision into the functional units is adopted in the disclosure below.

[0061] A first unit may include the scaler / inverse transform unit (551). The scaler / inverse transform unit (551) may receive a quantized transform coefficient as well as control information, including information indicating which type of inverse transform to use, block size, quantization factor/parameters, quantization scaling matrices, and the like as symbol(s) (521) from the parser (520). The scaler / inverse transform unit (551) can output blocks comprising sample values that can be input into aggregator (555).

[0062] In some cases, the output samples of the scaler / inverse transform (551) can pertain to an intra coded block, i.e., a block that does not use predictive information from previously reconstructed pictures, but can use predictive information from previously reconstructed parts of the current picture. Such predictive information can be provided by an intra picture prediction unit (552). In some cases, the intra picture prediction unit (552) may generate a block of the same size and shape of the block under reconstruction using surrounding block information that is already reconstructed and stored in the current picture buffer (558). The current picture buffer (558) buffers, for example, partly reconstructed current picture and/or fully reconstructed current picture. The aggregator (555), in some implementations, may add, on a per sample basis, the prediction information the intra prediction unit (552) has generated to the output sample information as provided by the scaler / inverse transform unit (551).

[0063] In other cases, the output samples of the scaler / inverse transform unit (551) can pertain to an inter coded, and potentially motion compensated block. In such a case, a motion compensation prediction unit (553) can access reference picture memory (557) to fetch samples used for inter-picture prediction. After motion compensating the fetched samples in accordance with the symbols (521) pertaining to the block, these samples can be added by the aggregator (555) to the output of the scaler / inverse transform unit (551) (output of unit 551 may be referred to as the residual samples or residual signal) so as to generate output sample information. The addresses within the reference picture memory (557) from where the motion compensation prediction unit (553) fetches prediction samples can be controlled by motion vectors, available to the motion compensation prediction unit (553) in the form of symbols (521) that can have, for example X, Y components (shift), and reference picture components (time). Motion compensation may also include interpolation of sample values as fetched from the reference picture memory (557) when sub-sample exact motion vectors are in use, and may also be associated with motion vector prediction mechanisms, and so forth.

[0064] The output samples of the aggregator (555) can be subject to various loop filtering techniques in the loop filter unit (556). Video compression technologies can include in-loop filter technologies that are controlled by parameters included in the coded video sequence (also referred to as coded video bitstream) and made available to the loop filter unit (556) as symbols (521) from the parser (520), but can also be responsive to meta-information obtained during the decoding of previous (in decoding order) parts of the coded picture or coded video sequence, as well as responsive to previously reconstructed and loop-filtered

sample values. Several type of loop filters may be included as part of the loop filter unit 556 in various orders, as will be described in further detail below.

[0065] The output of the loop filter unit (556) can be a sample stream that can be output to the rendering device (512) as well as stored in the reference picture memory (557) for use in future inter-picture prediction.

[0066] Certain coded pictures, once fully reconstructed, can be used as reference pictures for future inter-picture prediction. For example, once a coded picture corresponding to a current picture is fully reconstructed and the coded picture has been identified as a reference picture (by, for example, the parser (520)), the current picture buffer (558) can become a part of the reference picture memory (557), and a fresh current picture buffer can be reallocated before commencing the reconstruction of the following coded picture.

[0067] The video decoder (510) may perform decoding operations according to a predetermined video compression technology adopted in a standard, such as ITU-T Rec. H.265. The coded video sequence may conform to a syntax specified by the video compression technology or standard being used, in the sense that the coded video sequence adheres to both the syntax of the video compression technology or standard and the profiles as documented in the video compression technology or standard. Specifically, a profile can select certain tools from all the tools available in the video compression technology or standard as the only tools available for use under that profile. To be standard-compliant, the complexity of the coded video sequence may be within bounds as defined by the level of the video compression technology or standard. In some cases, levels restrict the maximum picture size, maximum frame rate, maximum reconstruction sample rate (measured in, for example megasamples per second), maximum reference picture size, and so on. Limits set by levels can, in some cases, be further restricted through Hypothetical Reference Decoder (HRD) specifications and metadata for HRD buffer management signaled in the coded video sequence.

[0068] In some example embodiments, the receiver (531) may receive additional (redundant) data with the encoded video. The additional data may be included as part of the coded video sequence(s). The additional data may be used by the video decoder (510) to properly decode the data and/or to more accurately reconstruct the original video data. Additional data can be in the form of, for example, temporal, spatial, or signal noise ratio (SNR) enhancement layers, redundant slices, redundant pictures, forward error correction codes, and so on.

[0069] FIG. 6 shows a block diagram of a video encoder (603) according to an example embodiment of the present disclosure. The video encoder (603) may be included in an electronic device (620). The electronic device (620) may further include a transmitter (640) (e.g., transmitting circuitry). The video encoder (603) can be used in place of the video encoder (403) in the example of FIG. 4.

[0070] The video encoder (603) may receive video samples from a video source (601) (that is not part of the electronic device (620) in the example of FIG. 6) that may capture video image(s) to be coded by the video encoder (603). In another example, the video source (601) may be implemented as a portion of the electronic device (620).

[0071] The video source (601) may provide the source video sequence to be coded by the video encoder (603) in the form of a digital video sample stream that can be of any suitable bit depth (for example: 8 bit, 10 bit, 12 bit, ...), any colorspace (for example, BT.601 YCrCb, RGB, XYZ...), and any suitable sampling structure (for example YCrCb 4:2:0, YCrCb 4:4:4). In a media serving system, the video source (601) may be a storage device capable of storing previously prepared video. In a videoconferencing system, the video source (601) may be a camera that captures local image information as a video sequence. Video data may be provided as a plurality of individual pictures or images that impart motion when viewed in sequence. The pictures themselves may be organized as a spatial array of pixels, wherein each pixel can comprise one or more samples depending on the sampling structure, color space, and the like being in use. A person having ordinary skill in the art can readily understand the relationship between pixels and samples. The description below focuses on samples.

[0072] According to some example embodiments, the video encoder (603) may code and compress the pictures of the source video sequence into a coded video sequence (643) in real time or under any other time constraints as required by the application. Enforcing appropriate coding speed constitutes one function of a controller (650). In some embodiments, the controller (650) may be functionally coupled to and control other functional units as described below. The coupling is not depicted for simplicity. Parameters set by the controller (650) can include rate control related parameters (picture skip, quantizer, lambda value of rate-distortion optimization techniques, ...), picture size, group of pictures (GOP) layout, maximum motion vector search range, and the like. The controller (650) can be configured to have other suitable functions that pertain to the video encoder (603) optimized for a certain system design.

[0073] In some example embodiments, the video encoder (603) may be configured to operate in a coding loop. As an oversimplified description, in an example, the coding loop can include a source coder (630) (e.g., responsible for creating symbols, such as a symbol stream, based on an input picture to be coded, and a reference picture(s)), and a (local) decoder (633) embedded in the video encoder (603). The decoder (633) reconstructs the symbols to create the sample data in a similar manner as a (remote) decoder would create even though the embedded decoder 633 process coded video steam by the source coder 630 without entropy coding (as any compression between symbols and coded video bitstream in entropy coding may be lossless in the video compression technologies considered in the disclosed subject matter). The reconstructed sample stream (sample data) is input to the reference picture memory (634). As the decoding of a symbol stream leads to bit-exact results independent of decoder location (local or remote), the content in the reference picture memory (634) is also bit exact between the local encoder and remote encoder. In other words, the prediction part of an encoder "sees" as reference picture samples exactly the same sample values as a decoder would "see" when using prediction during decoding. This fundamental principle of reference picture synchronicity (and resulting drift, if synchronicity cannot be maintained, for example because of channel errors) is used to improve coding quality.

[0074] The operation of the "local" decoder (633) can be the same as of a "remote" decoder, such as the video decoder (510), which has already been described in detail above in conjunction with FIG. 5. Briefly referring also to FIG. 5, however, as symbols are available and encoding/decoding of symbols to a coded video sequence by an entropy coder (645) and the parser (520) can be lossless, the entropy decoding parts of the video decoder (510), including the buffer memory (515), and parser (520) may not be fully implemented in the local decoder (633) in the encoder.

[0075] An observation that can be made at this point is that any decoder technology except the parsing/entropy decoding that may only be present in a decoder also may necessarily need to be present, in substantially identical functional form, in a corresponding encoder. For this reason, the disclosed subject matter may at times focus on decoder operation, which allies to the decoding portion of the encoder. The description of encoder technologies can thus be abbreviated as they are the inverse of the comprehensively described decoder technologies. Only in certain areas or aspects a more detail description of the encoder is provided below.

[0076] During operation in some example implementations, the source coder (630) may perform motion compensated predictive coding, which codes an input picture predictively with reference to one or more previously coded picture from the video sequence that were designated as "reference pictures." In this manner, the coding engine (632) codes differences (or residue) in the color channels between pixel blocks of an input picture and pixel blocks of reference picture(s) that may be selected as prediction reference(s) to the input picture. The term "residue" and its adjective form "residual" may be used interchangeably.

[0077] The local video decoder (633) may decode coded video data of pictures that may be designated as reference pictures, based on symbols created by the source coder (630). Operations of the coding engine (632) may advantageously be lossy processes. When the coded video data may be decoded at a video decoder (not shown in FIG. 6), the reconstructed video sequence typically may be a replica of the source video sequence with some errors. The local video decoder (633) replicates decoding processes that may be performed by the video decoder on reference pictures and may cause reconstructed reference pictures to be stored in the reference picture cache (634). In this manner, the video encoder (603) may store copies of reconstructed reference pictures locally that have common content as the reconstructed reference pictures that will be obtained by a far-end (remote) video decoder (absent transmission errors).

[0078] The predictor (635) may perform prediction searches for the coding engine (632). That is, for a new picture to be coded, the predictor (635) may search the reference picture memory (634) for sample data (as candidate reference pixel blocks) or certain metadata such as reference picture motion vectors, block shapes, and so on, that may serve as an appropriate prediction reference for the new pictures. The predictor (635) may operate on a sample block-by-pixel block basis to find appropriate prediction references. In some cases, as determined by search results obtained by the predictor (635), an input picture may have prediction references drawn from multiple reference pictures stored in the reference picture memory (634).

[0079] The controller (650) may manage coding operations of the source coder (630), including, for example, setting of parameters and subgroup parameters used for encoding the video data.

[0080] Output of all aforementioned functional units may be subjected to entropy coding in the entropy coder (645). The entropy coder (645) translates the symbols as generated by the various functional units into a coded video sequence, by lossless

compression of the symbols according to technologies such as Huffman coding, variable length coding, arithmetic coding, and so forth.

[0081] The transmitter (640) may buffer the coded video sequence(s) as created by the entropy coder (645) to prepare for transmission via a communication channel (660), which may be a hardware/software link to a storage device which would store the encoded video data. The transmitter (640) may merge coded video data from the video coder (603) with other data to be transmitted, for example, coded audio data and/or ancillary data streams (sources not shown).

[0082] The controller (650) may manage operation of the video encoder (603). During coding, the controller (650) may assign to each coded picture a certain coded picture type, which may affect the coding techniques that may be applied to the respective picture. For example, pictures often may be assigned as one of the following picture types:

[0083] An Intra Picture (I picture) may be one that may be coded and decoded without using any other picture in the sequence as a source of prediction. Some video codecs allow for different types of intra pictures, including, for example Independent Decoder Refresh (“IDR”) Pictures. A person having ordinary skill in the art is aware of those variants of I pictures and their respective applications and features.

[0084] A predictive picture (P picture) may be one that may be coded and decoded using intra prediction or inter prediction using at most one motion vector and reference index to predict the sample values of each block.

[0085] A bi-directionally predictive picture (B Picture) may be one that may be coded and decoded using intra prediction or inter prediction using at most two motion vectors and reference indices to predict the sample values of each block. Similarly, multiple-predictive pictures can use more than two reference pictures and associated metadata for the reconstruction of a single block.

[0086] Source pictures commonly may be subdivided spatially into a plurality of sample coding blocks (for example, blocks of 4 x 4, 8 x 8, 4 x 8, or 16 x 16 samples each) and coded on a block-by-block basis. Blocks may be coded predictively with reference to other (already coded) blocks as determined by the coding assignment applied to the blocks’ respective pictures. For example, blocks of I pictures may be coded non-predictively or they may be coded predictively with reference to already coded blocks of the same picture (spatial prediction or intra prediction). Pixel blocks of P pictures may be coded predictively, via spatial prediction or via temporal prediction with reference to one previously coded reference picture. Blocks of B pictures may be coded predictively, via spatial prediction or via

temporal prediction with reference to one or two previously coded reference pictures. The source pictures or the intermediate processed pictures may be subdivided into other types of blocks for other purposes. The division of coding blocks and the other types of blocks may or may not follow the same manner, as described in further detail below.

[0087] The video encoder (603) may perform coding operations according to a predetermined video coding technology or standard, such as ITU-T Rec. H.265. In its operation, the video encoder (603) may perform various compression operations, including predictive coding operations that exploit temporal and spatial redundancies in the input video sequence. The coded video data may accordingly conform to a syntax specified by the video coding technology or standard being used.

[0088] In some example embodiments, the transmitter (640) may transmit additional data with the encoded video. The source coder (630) may include such data as part of the coded video sequence. The additional data may comprise temporal/spatial/SNR enhancement layers, other forms of redundant data such as redundant pictures and slices, SEI messages, VUI parameter set fragments, and so on.

[0089] A video may be captured as a plurality of source pictures (video pictures) in a temporal sequence. Intra-picture prediction (often abbreviated to intra prediction) utilizes spatial correlation in a given picture, and inter-picture prediction utilizes temporal or other correlation between the pictures. For example, a specific picture under encoding/decoding, which is referred to as a current picture, may be partitioned into blocks. A block in the current picture, when similar to a reference block in a previously coded and still buffered reference picture in the video, may be coded by a vector that is referred to as a motion vector. The motion vector points to the reference block in the reference picture, and can have a third dimension identifying the reference picture, in case multiple reference pictures are in use.

[0090] In some example embodiments, a bi-prediction technique can be used for inter-picture prediction. According to such bi-prediction technique, two reference pictures, such as a first reference picture and a second reference picture that both proceed the current picture in the video in decoding order (but may be in the past or future, respectively, in display order) are used. A block in the current picture can be coded by a first motion vector that points to a first reference block in the first reference picture, and a second motion vector that points to a second reference block in the second reference picture. The block can be jointly predicted by a combination of the first reference block and the second reference block.

[0091] Further, a merge mode technique may be used in the inter-picture prediction to improve coding efficiency.

[0092] According to some example embodiments of the disclosure, predictions, such as inter-picture predictions and intra-picture predictions are performed in the unit of blocks. For example, a picture in a sequence of video pictures is partitioned into coding tree units (CTU) for compression, the CTUs in a picture may have the same size, such as 64 x 64 pixels, 32 x 32 pixels, or 16 x 16 pixels. In general, a CTU may include three parallel coding tree blocks (CTBs): one luma CTB and two chroma CTBs. Each CTU can be recursively quadtree split into one or multiple coding units (CUs). For example, a CTU of 64 x 64 pixels can be split into one CU of 64 x 64 pixels, or 4 CUs of 32 x 32 pixels. Each of the one or more of the 32 x 32 block may be further split into 4 CUs of 16 x 16 pixels. In some example embodiments, each CU may be analyzed during encoding to determine a prediction type for the CU among various prediction types such as an inter prediction type or an intra prediction type. The CU may be split into one or more prediction units (PUs) depending on the temporal and/or spatial predictability. Generally, each PU includes a luma prediction block (PB), and two chroma PBs. In an embodiment, a prediction operation in coding (encoding/decoding) is performed in the unit of a prediction block. The split of a CU into PU (or PBs of different color channels) may be performed in various spatial pattern. A luma or chroma PB, for example, may include a matrix of values (e.g., luma values) for samples, such as 8 x 8 pixels, 16 x 16 pixels, 8 x 16 pixels, 16 x 8 samples, and the like.

[0093] FIG. 7 shows a diagram of a video encoder (703) according to another example embodiment of the disclosure. The video encoder (703) is configured to receive a processing block (e.g., a prediction block) of sample values within a current video picture in a sequence of video pictures, and encode the processing block into a coded picture that is part of a coded video sequence. The example video encoder (703) may be used in place of the video encoder (403) in the FIG. 4 example.

[0094] For example, the video encoder (703) receives a matrix of sample values for a processing block, such as a prediction block of 8 x 8 samples, and the like. The video encoder (703) then determines whether the processing block is best coded using intra mode, inter mode, or bi-prediction mode using, for example, rate-distortion optimization (RDO). When the processing block is determined to be coded in intra mode, the video encoder (703) may use an intra prediction technique to encode the processing block into the coded picture; and when the processing block is determined to be coded in inter mode or bi-prediction mode, the video encoder (703) may use an inter prediction or bi-prediction technique, respectively, to encode the processing block into the coded picture. In some example embodiments, a merge mode may be used as a submode of the inter picture prediction where

the motion vector is derived from one or more motion vector predictors without the benefit of a coded motion vector component outside the predictors. In some other example embodiments, a motion vector component applicable to the subject block may be present. Accordingly, the video encoder (703) may include components not explicitly shown in FIG. 7, such as a mode decision module, to determine the prediction mode of the processing blocks.

[0095] In the example of FIG. 7, the video encoder (703) includes an inter encoder (730), an intra encoder (722), a residue calculator (723), a switch (726), a residue encoder (724), a general controller (721), and an entropy encoder (725) coupled together as shown in the example arrangement in FIG. 7.

[0096] The inter encoder (730) is configured to receive the samples of the current block (e.g., a processing block), compare the block to one or more reference blocks in reference pictures (e.g., blocks in previous pictures and later pictures in display order), generate inter prediction information (e.g., description of redundant information according to inter encoding technique, motion vectors, merge mode information), and calculate inter prediction results (e.g., predicted block) based on the inter prediction information using any suitable technique. In some examples, the reference pictures are decoded reference pictures that are decoded based on the encoded video information using the decoding unit 633 embedded in the example encoder 620 of FIG. 6 (shown as residual decoder 728 of FIG. 7, as described in further detail below).

[0097] The intra encoder (722) is configured to receive the samples of the current block (e.g., a processing block), compare the block to blocks already coded in the same picture, and generate quantized coefficients after transform, and in some cases also to generate intra prediction information (e.g., an intra prediction direction information according to one or more intra encoding techniques). The intra encoder (722) may calculate intra prediction results (e.g., predicted block) based on the intra prediction information and reference blocks in the same picture.

[0098] The general controller (721) may be configured to determine general control data and control other components of the video encoder (703) based on the general control data. In an example, the general controller (721) determines the prediction mode of the block, and provides a control signal to the switch (726) based on the prediction mode. For example, when the prediction mode is the intra mode, the general controller (721) controls the switch (726) to select the intra mode result for use by the residue calculator (723), and controls the entropy encoder (725) to select the intra prediction information and include the intra prediction information in the bitstream; and when the prediction mode for the block is

the inter mode, the general controller (721) controls the switch (726) to select the inter prediction result for use by the residue calculator (723), and controls the entropy encoder (725) to select the inter prediction information and include the inter prediction information in the bitstream.

[0099] The residue calculator (723) may be configured to calculate a difference (residue data) between the received block and prediction results for the block selected from the intra encoder (722) or the inter encoder (730). The residue encoder (724) may be configured to encode the residue data to generate transform coefficients. For example, the residue encoder (724) may be configured to convert the residue data from a spatial domain to a frequency domain to generate the transform coefficients. The transform coefficients are then subject to quantization processing to obtain quantized transform coefficients. In various example embodiments, the video encoder (703) also includes a residual decoder (728). The residual decoder (728) is configured to perform inverse-transform, and generate the decoded residue data. The decoded residue data can be suitably used by the intra encoder (722) and the inter encoder (730). For example, the inter encoder (730) can generate decoded blocks based on the decoded residue data and inter prediction information, and the intra encoder (722) can generate decoded blocks based on the decoded residue data and the intra prediction information. The decoded blocks are suitably processed to generate decoded pictures and the decoded pictures can be buffered in a memory circuit (not shown) and used as reference pictures.

[0100] The entropy encoder (725) may be configured to format the bitstream to include the encoded block and perform entropy coding. The entropy encoder (725) is configured to include in the bitstream various information. For example, the entropy encoder (725) may be configured to include the general control data, the selected prediction information (e.g., intra prediction information or inter prediction information), the residue information, and other suitable information in the bitstream. When coding a block in the merge submode of either inter mode or bi-prediction mode, there may be no residue information.

[0101] FIG. 8 shows a diagram of an example video decoder (810) according to another embodiment of the disclosure. The video decoder (810) is configured to receive coded pictures that are part of a coded video sequence, and decode the coded pictures to generate reconstructed pictures. In an example, the video decoder (810) may be used in place of the video decoder (410) in the example of FIG. 4.

[0102] In the example of FIG. 8, the video decoder (810) includes an entropy decoder (871), an inter decoder (880), a residual decoder (873), a reconstruction module (874), and an intra decoder (872) coupled together as shown in the example arrangement of FIG. 8.

[0103] The entropy decoder (871) can be configured to reconstruct, from the coded picture, certain symbols that represent the syntax elements of which the coded picture is made up. Such symbols can include, for example, the mode in which a block is coded (e.g., intra mode, inter mode, bi-predicted mode, merge submode or another submode), prediction information (e.g., intra prediction information or inter prediction information) that can identify certain sample or metadata used for prediction by the intra decoder (872) or the inter decoder (880), residual information in the form of, for example, quantized transform coefficients, and the like. In an example, when the prediction mode is the inter or bi-predicted mode, the inter prediction information is provided to the inter decoder (880); and when the prediction type is the intra prediction type, the intra prediction information is provided to the intra decoder (872). The residual information can be subject to inverse quantization and is provided to the residual decoder (873).

[0104] The inter decoder (880) may be configured to receive the inter prediction information, and generate inter prediction results based on the inter prediction information.

[0105] The intra decoder (872) may be configured to receive the intra prediction information, and generate prediction results based on the intra prediction information.

[0106] The residual decoder (873) may be configured to perform inverse quantization to extract de-quantized transform coefficients, and process the de-quantized transform coefficients to convert the residual from the frequency domain to the spatial domain. The residual decoder (873) may also utilize certain control information (to include the Quantizer Parameter (QP)) which may be provided by the entropy decoder (871) (data path not depicted as this may be low data volume control information only).

[0107] The reconstruction module (874) may be configured to combine, in the spatial domain, the residual as output by the residual decoder (873) and the prediction results (as output by the inter or intra prediction modules as the case may be) to form a reconstructed block forming part of the reconstructed picture as part of the reconstructed video. It is noted that other suitable operations, such as a deblocking operation and the like, may also be performed to improve the visual quality.

[0108] It is noted that the video encoders (403), (603), and (703), and the video decoders (410), (510), and (810) can be implemented using any suitable technique. In some example embodiments, the video encoders (403), (603), and (703), and the video decoders

(410), (510), and (810) can be implemented using one or more integrated circuits. In another embodiment, the video encoders (403), (603), and (603), and the video decoders (410), (510), and (810) can be implemented using one or more processors that execute software instructions.

[0109] Returning to the intra prediction process, in which samples in a block (e.g., a luma or chroma prediction block, or coding block if not further split into prediction blocks) is predicted by samples of neighboring, next neighboring, or other line or lines, or the combination thereof, to generate a prediction block. The residual between the actual block being coded and the prediction block may then be processed via transform followed by quantization. Various intra prediction modes may be made available and parameters related to intra mode selection and other parameters may be signaled in the bitstream. The various intra prediction modes, for example, may pertain to line position or positions for predicting samples, directions along which prediction samples are selected from predicting line or lines, and other special intra prediction modes.

[0110] For example, a set of intra prediction modes (interchangeably referred to as “intra modes”) may include a predefined number of directional intra prediction modes. As described above in relation to the example implementation of FIG. 1, these intra prediction modes may correspond to a predefined number of directions along which out-of-block samples are selected as prediction for samples being predicted in a particular block. In another particular example implementation, eight (8) main directional modes corresponding to angles from 45 to 207 degrees to the horizontal axis may be supported and predefined.

[0111] In some other implementations of intra prediction, to further exploit more varieties of spatial redundancy in directional textures, directional intra modes may be further extended to an angle set with finer granularity. For example, the 8-angle implementation above may be configured to provide eight nominal angles, referred to as V_PRED, H_PRED, D45_PRED, D135_PRED, D113_PRED, D157_PRED, D203_PRED, and D67_PRED, as illustrated in FIG. 9, and for each nominal angle, a predefined number (e.g., 7) of finer angles may be added. With such an extension, a larger total number (e.g., 56 in this example) of directional angles may be available for intra prediction, corresponding to the same number of predefined directional intra modes. A prediction angle may be represented by a nominal intra angle plus an angle delta. For the particular example above with 7 finer angular directions for each nominal angle, the angle delta may be $-3 \sim 3$ multiplies a step size of 3 degrees.

[0112] In some implementations, alternative or in addition to the direction intra modes above, a predefined number of non-directional intra prediction modes may also be

predefined and made available. For example, 5 non-direction intra modes referred to as smooth intra prediction modes may be specified. These non-directional intra mode prediction modes may be specifically referred to as DC, PAETH, SMOOTH, SMOOTH_V, and SMOOTH_H intra modes. Prediction of samples of a particular block under these example non-directional modes are illustrated in FIG. 10. As an example, FIG. 10 shows a 4 x 4 block 1002 being predicted by samples from a top neighboring line and/or left neighboring line. A particular sample 1010 in block 1002 may correspond to directly top sample 1004 of the sample 1010 in the top neighboring line of block 1002, a top-left sample 1006 of the sample 1010 as the intersection of the top and left neighboring lines, and a directly left sample 1008 of the sample 1010 in the left neighboring line of block 1002. For the example DC intra prediction mode, an average of the left and above neighboring samples 1008 and 1004 may be used as the predictor of the sample 1010. For the example PAETH intra prediction mode, the top, left, and top-left reference samples 1004, 1008, and 1006 may be fetched, and then whichever value among these three reference samples that is the closest to $(\text{top} + \text{left} - \text{topleft})$ may be set as the predictor for the sample 1010. For the example SMOOTH_V intra prediction mode, the sample 1010 may be predicted by a quadratic interpolation in vertical direction of the top-left neighboring sample 1006 and the left neighboring sample 1008. For the example SMOOTH_H intra prediction mode, the sample 1010 may be predicted by a quadratic interpolation in horizontal direction of the top-left neighboring sample 1006 and the top neighboring sample 1004. For the example SMOOTH intra prediction mode, the sample 1010 may be predicted by an average of the quadratic interpolations in the vertical and the horizontal directions. The non-directional intra mode implementations above are merely illustrated as a non-limiting example. Other neighboring lines, and other non-directional selection of samples, and manners of combining predicting samples for predicting a particular sample in a prediction block are also contemplated.

[0113] Selection of a particular intra prediction mode by the encoder from the directional or non-directional modes above at various coding levels (picture, slice, block, unit, etc.) may be signaled in the bitstream. In some example implementations, the exemplary 8 nominal directional modes together with 5 non-angular smooth modes (a total of 13 options) may be signaled first. Then if the signaled mode is one of the 8 nominal angular intra modes, an index is further signaled to indicate the selected angle delta to the corresponding signaled nominal angle. In some other example implementations, all intra prediction modes may be indexed all together (e.g., 56 directional modes plus 5 non-directional modes to yield 61 intra prediction modes) for signaling.

[0114] In some example implementations, the example 56 or other number of directional intra prediction modes may be implemented with a unified directional predictor that projects each sample of a block to a reference sub-sample location and interpolates the reference sample by a 2-tap bilinear filter.

[0115] In some implementations, to capture decaying spatial correlation with references on the edges, additional filter modes referred to as FILTER INTRA modes may be designed. For these modes, predicted samples within the block in addition to out-of-block samples may be used as intra prediction reference samples for some patches within the block. These modes, for example, may be predefined and made available to intra prediction for at least luma blocks (or only luma blocks). A predefined number (e.g., five) of filter intra modes may be pre-designed, each represented by a set of n-tap filters (e.g., 7-tap filters) reflecting correlation between samples in, for example, a 4 x 2 patch and n neighbors adjacent to it. In other words, the weighting factors for an n-tap filter may be position dependent. Taking an 8 x 8 block, 4 x 2 patch, and 7-tap filtering as an example, as shown in FIG. 11, the 8 x 8 block 1102 may be split into eight 4 x 2 patches. These patches are indicated by B0, B1, B1, B3, B4, B5, B6, and B7 in FIG. 11. For each patch, its 7 neighbors, indicated by R0 ~ R7 in FIG. 11, may be used to predict the samples in a current patch. For patch B0, all the neighbors may have been already reconstructed. But for other patches, some of the neighbors are in the current block and thus may not have been reconstructed, then the predicted values of immediate neighbors are used as the reference. For example, all the neighbors of patch B7 as indicated in FIG. 11 are not reconstructed, so the prediction samples of neighbors are used instead.

[0116] In some implementation of intra prediction, one color component may be predicted using one or more other color components. A color component may be any one of components in YCrCb, RGB, XYZ color space and the like. For example, a prediction of chroma component (e.g., chroma block) from luma component (e.g., luma reference samples), referred to as Chroma from Luma, or CfL), may be implemented. In some example implementations, cross-color prediction may only be allowed from luma to chroma. For example, a chroma sample in a chroma block may be modeled as a linear function of coincident reconstructed luma samples. The CfL prediction may be implemented as follows:

$$CfL(\alpha) = \alpha \times L^{AC} + DC \quad (1)$$

[0117] where L^{AC} denotes an AC contribution of luma component, α denotes a parameter of the linear model, and DC denotes a DC contribution of the chroma component. The AC components, for example is obtained for each samples of the block whereas the DC

component is obtained for the entire block. To be specific, the reconstructed luma samples may be subsampled into the chroma resolution, and then the average luma value (DC of luma) may be subtracted from each luma value to form the AC contribution in luma. The AC contribution of Luma is then used in the linear mode of Eq. (1) to predict the AC values of the chroma component. To approximate or predict chroma AC component from the luma AC contribution, instead of requiring the decoder to calculate the scaling parameters, an example CfL implementation may determine the parameter α based on the original chroma samples and signal them in the bitstream. This reduces decoder complexity and yields more precise predictions. As for the DC contribution of the chroma component, it may be computed using intra DC mode within the chroma component in some example implementations.

[0118] In some example implementations of the reference lines, multi-line intra prediction may be used. In these implementations, more than one reference lines may be available for selection in intra prediction, and encoder decides and signals which reference line is used to generate the intra predictor. The reference line index may be signaled before intra prediction modes, and only the most probable prediction modes are allowed in case a non-zero reference line index is signaled. Referring to FIG. 15, an example of 4 reference lines is depicted (from reference line 0 to reference line 3), where each reference line is composed of six segments, i.e., Segment A to F (as indicated by 1502-1512), together with the top-left reference sample. In addition, Segment A and F may be padded with the closest samples from Segment B and E, respectively.

[0119] Transform of a residual of either an intra prediction block or an inter prediction block may then be implemented followed by quantization of the transform coefficient. For the purpose of performing transform, both intra and inter coded blocks may be further partitioned into multiple transform blocks (sometimes interchangeably used as “transform units”, even though the term “unit” is normally used to represent a congregation of the three-color channels, e.g., a “coding unit” would include luma coding block, and chroma coding blocks) prior to the transform. In some implementations, the maximum partitioning depth of the coded blocks (or prediction blocks) may be specified (the term “coded blocks” may be used interchangeably with “coding blocks”). For example, such partitioning may not go beyond 2 levels. The division of prediction block into transform blocks may be handled differently between intra prediction blocks and inter prediction blocks. In some implementations, however, such division may be similar between intra prediction blocks and inter prediction blocks.

[0120] In some example implementations, and for intra coded blocks, the transform partition may be done in a way that all the transform blocks have the same size, and the transform blocks are coded in a raster scan order. An example of such transform block partitioning of an intra coded block is shown in FIG. 12. Specifically, FIG. 12 illustrates the coded block 1202 is partitioned via an intermediate level quadtree splitting 1204 into 16 transform blocks of the same block size, as shown by 1206. The example raster scan order for coding is illustrated by the ordered arrows in FIG. 12.

[0121] In some example implementations, and for inter coded blocks, the transform unit partitioning may be done in a recursive manner with the partitioning depth up to a predefined number of levels (e.g., 2 levels). Split may stop or continue recursively for any sub partition and at any level, as shown in FIG. 13. In particular, FIG. 13 shows an example where the block 1302 is split into four quadtree sub blocks 1304 and one of the subblocks is further split into four second level transform blocks whereas division of the other subblocks stops after the first level, yielding a total of 7 transform blocks of two different sizes. The example raster scan order for coding is further illustrated by the ordered arrows in FIG. 13. While FIG. 13 shows an example implementation of quadtree split of up-to two levels of square transform blocks, in some generation implementations, the transform partitioning may support 1:1 (square), 1:2/2:1, and 1:4/4:1 transform block shapes and sizes ranging from 4×4 to 64×64 . In some example implementations, if the coding block is smaller than or equal to 64×64 , the transform block partitioning may only be applied to luma component (in other words, the chroma transform block would be the same as the coding block under that condition). Otherwise, if the coding block width or height is greater than 64, both the luma and chroma coding blocks may be implicitly split into multiples of $\min(W, 64) \times \min(H, 64)$ and $\min(W, 32) \times \min(H, 32)$ transform blocks, respectively.

[0122] In some example implementations, as shown in FIG. 16, another alternative example scheme for partitioning a coding block or prediction block into transform blocks is provided. As shown in FIG. 16, instead of using recursive transform partitioning, a predefined set of partition types may be applied to a coding block according a transform type of the coding block. In the particular example shown in FIG. 16, one of the 6 example partition types may be applied to split a coding block into various number of transform blocks. Such scheme may be applied to either a coding block or a prediction block. In this disclosure, the term “partition type” may generally refer to the manner in which a block (e.g., a prediction block or a coding block) is partitioned, and it may refer to “transform partition type”, “prediction block partition type”, or “coding block partition type”. Additionally, for

description under “transform partition type”, the same concept may also apply to “coding block partition type”, and vice versa.

[0123] In more detail, the partitioning scheme of FIG. 16 provides up to 6 partition types for any given transform type as shown in FIG. 16. In this scheme, every coding block or prediction block may be assigned a transform type based on, for example, a rate-distortion cost. In an example, the partition type assigned to the coding block or prediction block may be determined based on the transform type of the coding block or prediction block. A particular partition type may correspond to a transform block split size and pattern (or partition type), as shown by the 4 partition types illustrated in FIG. 16. A correspondence relationship between various transform types and the various partition types may be predefined. An example correspondence is shown below with the capitalized labels indicating the transform types that may be assigned to the coding block or prediction block based on rate distortion cost:

[0124] • PARTITION_NONE: Assigns a transform size that is equal to the block size.

[0125] • PARTITION_SPLIT: Assigns a transform size that is $\frac{1}{2}$ the width of the block size and $\frac{1}{2}$ the height of the block size.

[0126] • PARTITION_HORZ: Assigns a transform size with the same width as the block size and $\frac{1}{2}$ the height of the block size.

[0127] • PARTITION_VERT: Assigns a transform size with $\frac{1}{2}$ the width of the block size and the same height as the block size.

[0128] • PARTITION_HORZ4: Assigns a transform size with the same width as the block size and $\frac{1}{4}$ the height of the block size.

[0129] • PARTITION_VERT4: Assigns a transform size with $\frac{1}{4}$ the width of the block size and the same height as the block size.

[0130] In the example above, the partition types as shown in FIG. 16 all contain uniform transform sizes for the partitioned transform blocks. This is a mere example rather than a limitation. In some other implementations, mixed transform blocks sizes may be used for the partitioned transform blocks in a particular partition type (or pattern).

[0131] Each of the transform blocks obtained above may then be subject to a primary transform. The primary transform essentially moves the residual in a transform block from spatial domain to frequency domain. In some implementation of the actual primary transform, in order to support the example extended coding block partitions above, multiple transform sizes (ranging from 4-point to 64-point for each dimension of the two dimensions) and

transform shapes (square; rectangular with width/height ratio's 2:1/1:2, and 4:1/1:4) may be allowed.

[0132] Turning to the actual primary transform, in some example implementations, a 2-D transform process may involve a use of hybrid transform kernels (which, for example, may be composed of different 1-D transforms for each dimension of the coded residual transform block). Example 1-D transform kernels may include but are not limited to: a) 4-point, 8-point, 16-point, 32-point, 64-point DCT-2; b) 4-point, 8-point, 16-point asymmetric DST's (DST-4, DST-7) and their flipped versions; c) 4-point, 8-point, 16-point, 32-point identity transforms. Selection of transform kernels to be used for each dimension may be based on a rate-distortion (RD) criterion. For example, the basis functions for the DCT-2 and asymmetric DST's that may be implemented are listed in Table 1.

Table 1: Example primary transform basis functions (DCT-2, DST-4 and DST-7 for N-point input).

Transform Type	Basis function $T_i(j)$, $i, j=0, 1, \dots, N-1$
DCT-2	$T_i(j) = \omega_0 \cdot \sqrt{\frac{2}{N}} \cdot \cos\left(\frac{\pi \cdot i \cdot (2j + 1)}{2N}\right)$ $\text{where } \omega_0 = \begin{cases} \sqrt{\frac{2}{N}} & i = 0 \\ 1 & i \neq 0 \end{cases}$
DST-4	$T_i(j) = \sqrt{\frac{2}{N}} \cdot \sin\left(\frac{\pi \cdot (2i + 1) \cdot (2j + 1)}{4N}\right)$
DST-7	$T_i(j) = \sqrt{\frac{4}{2N + 1}} \cdot \sin\left(\frac{\pi \cdot (2i + 1) \cdot (j + 1)}{2N + 1}\right)$

[0133] In some example implementations, the availability of hybrid transform kernels for a particular primary transform implementation may be based on the transform block size and prediction mode. An example dependency is listed in Table 2. For a chroma component, the transform type selection may be performed in an implicit way. For example, for intra prediction residuals, the transform type may be selected according to the intra prediction mode, as specified in Table 3. For inter prediction residuals, the transform type for chroma blocks may be selected according to the transform type selection of the co-located luma blocks. Therefore, for chroma component, there is no transform type signaling in the bitstream.

Table 2: AV1 hybrid transform kernels and their availability based on prediction modes and block sizes. Here $\rightarrow$ and $\downarrow$ denote the horizontal and vertical dimensions; $\checkmark$ and $\times$ denotes the availability of a kernel for that block size & prediction mode.

Transform Types	Description	Prediction mode	
		Intra	Inter
DCT_DCT	DCT ↓ and →	✓ (all block sizes)	✓ (all block sizes)
ADST_DCT	ADST ↓; DCT →	✓ (block size ≤ 16×16)	✓ (block size ≤ 16×16)
DCT_ADST	DCT ↓; ADST →		
ADST_ADST	ADST ↓ and →		
FLIPADST_DCT	FLIPADST ↓; DCT →	✗	✓ (block size ≤ 16×16)
DCT_FLIPADST	DCT ↓; FLIPADST →		
FLIPADST_FLIPADST	FLIPADST ↓ and →		
ADST_FLIPADST	ADST ↓; FLIPADST →		
FLIPADST_ADST	FLIPADST ↓; ADST →		
IDTX	IDTX ↓ and →	✓ (block size ≤ 16×16)	✓ (block size ≤ 32×32)
V_DCT	DCT ↓; IDTX →	✓ (block size < 16×16)	✓ (block size ≤ 16×16)
H_DCT	IDTX ↓; DCT →		
V_ADST	ADST ↓; IDTX →	✗	✓ (block size < 16×16)
H_ADST	IDTX ↓; ADST →		
V_FLIPADST	FLIPADST ↓; IDTX →	✗	✓ (block size < 16×16)
H_FLIPADST	IDTX ↓; FLIPADST →		

Table 3: Transform type selection for chroma component intra prediction residuals.

Intra prediction	Vertical Transform	Horizontal Transform
DC_PRED	DCT	DCT
V_PRED	ADST	DCT

H_PRED	DCT	ADST
D45_PRED	DCT	DCT
D135_PRED	ADST	ADST
D113_PRED	ADST	DCT
D157_PRED	DCT	ADST
D203_PRED	DCT	ADST
D67_PRED	ADST	DCT
SMOOTH_PRED	ADST	ADST
SMOOTH_V_PRED	ADST	DCT
SMOOTH_H_PRED	DCT	ADST
PAETH_PRED	ADST	ADST

[0134] In some implementation, secondary transform on the primary transform coefficients may be performed. For example, LFNST (low-frequency non-separable transform), which is known as reduced secondary transform may be applied between forward primary transform and quantization (at encoder) and between de-quantization and inverse primary transform (at decoder side), as shown in FIG. 14, to further decorrelate the primary transform coefficients. In essence, LFNST may take a portion of the primary transform coefficient, e.g., the low frequency portion (hence “reduced” from the full set of primary transform coefficients of the transform block) to proceed to secondary transform. In an example LFNST, 4 x 4 non-separable transform or 8 x 8 non-separable transform may be applied according to transform block size. For example, 4 x 4 LFNST may be applied for small transform blocks (e.g., min (width, height) < 8) whereas 8 x 8 LFNST may be applied for larger transform blocks (e.g., min (width, height) > 8). For example, if an 8 x 8 transform block is subject to 4 x 4 LFNST, then only the low frequency 4 x 4 portion of the 8 x 8 primary transform coefficients is further undergo secondary transform.

[0135] As specifically shown in FIG. 14, a transform block may be 8 x 8 (or 16 x 16). Thus, the forward primary transform 1402 of the transform block yields a 8 x 8 (or 16 x 16) primary transform coefficient matrix 1404, where each square unit represent a 2 x 2 (or 4 x 4) portion. The input to the forward LFNST, for example, may not be the entire 8 x 8 (or 16 x 16) primary transform coefficients. For example, a 4 x 4 (or 8 x 8) LFNST may be used for secondary transform. As such, only the 4 x 4 (or 8 x 8) low frequency primary transform coefficients of the primary transform coefficient matrix 1404, as indicated in the shaded portion (upper left) 1406 may be used as input to the LFNST. The remaining portions of the primary transform coefficient matrix may not be subject to secondary transform. As such, after the

secondary transform, the portion of the primary transform coefficients subject to the LFNST becomes the secondary transform coefficients whereas the remaining portions not subject to LFNST (e.g., the unshaded portions of the matrix 1404) maintain the corresponding primary transform coefficients. In some example implementations, the remaining portion not subject to secondary transform may be all set to zero coefficient.

[0136] An example for application of a non-separable transform used in LFNST, is described below. To apply an example 4 x 4 LFNST, the 4 x 4 input block X (representing, e.g., the 4 x 4 low-frequency portion of a primary transform coefficient block such as the shaded portion 1406 of the primary transform matrix 1404 of FIG. 14) may be denoted as:

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

[0137] This 2-D input matrix may be first linearized or scanned to a vector $\vec{X}$ in an example order:

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

[0138] The non-separable transform for the 4 x 4 LFNST may then be calculated as $\vec{F} = T \cdot \vec{X}$, where $\vec{F}$ indicates the output transform coefficient vector, and T is a 16 x 16 transform matrix. The resulting 16 x 1 coefficient vector $\vec{F}$ is subsequently reverse scanned as 4 x 4 block using the scanning order for that block (e.g., horizontal, vertical or diagonal). The coefficients with smaller index may be placed with the smaller scanning index in the 4 x 4 coefficient block. In such a manner, redundancy in a primary transform coefficients X may be further exploit via the second transform T , thereby providing additional compression enhancement.

[0139] The example LFNST above is based on a direct matrix multiplication approach to apply non-separable transform so that it is implemented in a single pass without multiple iterations. In some further example implementations, the dimension of the non-separable transform matrix (T) for the example 4 x 4 LFNST may be further reduced to minimize computational complexity and memory space requirement for storing the transform coefficients. Such implementations may be referred to as reduced non-separate transform (RST). In more detail, the main idea of the RST is to map an $\mathbf{N}$ ($\mathbf{N}$ is 4 x 4 =16 in the example above, but may be equal to 64 for 8 x 8 blocks) dimensional vector to an $\mathbf{R}$ dimensional vector in a different space, where $\mathbf{N}/\mathbf{R}$ ($\mathbf{R} < \mathbf{N}$) represents the dimension reduction factor. Hence, instead of $\mathbf{N} \times \mathbf{N}$ transform matrix, RST matrix becomes an $\mathbf{R} \times \mathbf{N}$ matrix as follows:

$$T_{R \times N} = \begin{bmatrix} t_{11} & t_{12} & t_{13} & \dots & t_{1N} \\ t_{21} & t_{22} & t_{23} & \dots & t_{2N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ t_{R1} & t_{R2} & t_{R3} & \dots & t_{RN} \end{bmatrix} \quad (4)$$

[0140] where the **R** rows of the transform matrix are reduced **R** basis of the **N** dimensional space. The transformation thus converts an input vector or **N** dimension to an output vector of reduced **R** dimension. As such, and as shown in FIG. 14, the secondary transform coefficients 1408 transformed from the primary coefficients 1406 is reduced by a factor or **N/R** in dimension. The three squares around 1408 in FIG. 14 may be zero-padded.

[0141] The inverse transform matrix for RTS may be the transpose of its forward transform. For an example 8 x 8 LFNST (contrasted with the 4 x 4 LFNST above, for a more diverse description here), an example reduction factor of 4 may be applied, and thus a 64 x 64 direct non-separable transform matrix is accordingly reduced to 16 x 64 direct matrix. Further, in some implementations, a portion rather than an entirety of the input primary coefficients may be linearized into the input vector for the LFNST. For example, only a portion of the example 8 x 8 input primary transform coefficients may be linearized into the X vector above. For a particular example, out of the four 4 x 4 quadrants of the 8 x 8 primary transform coefficient matrix, the bottom right (high frequency coefficients) may be left out, and only the other three quadrants are linearized into a 48 x 1 vector using a predefined scan order rather than a 64 x 1 vector. In such implementations, the non-separable transform matrix may be further reduced from 16 x 64 to 16 x 48.

[0142] Hence, an example reduced 48x16 inverse RST matrix may be used at the decoder side to generate the top-left, top-right, and bottom-left 4 x 4 quadrants of the 8 x 8 core (primary) transform coefficients. Specifically, when the further reduced 16 x 48 RST matrices are applied instead of the 16 x 64 RST with the same transform set configuration, the non-separable secondary transformation would take as input the vectorized 48 matrix elements from three 4 x 4 quadrant blocks of the 8 x 8 primary coefficient block excluding right-bottom 4 x 4 block. In such implementations the omitted right-bottom 4 x 4 primary transform coefficient would be ignored in the secondary transformation. This further reduced transformation, would convert a vector of 48 x 1 into an output vector of 16 x 1, which is reverse scanned into a 4 x 4 matrix to fill 1408 of FIG. 14. The three squares of the secondary transform coefficients surrounding 1408 may be zero padded.

[0143] With the help of the such reduction of dimensions in the RST, memory usage for storing all LFNST matrices is reduced. In the above example, the memory usage, for

example, may be reduced from 10 KB to 8 KB with reasonably insignificant performance drop compared to the implementation without dimension reduction.

[0144] In some implementations, in order to reduce complexity, LFNST may be further restricted to be applicable only if all coefficients outside the primary transform coefficient portion to be subject to LFNST (e.g., outside of the 1406 portion of 1404 in FIG. 14) are non-significant. Hence, all primary-only transform coefficients (e.g., the unshaded portion of the primary coefficient matrix 1404 of FIG. 14) may be near zero when LFNST is applied. Such a restriction allows for a conditioning of the LFNST index signalling on the last-significant position, and hence avoids some extra coefficient scanning, which may be needed for checking for significant coefficients at specific positions when this restriction is not applied. In some implementations, the worst-case handling of LFNST (in terms of multiplications per pixel) may restrict the non-separable transforms for 4×4 and 8×8 blocks to 8×16 and 8×48 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 $4 \times N$ and $N \times 4$ and $N > 8$, the restriction above implies that the LFNST is now applied only once to the top-left 4×4 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 an encoder perspective, the quantization of coefficients can be simplified when LFNST transforms are tested. A rate-distortion optimized quantization (RDO) has to be done at maximum for the first 16 coefficients (in scan order), the remaining coefficients may be enforced to be zero.

[0145] In some example implementations, the RST kernels available may be specified as a number of transform sets with each transform sets including a number of non-separable transform matrices. For example, there may be a total of 4 transform sets and 2 non-separable transform matrices (kernels) per transform set for used in LFNST. These kernels may be pre-trained offline and they are thus data driven. The offline trained transform kernels may be stored in memory or hard coded in an encoding or decoding device for use during encoding/decoding process. A selection of the transform set during encoding or decoding process may be determined by the intra prediction mode. A mapping from the intra prediction modes to the transform sets may be pre-defined. An example of such predefined mapping is shown in Table 4. For example, as shown in Table 4, if one of three Cross-Component Linear Model (CCLM) modes (INTRA_LT_CCLM, INTRA_T_CCLM or INTRA_L_CCLM) is used for the current block (i.e., $81 \leq \text{predModeIntra} \leq 83$), transform set 0 may be selected for the current chroma block. For each transform set, the selected non-

separable secondary transform candidate may be further specified by the explicitly signalled LFNST index. For example, the index may be signalled in a bit-stream once per intra CU after transform coefficients.

Table 4: Transform selection table

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

[0146] Because LFNST is restricted to be applicable only if all coefficients outside the first coefficient sub-group or portion are non-significant in the example implementations above, LFNST index coding depends on the position of the last significant coefficient. In addition, the LFNST index may be context coded but does not depend on intra prediction mode, and only the first bin may be context coded. Furthermore, LFNST may be 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 may be signaled separately. For inter slice (the dual tree is disabled), a single LFNST index may be signaled and used for both Luma and Chroma.

[0147] In some example implementations, when Intra Sub-Partitioning (ISP) mode is selected, LFNST may be disabled and RST index may not be signaled, because performance improvement is likely to be marginal even if RST is applied to every feasible partition block. Furthermore, disabling RST for ISP-predicted residual could reduce encoding complexity. In some further implementations, LFNST may also be disabled and the RST index may not be signaled when Multiple linear regression Intra Prediction (MIP) mode is selected.

[0148] Considering that a large CU greater than 64 x 64 (or any other predefined sizes representing the maximum transform block size) is implicitly split (e.g., TU tiling) due to the existing maximum transform size restriction (e.g., 64 x 64), an LFNST index search could increase data buffering by four times for a certain number of decode pipeline stages. Therefore, in some implementations, the maximum size that LFNST is allowed may be

restricted to, for example, 64 x 64. In some implementations, LFNST may be enabled with DCT2 as primary transform only.

[0149] In some other implementations, intra secondary transform (IST) is provided for luma component by defining, e.g., 12 sets of secondary transforms, with, e.g., 3 kernels in each set. An intra mode dependent index may be used for transform set selection. The kernel selection within a set may be based on a signaled syntax element. The IST may be enabled when either DCT2 or ADST is used as both horizontal and vertical primary transform. In some implementations, according to the block size, a 4 x 4 non-separable transform or 8 x 8 non-separable transform can be selected. If $\min(\text{tx_width}, \text{tx_height}) < 8$ the 4 x 4 IST can be selected. For larger blocks 8 x 8 IST can be used. Here tx_width & tx_height correspond to transform block width & height, respectively. The input to IST may be low frequency primary transform coefficients in a zig-zag scan order.

[0150] The various transforms in video coding or decoding processes, e.g., either primary transforms of samples in a residual blocks or secondary transforms of blocks of primary transform coefficients processes may not be every efficient in capturing directional texture patterns, such as edges which are 45-degree direction (e.g., directions that are substantially away from the horizontal or vertical directions), when using separable transform schemes only. As described above, in some example implementations, one or more non-separable transform designs may be used for secondary transform of primary transform coefficient.

[0151] The transform block partitioning and the transform types to apply to the partitioned transform blocks may be inter related. For example, certain transform types may be more suitable for a particular partition type. For example, the transform partitioning scheme illustrated in FIG. 16 and described above provides non-recursive partition types compared with recursive partitioning, such as the partition described earlier in FIG. 13. If all available transform types are allowed for transform blocks partitioned under all available partition patterns (e.g., the transform partition types in FIG. 16), the encoder would need to perform an optimization in a large parameter space when determining which transform partition type to use for obtaining transform blocks and what transform types to use for each partitioned transform block. In practice, a certain set of transform types may be generally more suitable than other transform types for a particular type of transform partition type. In the various implementations described below, an interaction between transform partition types and transform types may be considered and used to arrive at schemes to restrict the allowed transform types for particular partition types, and likewise, to restrict allowed

partition types for particular transform types. Such implementations may allow for a reduction of optimization space for an encoder when determining a transform partition pattern and transform type selection for each partitioned transform block, particular in the situation where non-recursive transform partitioning is employed.

[0152] These example implementations may be used separately or combined in any order or in any manner. In the discourse above and below, the terms “coded block”, “coding block”, and the like may be used to refer to a picture unit that prediction or transform is performed. A coding block may be a luma coding block or may be a chroma coding block. In some situations, coded/coding block may refer to a prediction block. The term “block size” is used to refer to either the width or height, or maximum value of width and height, or minimum of width and height, or area size (width * height), or aspect ratio (width:height, or height:width) of the coding block.

[0153] Multiple Candidate Primary Transform Types

[0154] In one embodiment, there may be multiple candidate primary transform types for a block. The block may include a transform block resulting from a partition. The selection and/or signaling of the primary transform type may be restricted to a predefined set of transform partition types. The predefined set of transform partition types may be a subset of a larger set of available transform partition types (such as the full set of partition types in FIG. 16). In other words, only when the transform partition type of a block belongs to the predefined set of transform partition types, a selection of primary transform is made and signaled. Otherwise, for example, if the transform partition type is of other types, a default transform type may be used rather than selected and signaled.

[0155] In one implementation, the primary transform type may include at least one of: a Discrete Cosine Transform (DCT) type 1 through DCT type 8; an Asymmetric Discrete Sine Transform (ADST); a Discrete Sine Transform (DST) type 1 through DST type 8; a Line Graph Transform (LGT); or a Karhunen-Loeve Transform (KLT).

[0156] In one implementation, the predefined set of transform partition types only includes PARTITION_NONE out of the various transform partition types of, e.g., FIG. 16, i.e., the transform block size is equal to the prediction block (or coding block) size. Therefore, only when the transform type belongs to this predefined set, a primary transform type may be selected and/or signaled.

[0157] In one implementation, the number of partitions may also be considered for determining the predefined set of transform partition types. For example, only when the number of partitions of a particular transform partition type is less than or equal to a

predefined threshold, it would be considered as part of the predefined set of transform partition types for which a primary transform type may be selected and/or signaled. In one implementation, the predefined threshold may be an integer from 1 to 16.

[0158] Multiple Candidate Secondary Transform Types

[0159] In one embodiment, there may be multiple candidate secondary transform types for a block. The block may include a transform block resulting from a partition. The selection and/or signaling of the secondary transform type may only apply for a predefined set of transform partition types. The predefined set of transform partition types may be a subset of a larger set of available transform partition types (such as the full set of transform partition types in FIG. 16). In other words, only when the transform partition type of a block belongs to the predefined set of transform partition types, a selection of secondary transform is made and signaled. Otherwise, for example, if the transform partition type is of other types, a default secondary transform type may be used rather than selected and signaled, or no secondary transformation may be performed.

[0160] In one implementation, the secondary transform type may include a KLT. The KLT may be configured with different kernels.

[0161] In one implementation, the predefined set of transform partition types may only includes PARTITION_NONE, out of the various transform partition types of, e.g., FIG. 16, i.e., the transform block size is equal to the prediction block (or coding block) size. Therefore, only when the transform type belongs to this predefined set, a secondary transform type may be selected and/or signaled.

[0162] In one implementation, the number of partitions may also be considered for determining the predefined set of transform partition types. For example, only when the number of partitions of a particular transform partition type is less than or equal to a predefined threshold, it would be considered as part of the predefined set of transform partition types for which a secondary transform type may be selected and/or signaled. In one implementation, the predefined threshold may be an integer from 1 to 16.

[0163] In one implementation, the selection and/or signaling of a secondary transform type may be based on a combination of transform partition type and primary transform type. The combination may include a transform partition type in a predefined set of transform partition types and a primary transform type in a predefined set of transform types. For example, the secondary transform type may only need to be selected and/or signaled when the transform partition type is PARTITION_NONE, and the primary transform types used for the block is a DCT or an ADST. Otherwise, for example, if the transform partition type is of

other types, a default secondary transform type may be used rather than selected and signaled or no secondary transformation may be performed.

[0164] Transform Related Signaling

[0165] In this disclosure, various signaling mechanisms are disclosed aiming for increasing signaling efficiency, with the order of transform related syntax elements/parameters taken into consideration.

[0166] In one embodiment, the transform partition type information may be signaled before primary/secondary transform type selection information. The primary/secondary transform type selection only needs to be signaled when the transform partition belongs to a predefined set of transform partition types, for example, PARTITION_NONE. Otherwise, if the transform partition does not belong to the predefined set of transform partition types, the primary/secondary transform type selection may not need to be signaled. Instead, the primary/secondary transform type may be derived as a predefined default transform type.

[0167] In one embodiment, the primary/secondary transform type selection information may be signaled before the transform partition type information. In this case, the selection and/or signaling of the transform partition type may depend on the primary/secondary transform type selection information.

[0168] In one implementation, only when primary transform type belongs to a predefined set of transform types, the transform partition type information may need to be signaled. Otherwise, the transform partition type information may not need to be signaled. As an example, the predefined set of transform types may include but not limited to, a DCT type 1 to DCT type 8, an ADST, a DST type 1 to DST type 8, an LGT, and a KLT.

[0169] In one implementation, only when secondary transform type belongs to a predefined set of transform types, the transform partition type information may need to be signaled. As an example, the predefined set of transform types may include but not limited to, but not limited to a specific KLT with a kernel that is associated with a predefined KLT index. Otherwise, if the secondary transform type does not belong to the predefined set of transform types, the transform partition type information may not need to be signaled. Instead, the transform partition type information may be derived as a predefined default transform partition type, such as PARTITION_NONE.

[0170] FIG. 17 shows an exemplary method 1700 for decoding video data. The method 1700 may include a portion or all of the following step: step 1710, receiving a coded video bitstream for a data block; step 1720, extracting, from the coded video bitstream, a transform partition type associated with the data block; step 1730, in response to the

transform partition type belonging to a subset of a predefined set of transform partition types each specifying a split pattern for dividing the data block into transform blocks: extracting, as signaled in coded video bitstream, a transform type of a transform associated with a transform block split from the data block, wherein the transform type belongs to a first predefined set of transform types; and performing inverse transform on the transform block according to the transform type.

[0171] In embodiments of this disclosure, any steps and/or operations may be combined or arranged in any amount or order, as desired. Two or more of the steps and/or operations may be performed in parallel.

[0172] Embodiments in the disclosure may be used separately or combined in any order. Further, each of the methods (or embodiments), an encoder, and a decoder may be implemented by processing circuitry (e.g., one or more processors or one or more integrated circuits). In one example, the one or more processors execute a program that is stored in a non-transitory computer-readable medium. Embodiments in the disclosure may be applied to a luma block or a chroma block.

[0173] The techniques described above, can be implemented as computer software using computer-readable instructions and physically stored in one or more computer-readable media. For example, FIG. 18 shows a computer system (1800) suitable for implementing certain embodiments of the disclosed subject matter.

[0174] The computer software can be coded using any suitable machine code or computer language, that may be subject to assembly, compilation, linking, or like mechanisms to create code comprising instructions that can be executed directly, or through interpretation, micro-code execution, and the like, by one or more computer central processing units (CPUs), Graphics Processing Units (GPUs), and the like.

[0175] The instructions can be executed on various types of computers or components thereof, including, for example, personal computers, tablet computers, servers, smartphones, gaming devices, internet of things devices, and the like.

[0176] The components shown in FIG. 18 for computer system (1800) are exemplary in nature and are not intended to suggest any limitation as to the scope of use or functionality of the computer software implementing embodiments of the present disclosure. Neither should the configuration of components be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the exemplary embodiment of a computer system (1800).

[0177] Computer system (1800) may include certain human interface input devices. Such a human interface input device may be responsive to input by one or more human users through, for example, tactile input (such as: keystrokes, swipes, data glove movements), audio input (such as: voice, clapping), visual input (such as: gestures), olfactory input (not depicted). The human interface devices can also be used to capture certain media not necessarily directly related to conscious input by a human, such as audio (such as: speech, music, ambient sound), images (such as: scanned images, photographic images obtain from a still image camera), video (such as two-dimensional video, three-dimensional video including stereoscopic video).

[0178] Input human interface devices may include one or more of (only one of each depicted): keyboard (1801), mouse (1802), trackpad (1803), touch screen (1810), data-glove (not shown), joystick (1805), microphone (1806), scanner (1807), camera (1808).

[0179] Computer system (1800) may also include certain human interface output devices. Such human interface output devices may be stimulating the senses of one or more human users through, for example, tactile output, sound, light, and smell/taste. Such human interface output devices may include tactile output devices (for example tactile feedback by the touch-screen (1810), data-glove (not shown), or joystick (1805), but there can also be tactile feedback devices that do not serve as input devices), audio output devices (such as: speakers (1809), headphones (not depicted)), visual output devices (such as screens (1810) to include CRT screens, LCD screens, plasma screens, OLED screens, each with or without touch-screen input capability, each with or without tactile feedback capability—some of which may be capable to output two dimensional visual output or more than three dimensional output through means such as stereographic output; virtual-reality glasses (not depicted), holographic displays and smoke tanks (not depicted)), and printers (not depicted).

[0180] Computer system (1800) can also include human accessible storage devices and their associated media such as optical media including CD/DVD ROM/RW (1820) with CD/DVD or the like media (1821), thumb-drive (1822), removable hard drive or solid state drive (1823), legacy magnetic media such as tape and floppy disc (not depicted), specialized ROM/ASIC/PLD based devices such as security dongles (not depicted), and the like.

[0181] Those skilled in the art should also understand that term “computer readable media” as used in connection with the presently disclosed subject matter does not encompass transmission media, carrier waves, or other transitory signals.

[0182] Computer system (1800) can also include an interface (1854) to one or more communication networks (1855). Networks can for example be wireless, wireline, optical.

Networks can further be local, wide-area, metropolitan, vehicular and industrial, real-time, delay-tolerant, and so on. Examples of networks include local area networks such as Ethernet, wireless LANs, cellular networks to include GSM, 3G, 4G, 5G, LTE and the like, TV wireline or wireless wide area digital networks to include cable TV, satellite TV, and terrestrial broadcast TV, vehicular and industrial to include CAN bus, and so forth. Certain networks commonly require external network interface adapters that attached to certain general-purpose data ports or peripheral buses (1849) (such as, for example USB ports of the computer system (1800)); others are commonly integrated into the core of the computer system (1800) by attachment to a system bus as described below (for example Ethernet interface into a PC computer system or cellular network interface into a smartphone computer system). Using any of these networks, computer system (1800) can communicate with other entities. Such communication can be uni-directional, receive only (for example, broadcast TV), uni-directional send-only (for example CANbus to certain CANbus devices), or bi-directional, for example to other computer systems using local or wide area digital networks. Certain protocols and protocol stacks can be used on each of those networks and network interfaces as described above.

[0183] Aforementioned human interface devices, human-accessible storage devices, and network interfaces can be attached to a core (1840) of the computer system (1800).

[0184] The core (1840) can include one or more Central Processing Units (CPU) (1841), Graphics Processing Units (GPU) (1842), specialized programmable processing units in the form of Field Programmable Gate Areas (FPGA) (1843), hardware accelerators for certain tasks (1844), graphics adapters (1850), and so forth. These devices, along with Read-only memory (ROM) (1845), Random-access memory (1846), internal mass storage such as internal non-user accessible hard drives, SSDs, and the like (1847), may be connected through a system bus (1848). In some computer systems, the system bus (1848) can be accessible in the form of one or more physical plugs to enable extensions by additional CPUs, GPU, and the like. The peripheral devices can be attached either directly to the core's system bus (1848), or through a peripheral bus (1849). In an example, the screen (1810) can be connected to the graphics adapter (1850). Architectures for a peripheral bus include PCI, USB, and the like.

[0185] CPUs (1841), GPUs (1842), FPGAs (1843), and accelerators (1844) can execute certain instructions that, in combination, can make up the aforementioned computer code. That computer code can be stored in ROM (1845) or RAM (1846). Transitional data can also be stored in RAM (1846), whereas permanent data can be stored for example, in the

internal mass storage (1847). Fast storage and retrieve to any of the memory devices can be enabled through the use of cache memory, that can be closely associated with one or more CPU (1841), GPU (1842), mass storage (1847), ROM (1845), RAM (1846), and the like.

[0186] The computer readable media can have computer code thereon for performing various computer-implemented operations. The media and computer code can be those specially designed and constructed for the purposes of the present disclosure, or they can be of the kind well known and available to those having skill in the computer software arts.

[0187] As a non-limiting example, the computer system having architecture (1800), and specifically the core (1840) can provide functionality as a result of processor(s) (including CPUs, GPUs, FPGA, accelerators, and the like) executing software embodied in one or more tangible, computer-readable media. Such computer-readable media can be media associated with user-accessible mass storage as introduced above, as well as certain storage of the core (1840) that are of non-transitory nature, such as core-internal mass storage (1847) or ROM (1845). The software implementing various embodiments of the present disclosure can be stored in such devices and executed by core (1840). A computer-readable medium can include one or more memory devices or chips, according to particular needs. The software can cause the core (1840) and specifically the processors therein (including CPU, GPU, FPGA, and the like) to execute particular processes or particular parts of particular processes described herein, including defining data structures stored in RAM (1846) and modifying such data structures according to the processes defined by the software. In addition or as an alternative, the computer system can provide functionality as a result of logic hardwired or otherwise embodied in a circuit (for example: accelerator (1844)), which can operate in place of or together with software to execute particular processes or particular parts of particular processes described herein. Reference to software can encompass logic, and vice versa, where appropriate. Reference to a computer-readable media can encompass a circuit (such as an integrated circuit (IC)) storing software for execution, a circuit embodying logic for execution, or both, where appropriate. The present disclosure encompasses any suitable combination of hardware and software.

[0188] While this disclosure has described several exemplary embodiments, there are alterations, permutations, and various substitute equivalents, which fall within the scope of the disclosure. It will thus be appreciated that those skilled in the art will be able to devise numerous systems and methods which, although not explicitly shown or described herein, embody the principles of the disclosure and are thus within the spirit and scope thereof.

Appendix A: Acronyms

JEM: joint exploration model

VVC: versatile video coding

BMS: benchmark set

MV: Motion Vector

HEVC: High Efficiency Video Coding

SEI: Supplementary Enhancement Information

VUI: Video Usability Information

GOPs: Groups of Pictures

TUs: Transform Units,

PUs: Prediction Units

CTUs: Coding Tree Units

CTBs: Coding Tree Blocks

PBs: Prediction Blocks

HRD: Hypothetical Reference Decoder

SNR: Signal Noise Ratio

CPUs: Central Processing Units

GPUs: Graphics Processing Units

CRT: Cathode Ray Tube

LCD: Liquid-Crystal Display

OLED: Organic Light-Emitting Diode

CD: Compact Disc

DVD: Digital Video Disc

ROM: Read-Only Memory

RAM: Random Access Memory

ASIC: Application-Specific Integrated Circuit

PLD: Programmable Logic Device

LAN: Local Area Network

GSM: Global System for Mobile communications

LTE: Long-Term Evolution

CANBus: Controller Area Network Bus

USB: Universal Serial Bus

PCI: Peripheral Component Interconnect

FPGA: Field Programmable Gate Areas

SSD: solid-state drive
IC: Integrated Circuit
HDR: high dynamic range
SDR: standard dynamic range
JVET: Joint Video Exploration Team
MPM: most probable mode
WAIP: Wide-Angle Intra Prediction
CU: Coding Unit
PU: Prediction Unit
TU: Transform Unit
CTU: Coding Tree Unit
PDPC: Position Dependent Prediction Combination
ISP: Intra Sub-Partitions
SPS: Sequence Parameter Setting
PPS: Picture Parameter Set
APS: Adaptation Parameter Set
VPS: Video Parameter Set
DPS: Decoding Parameter Set
ALF: Adaptive Loop Filter
SAO: Sample Adaptive Offset
CC-ALF: Cross-Component Adaptive Loop Filter
CDEF: Constrained Directional Enhancement Filter
CCSO: Cross-Component Sample Offset
LSO: Local Sample Offset
LR: Loop Restoration Filter
AV1: AOMedia Video 1
AV2: AOMedia Video 2

What is claimed is:

1. A method for decoding video data in a decoder, the method comprising:
 - receiving a coded video bitstream for a data block;
 - extracting, from the coded video bitstream, a transform partition type associated with the data block; and
 - in response to the transform partition type belonging to a subset of a predefined set of transform partition types each specifying a split pattern for dividing the data block into transform blocks:
 - extracting, as signaled in coded video bitstream, a transform type of a transform associated with a transform block split from the data block, wherein the transform type belongs to a first predefined set of transform types; and
 - performing inverse transform on the transform block according to the transform type.
2. The method of claim 1, further comprising in response to the transform partition type not belonging to the subset of the predefined set of transform partition types:
 - determining the transform type associated with the transform block to be a predefined default transform type.
3. The method of claim 1, wherein the transform is a primary transform, and the first predefined set of transform types comprises:
 - Discrete Cosine Transform (DCT) type 1 through DCT type 8;
 - Asymmetric Discrete Sine Transform (ADST);
 - Discrete Sine Transform (DST) type 1 through DST type 8;
 - Line Graph Transform (LGT); and
 - Karhunen-Loeve Transform (KLT).

4. The method of claim 1, wherein the subset of the predefined set of transform partition types consist of PARTITION_NONE for no transform block partitioning.
5. The method of claim 1, wherein a number of transform partitions associated with each transform partition type in the subset of the predefined set of transform partition types is less than or equal to a predefined threshold.
6. The method of claim 5, wherein the predefined threshold comprises an integer from 1 to 16, inclusive.
7. The method of any of claims 1 to 6, wherein the transform is a secondary transform, and the first predefined set of transform types comprises a KLT.
8. The method of claim 7, wherein the subset of the predefined set of transform partition types consists of PARTITION_NONE for no transform block partitioning.
9. The method of claim 7, wherein a number of transform partitions associated with each transform partition type in the subset of the predefined set of transform partition types is less than or equal to a predefined threshold.
10. The method of claim 7, wherein the transform type further indicates that a type of a primary transform associated with the transform block belongs to a second predefined set of transform types.

11. The method of claim 10, wherein the subset of the predefined set of transform partition types consists of PARTITION_NONE for no transform block partitioning, and wherein the second predefined set of transform types consist of DCT and ADST.
12. A method for decoding video data, the method comprising:
- receiving a coded video bitstream for a data block;
 - extracting, from the coded video bitstream, a transform partition type associated with a data block of the video data;
 - in response to the transform partition type belonging to a subset of a predefined set of transform partition types, extracting, from the coded video bitstream, a transform type associated with a transform block of the data block; and
 - in response to the transform partition type not belonging to the predefined set of transform partition types, identifying the transform type for the data block by default.
13. The method of claim 12, wherein the subset of the predefined set of transform partition types consists of PARTITION_NONE for no transform block partitioning.
14. The method of claim 12, wherein the transform type comprises a primary transform type or a secondary transform type.
15. A method for decoding video data, the method comprising:
- receiving a coded video bitstream for a data block;
 - extracting, from the coded video bitstream, a transform type of a transform associated with a transform block of the data block; and
 - in response to the transform type belonging to a predefined set of transform types:

extracting, from the coded video bitstream, a transform partition type associated with the data block.

16. The method of claim 15, further comprising:

in response to the transform type not belonging to the predefined set of transform types:

determining the transform partition type associated with the data block to be a predefined default transform partition type, the predefined default transform partition type comprising PARTITION_NONE.

17. The method of claim 15, wherein the transform is a primary transform, and the predefined set of transform types comprises:

DCT type 2;

ADST;

DCT type 1 through DCT type 8;

DST type 1 through DST type 8;

LGT; or

KLT.

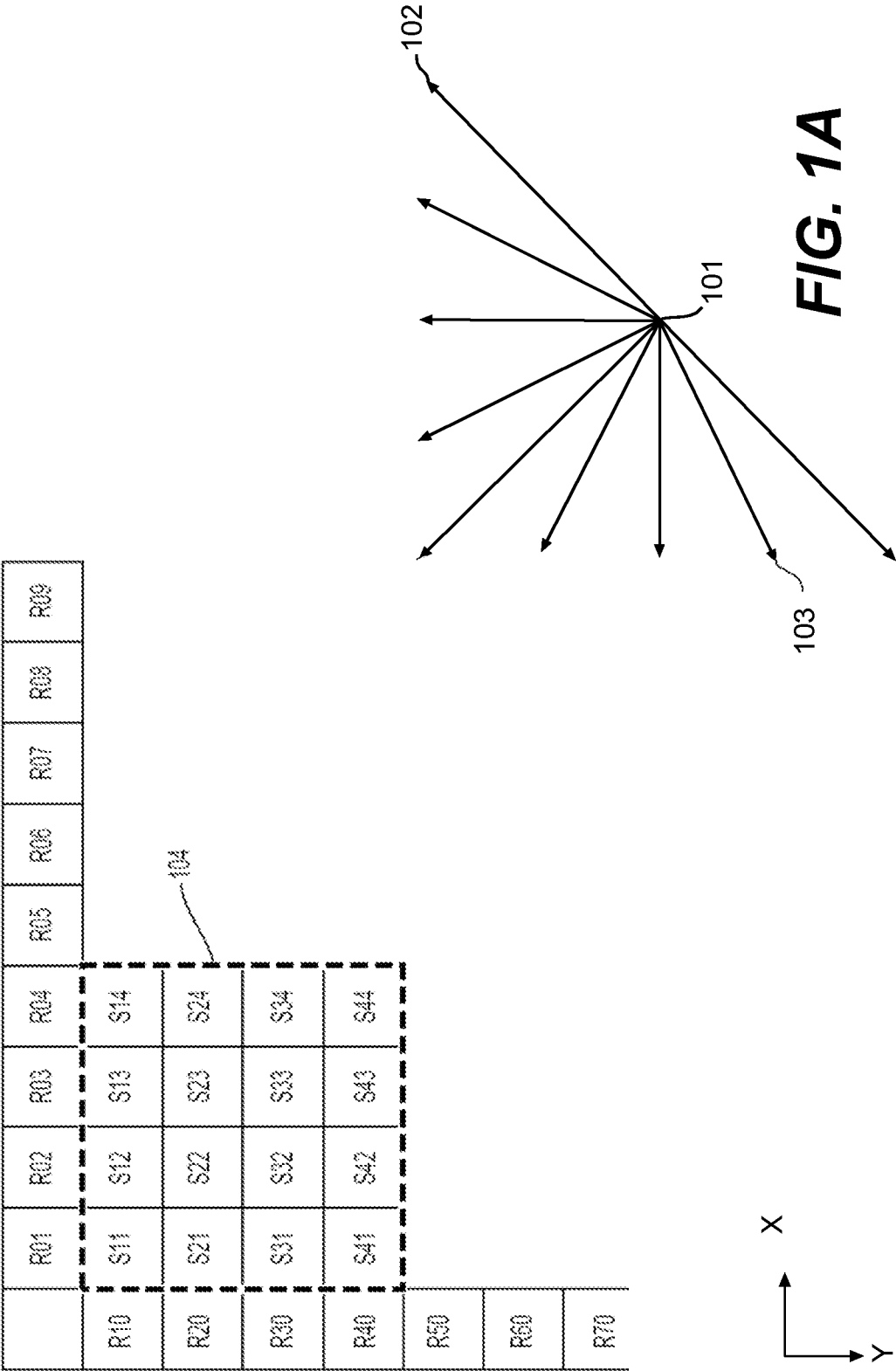
18. The method of claim 15, wherein the transform is a secondary transform, and the predefined set of transform types comprise a KLT with a kernel associated with a predefined KLT index.

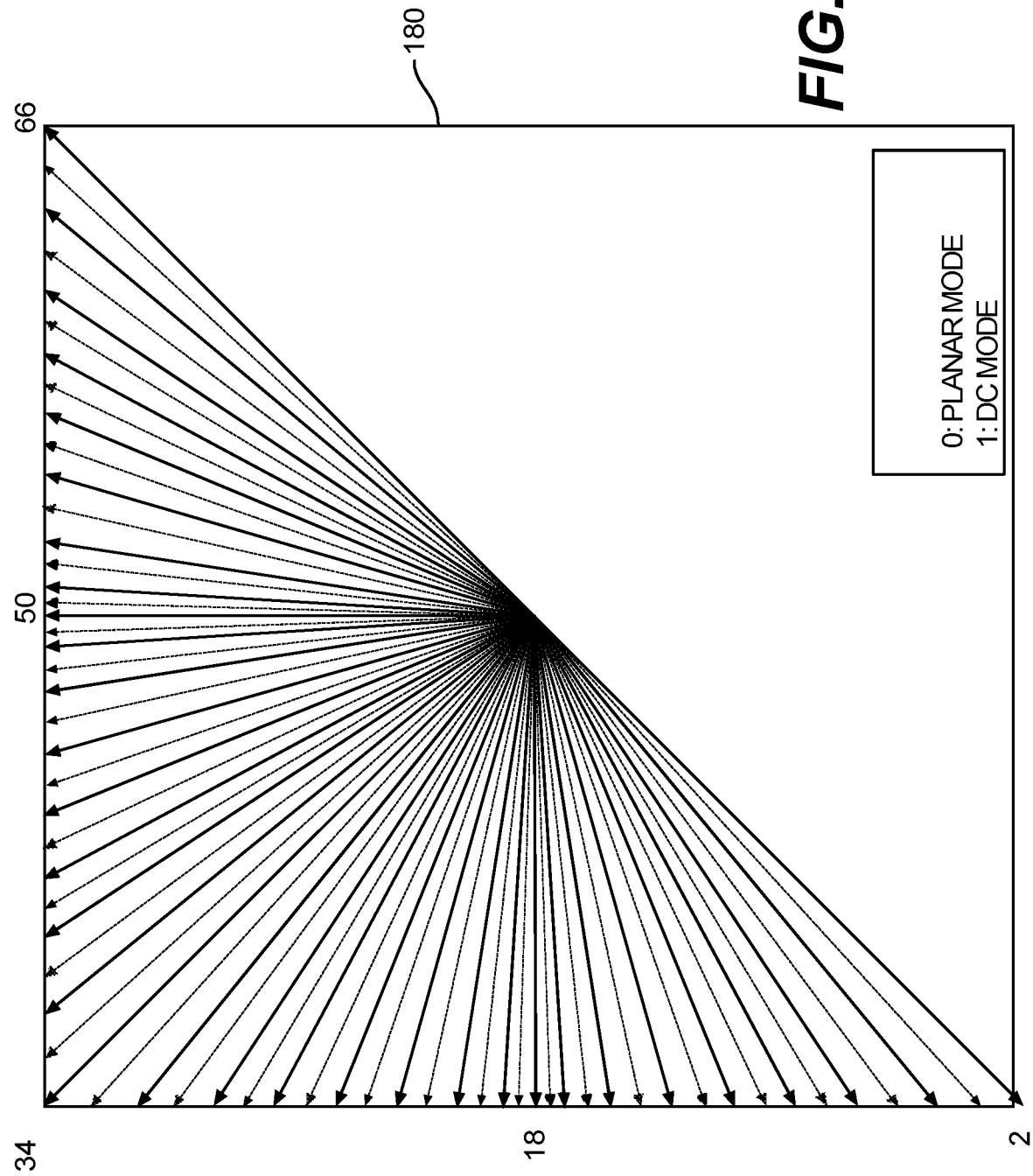
19. A device comprising a circuitry configured to implement the method of any of claims 1-6, or 12-18.

20. A device comprising a circuitry configured to implement the method of claim 7.

21. A computer program product comprising a non-transitory computer-readable program medium with computer code stored thereupon, the computer code, when executed by one or more processors, causing the one or more processors to implement the method of any of claims 1-6, or 12-18.

22. A computer program product comprising a non-transitory computer-readable program medium with computer code stored thereupon, the computer code, when executed by one or more processors, causing the one or more processors to implement the method of claim 7.





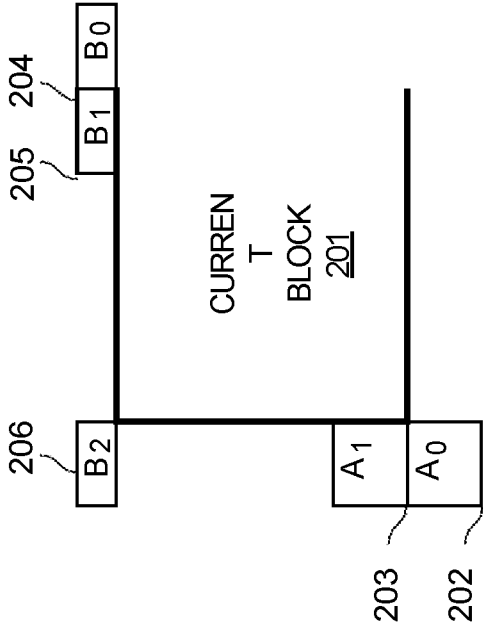


FIG. 2

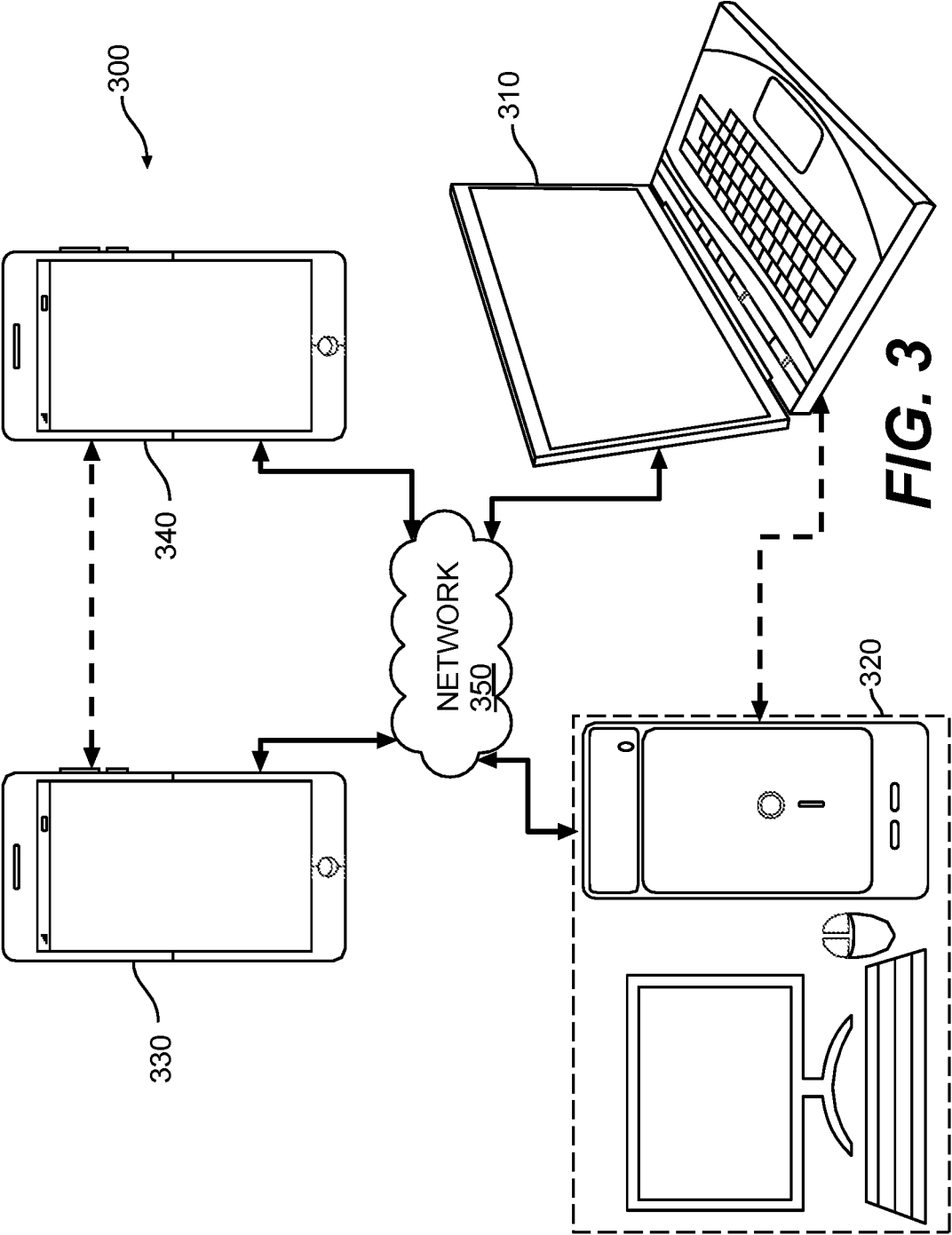


FIG. 3

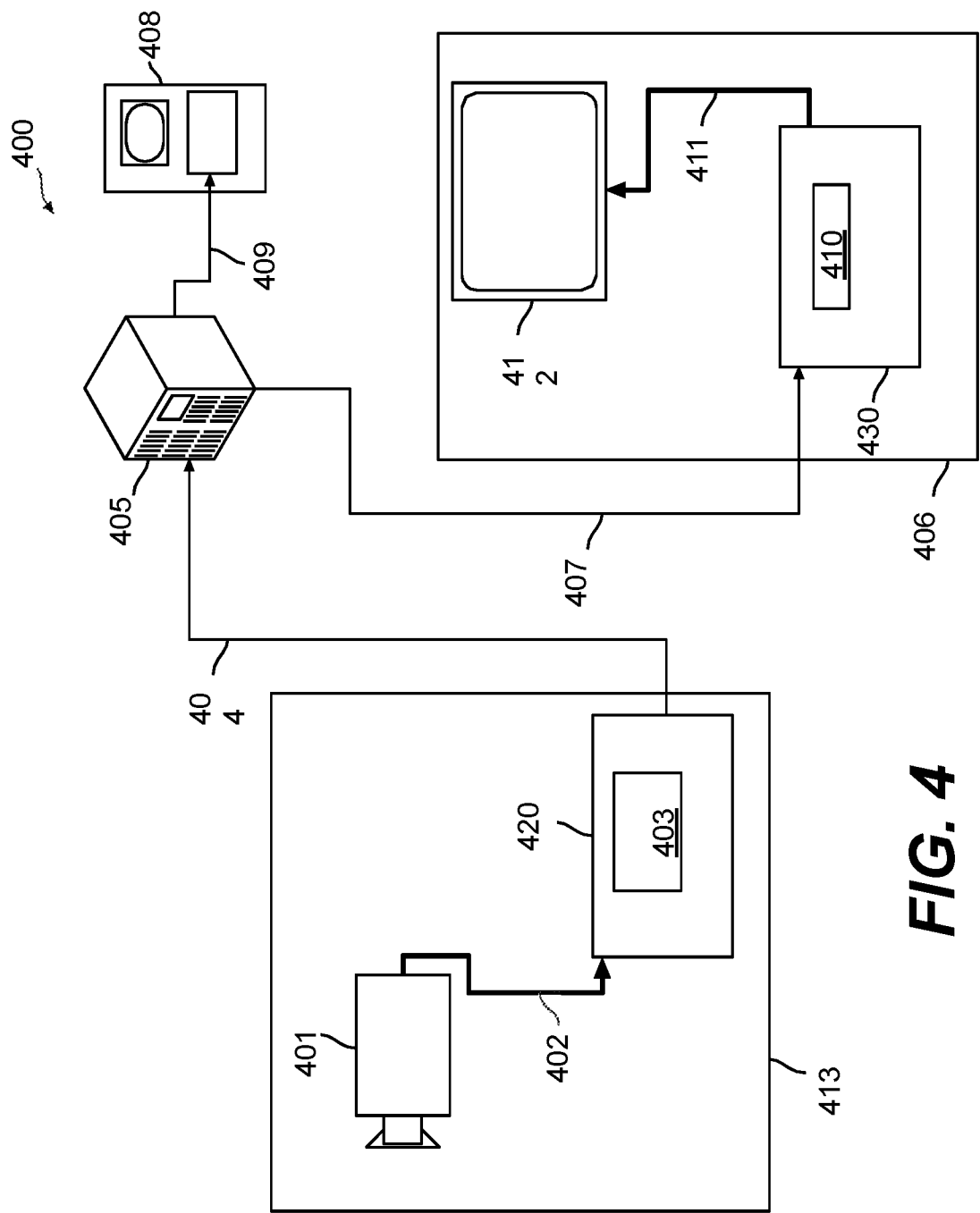


FIG. 4

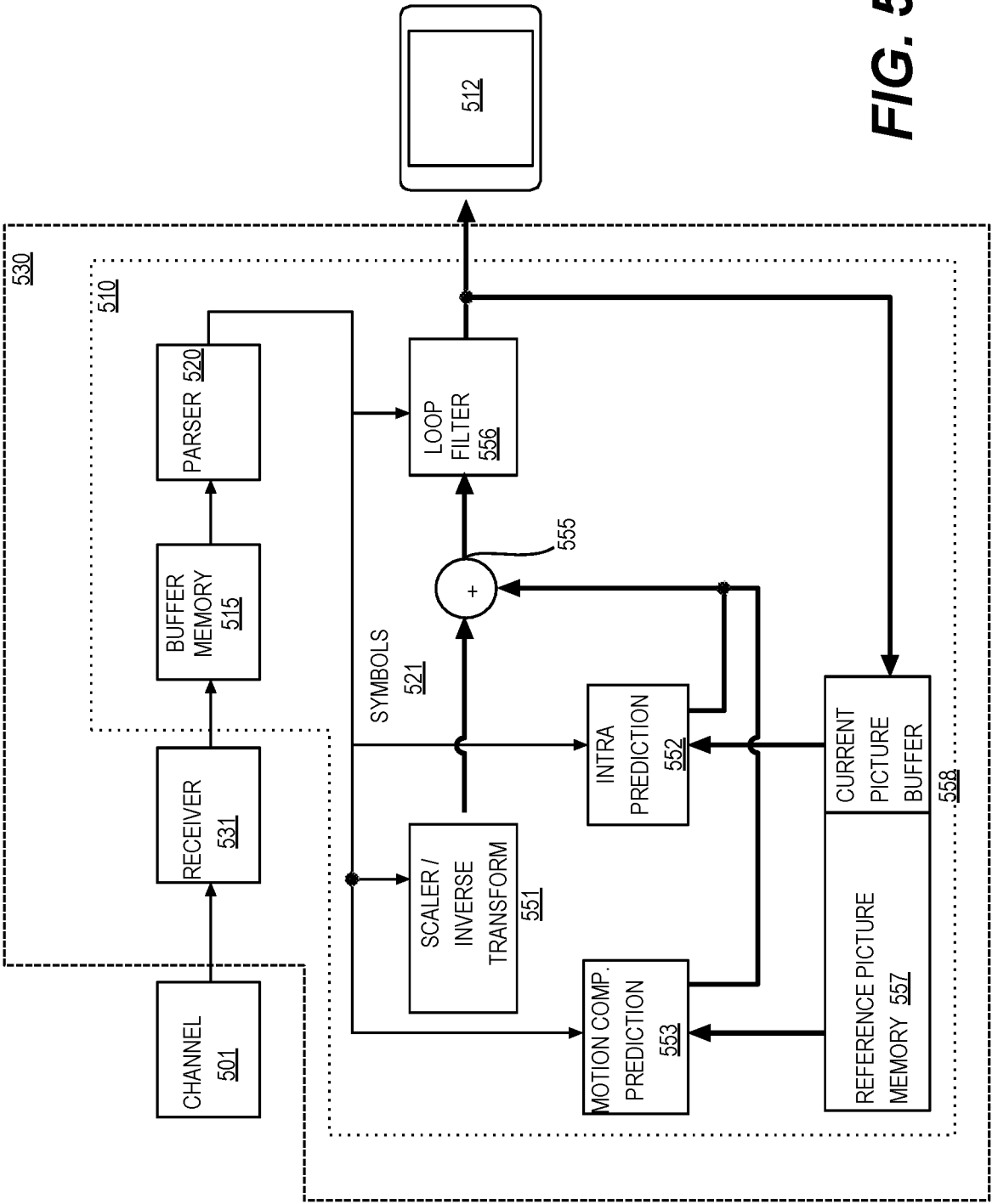


FIG. 5

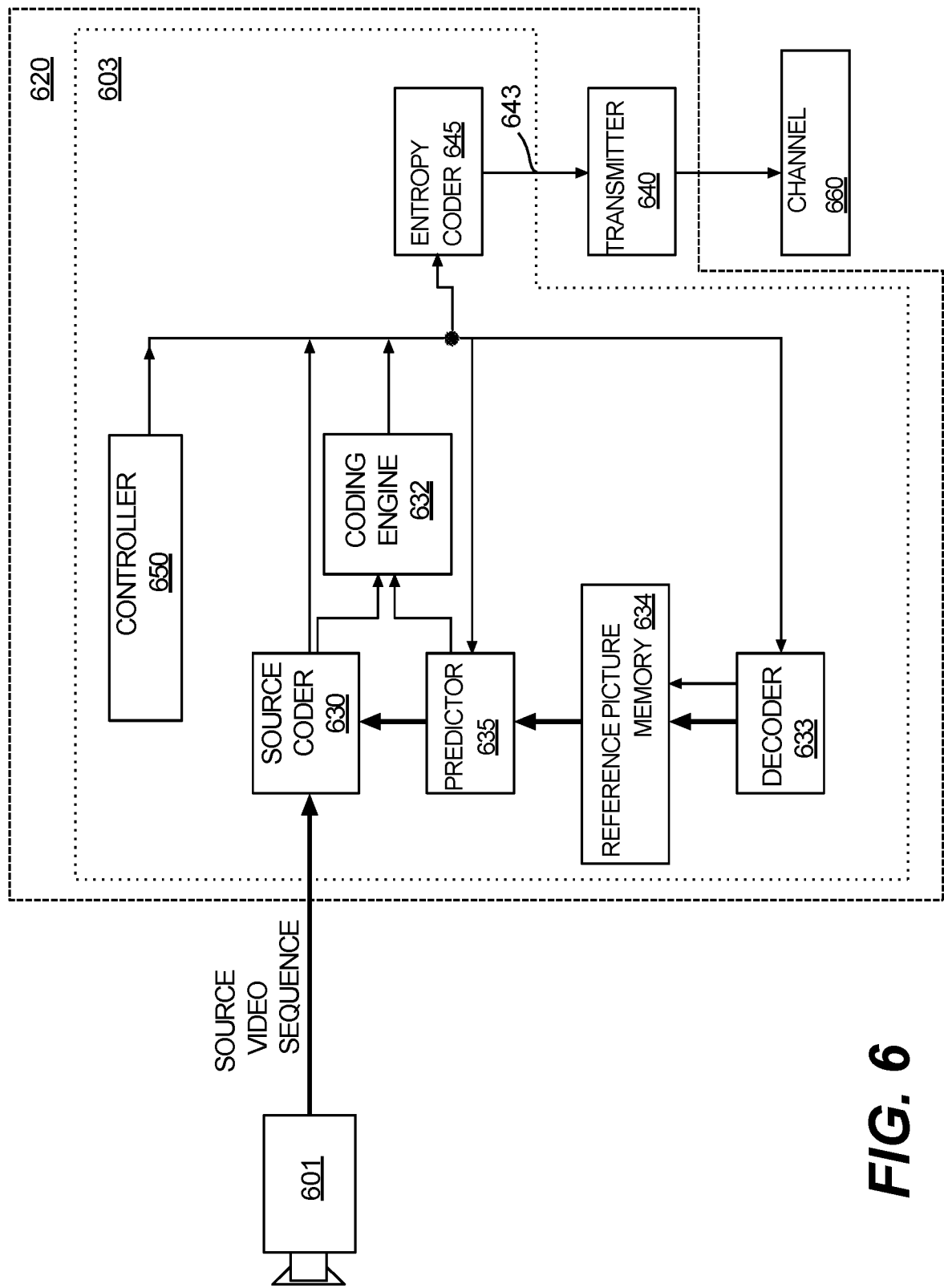


FIG. 6

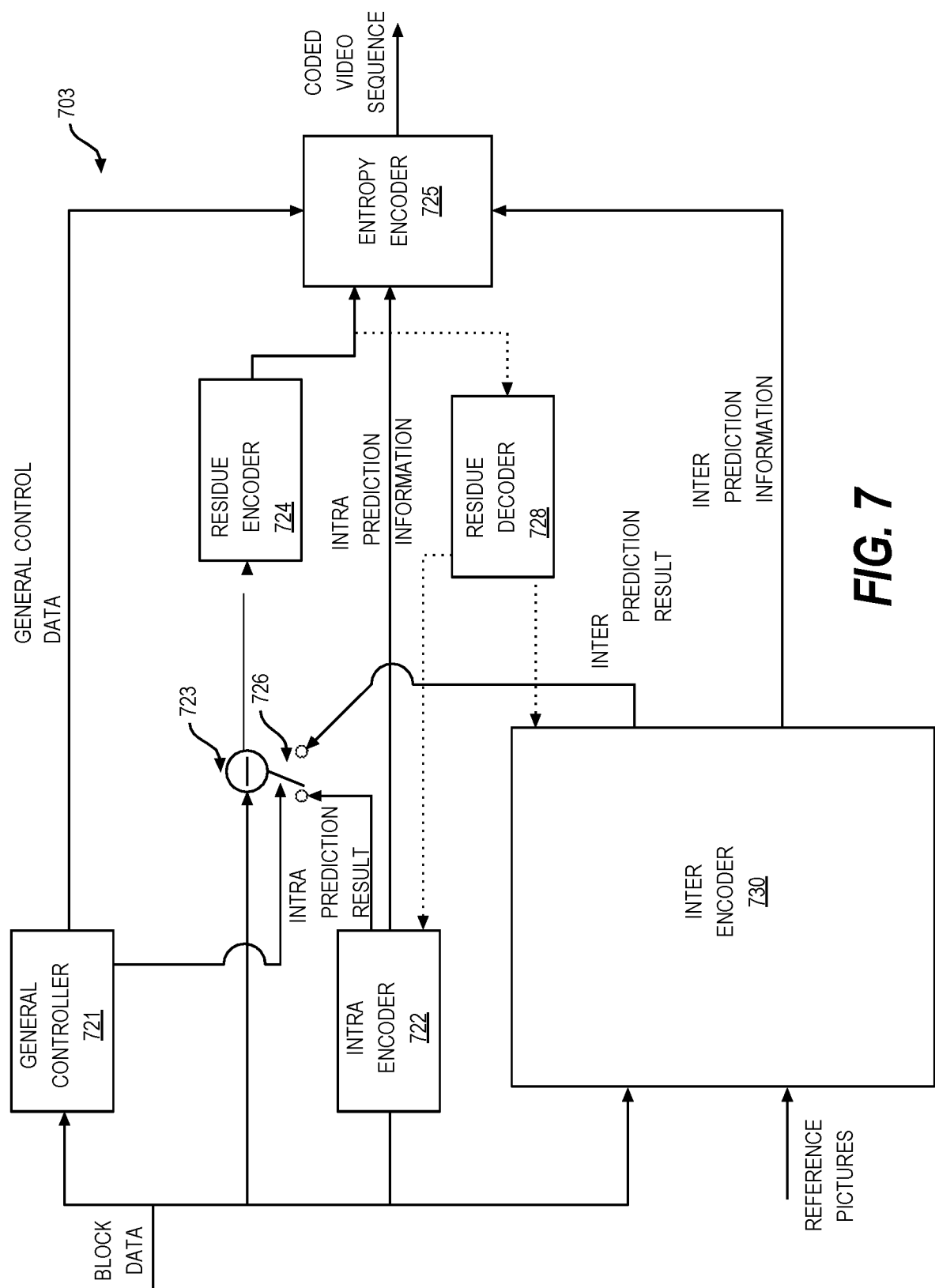


FIG. 7

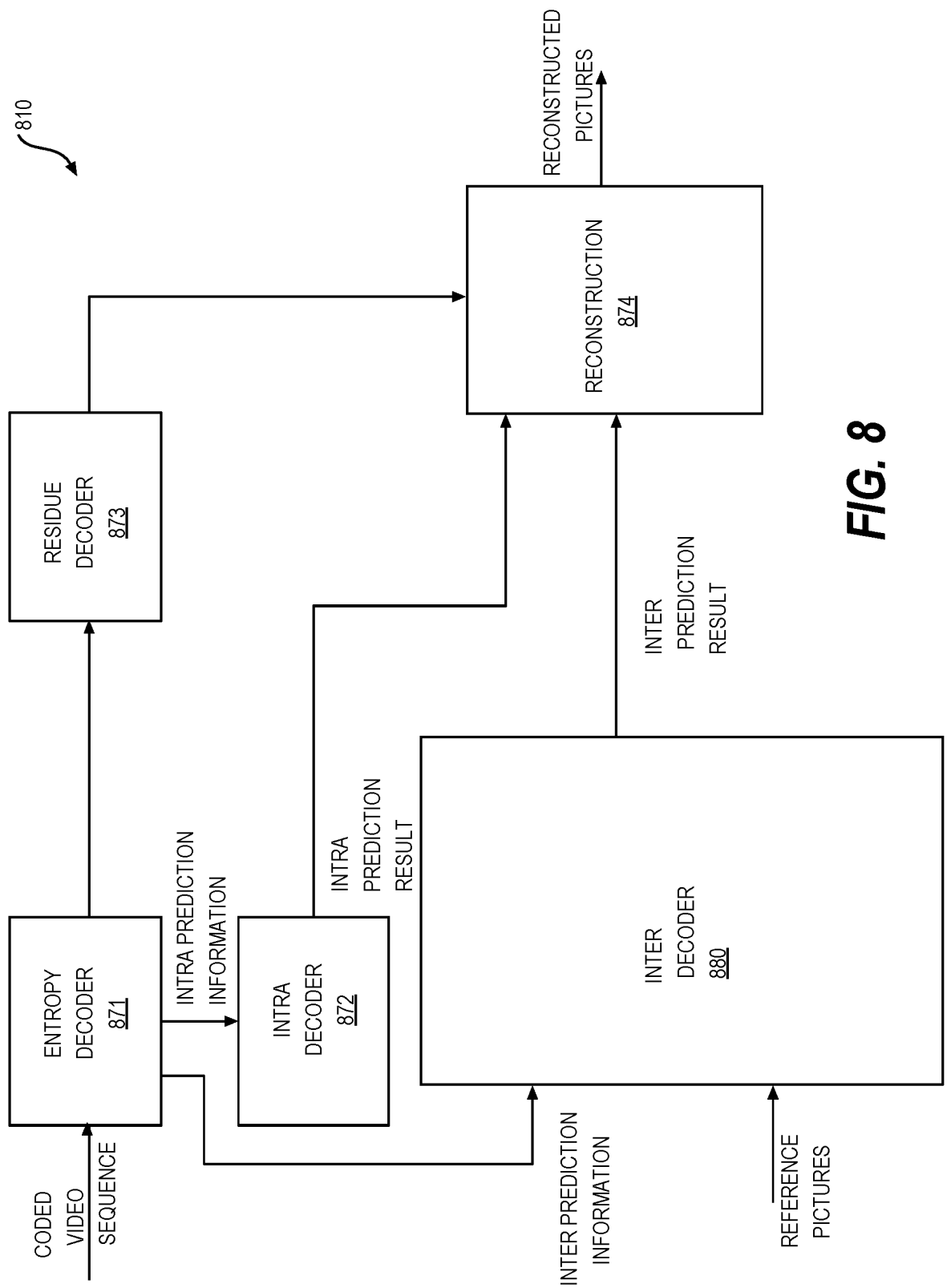


FIG. 8

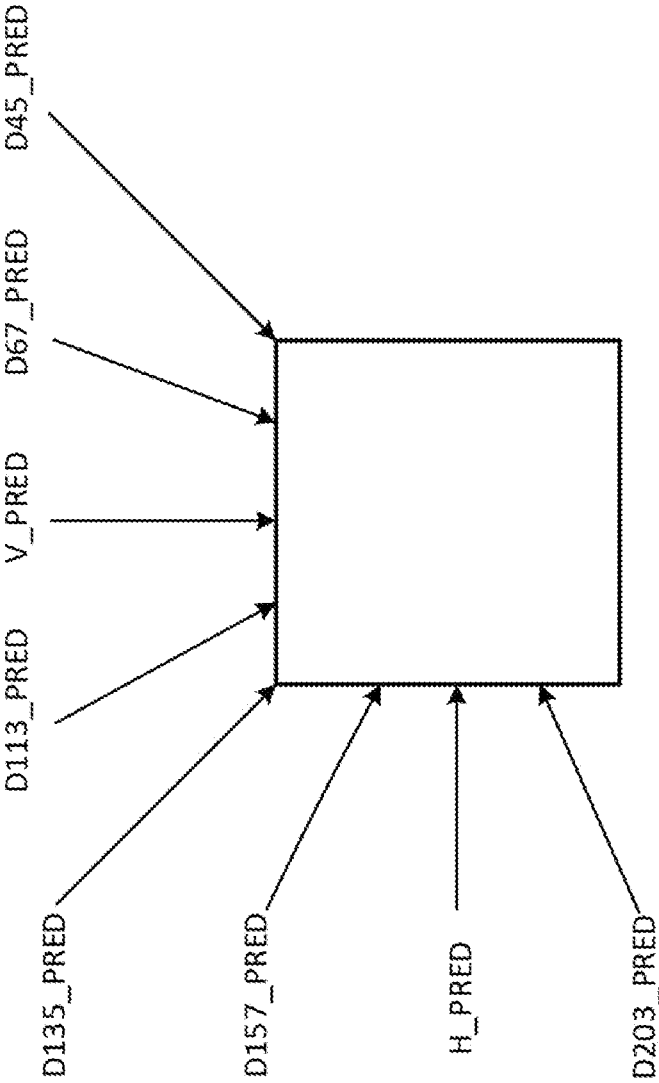


FIG. 9

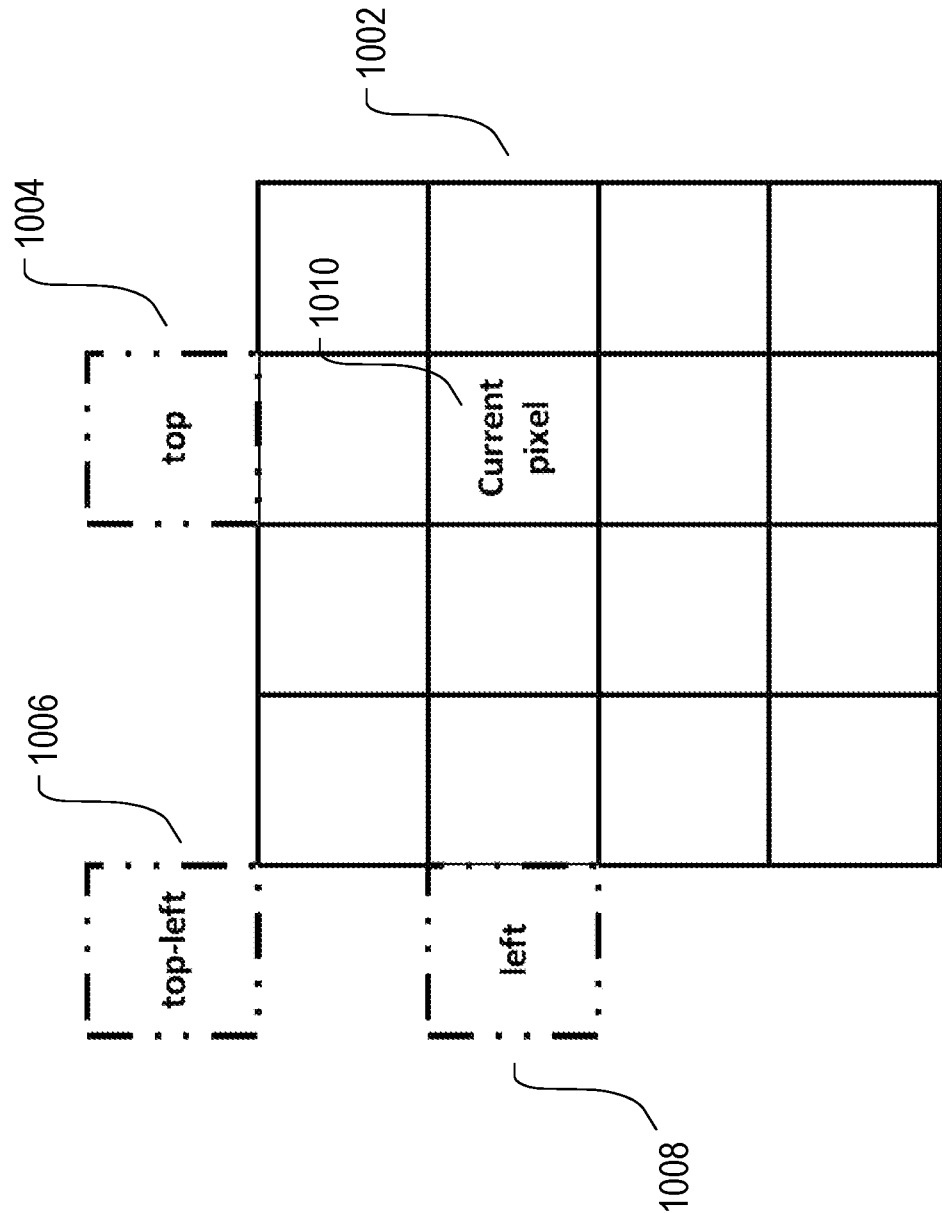


FIG. 10

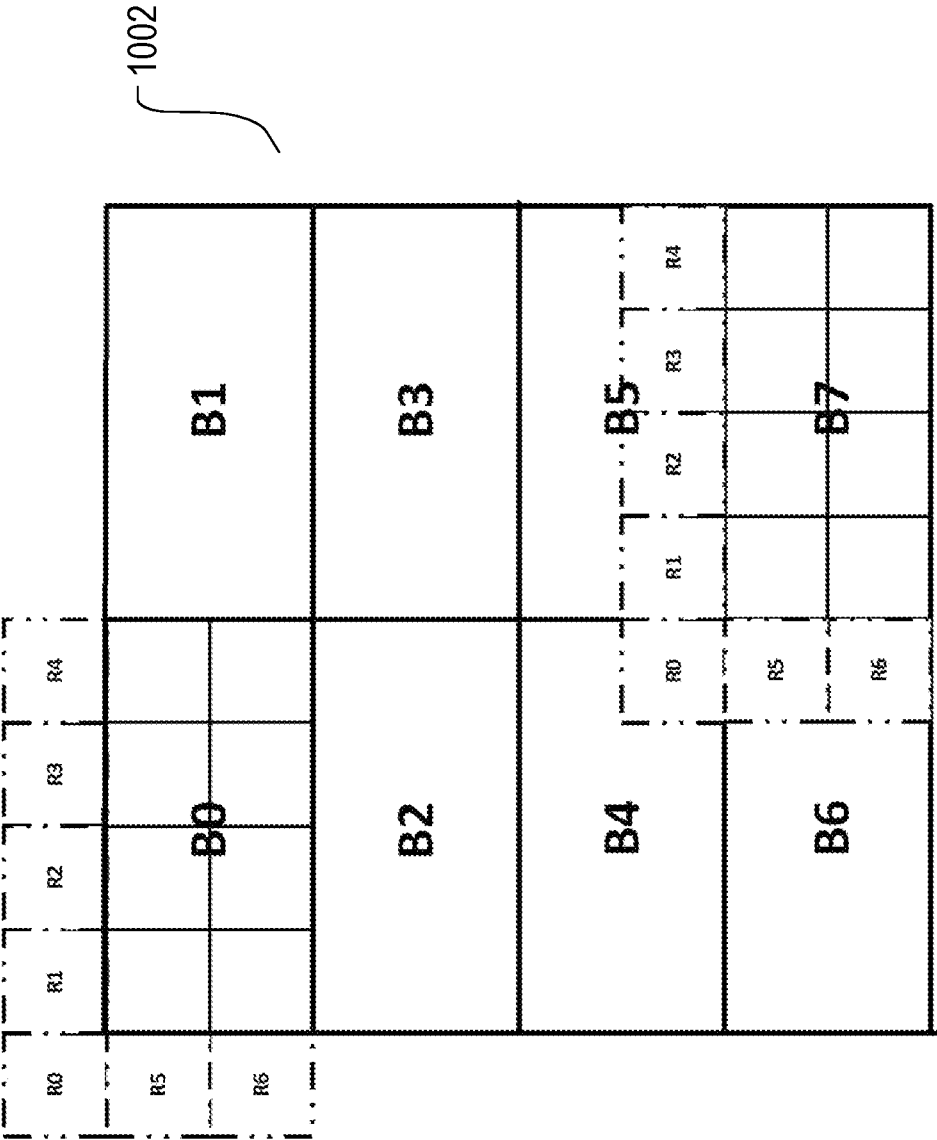


FIG. 11

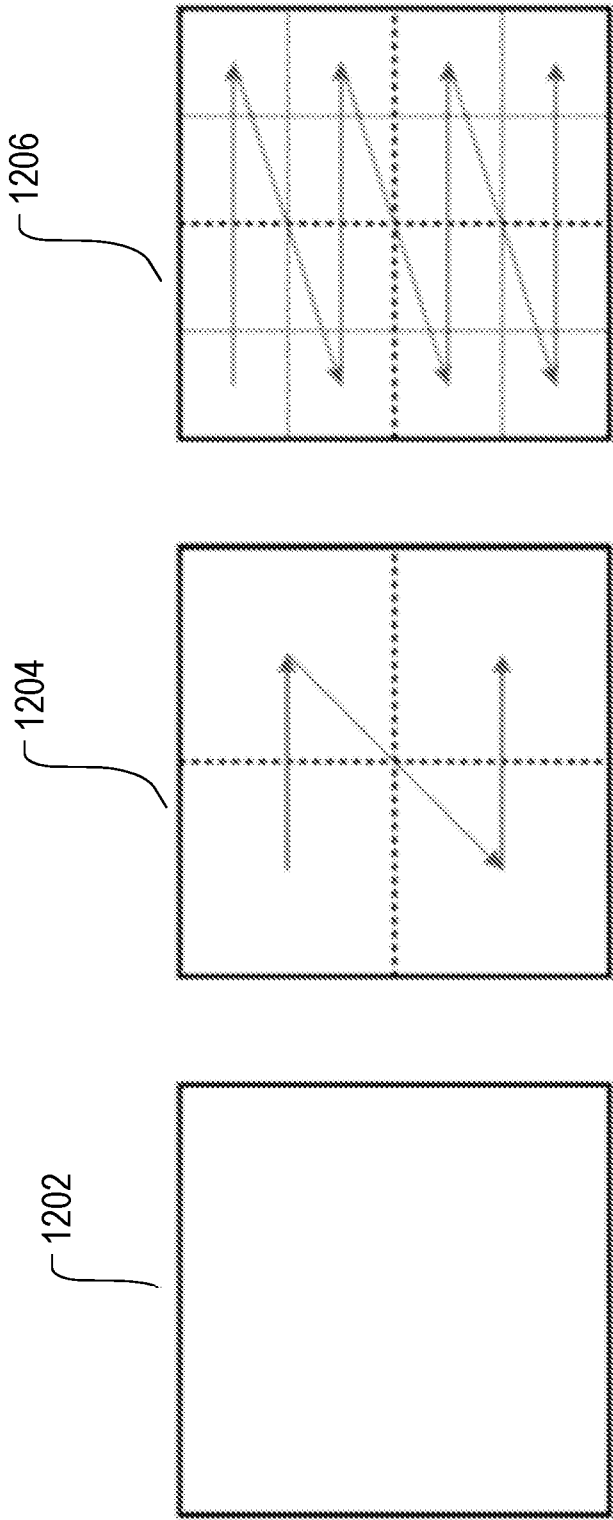


FIG. 12

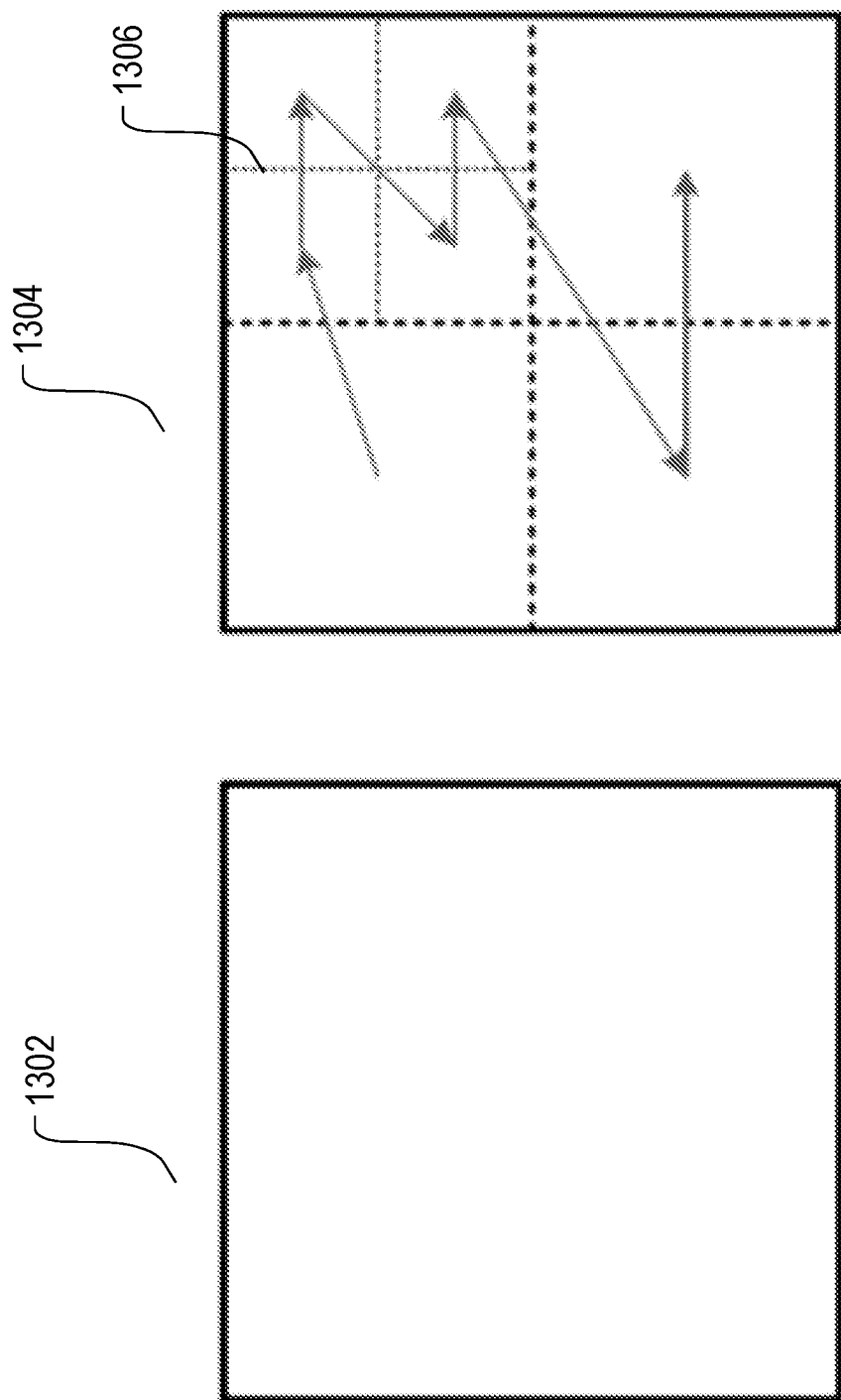


FIG. 13

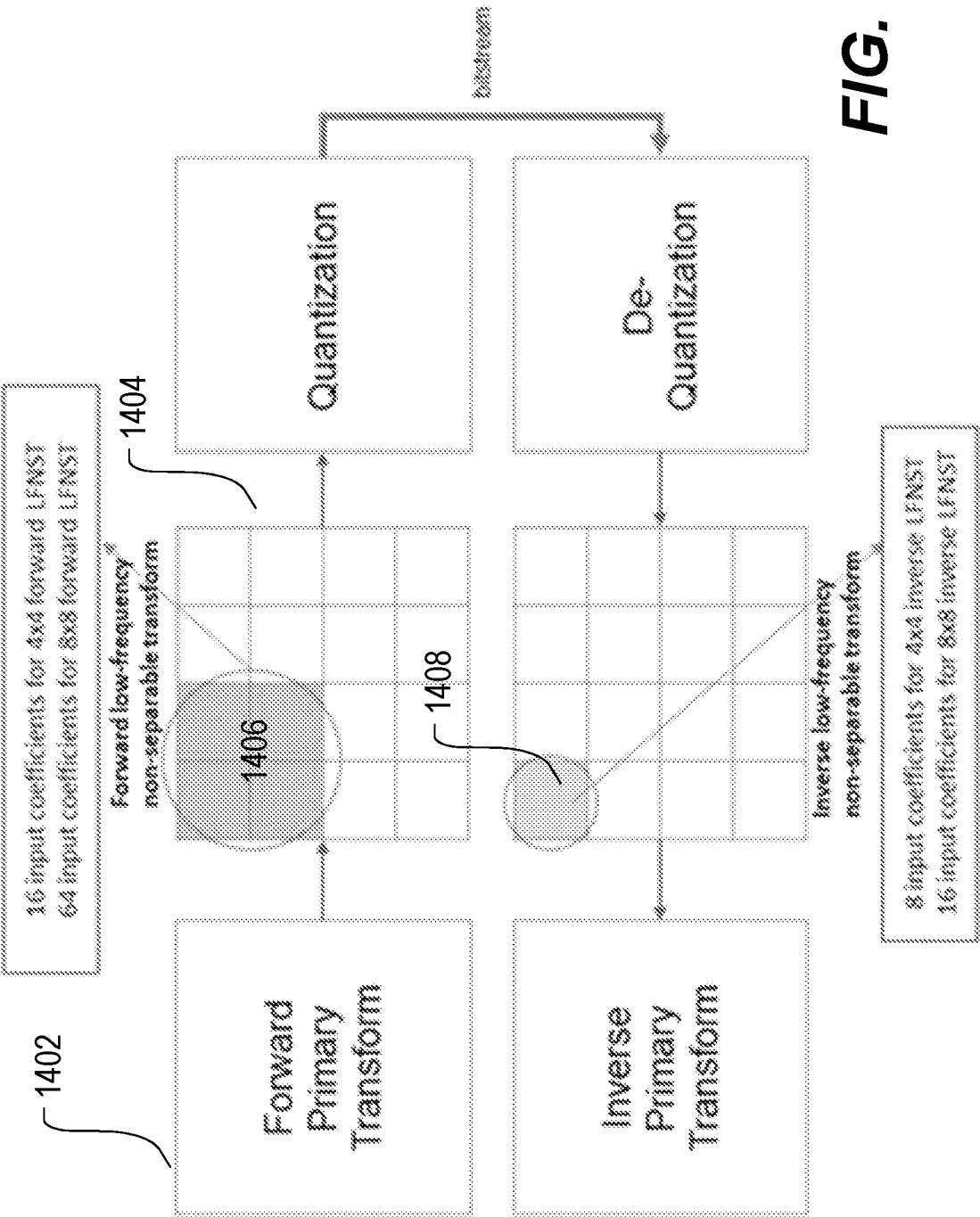


FIG. 14

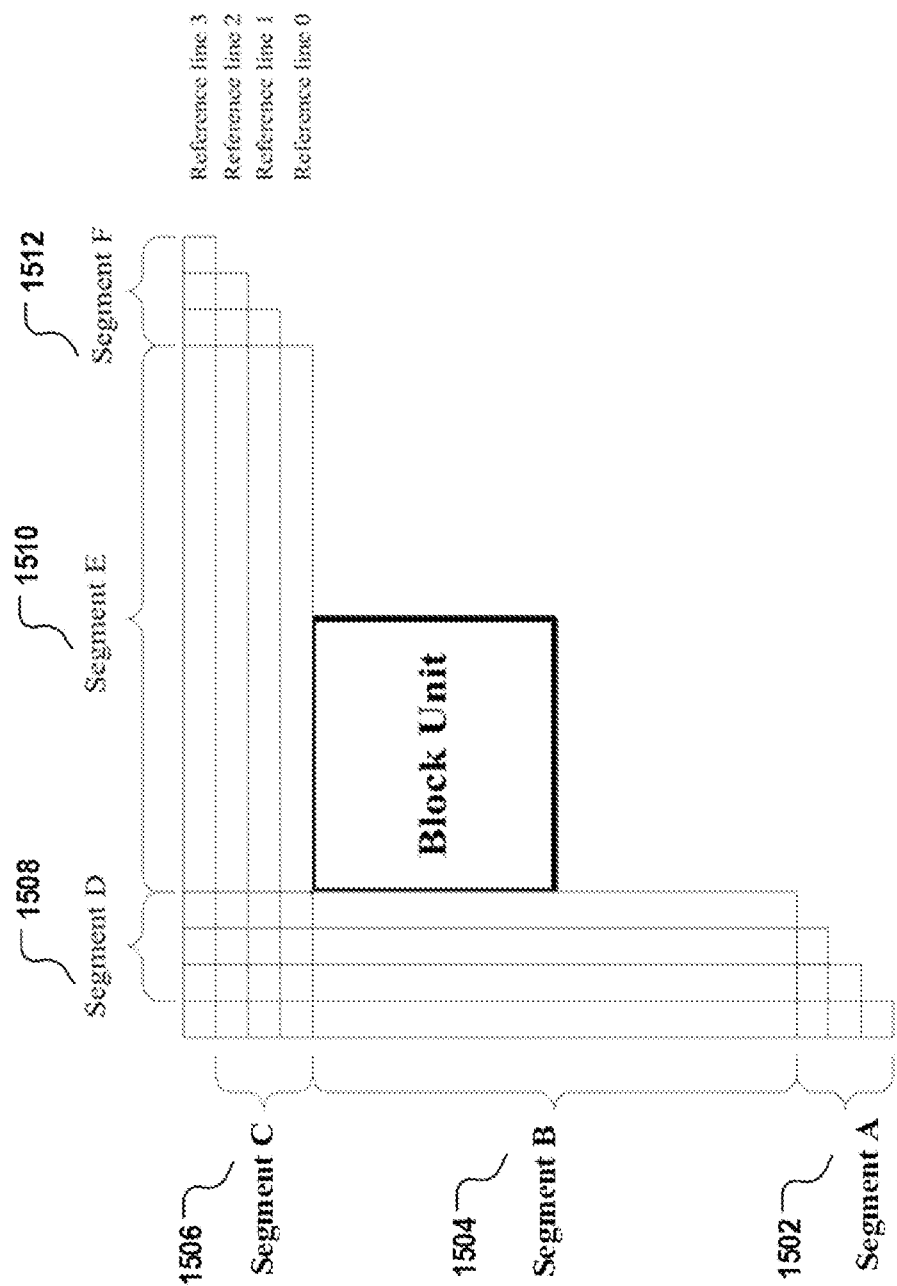


FIG. 15

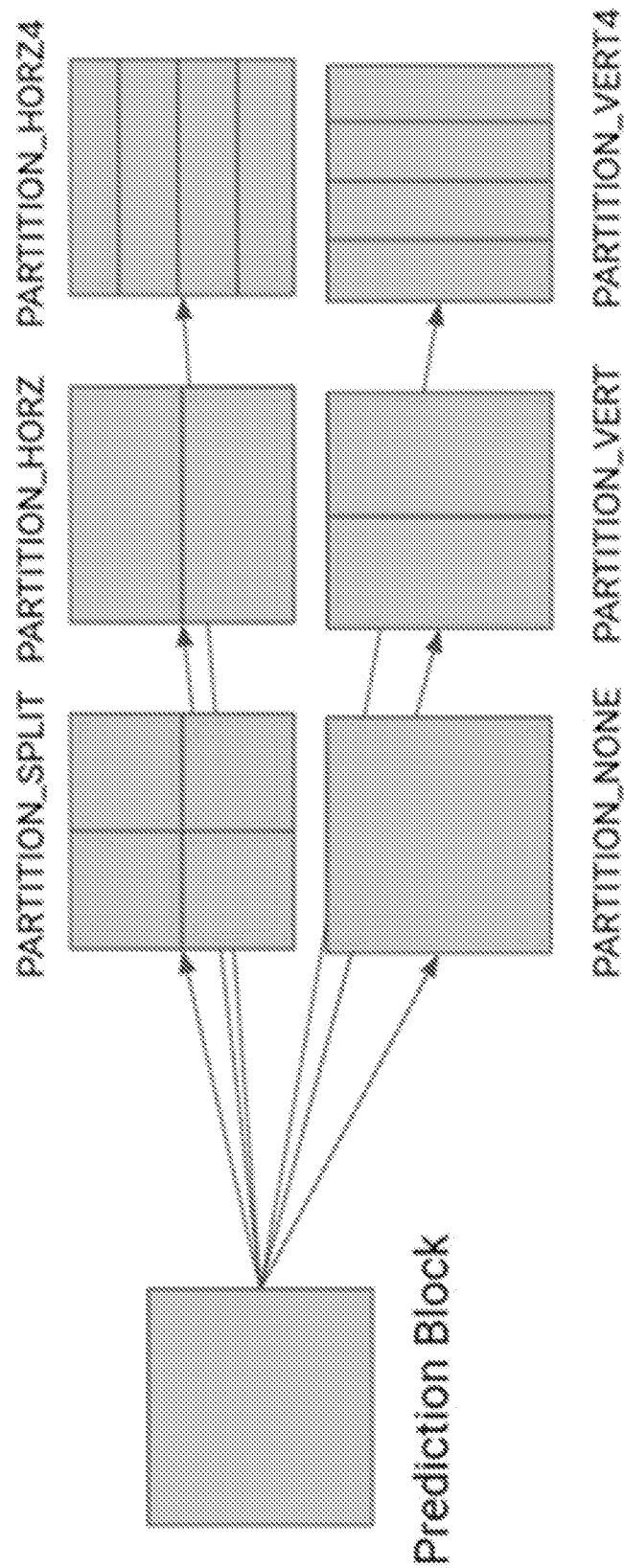
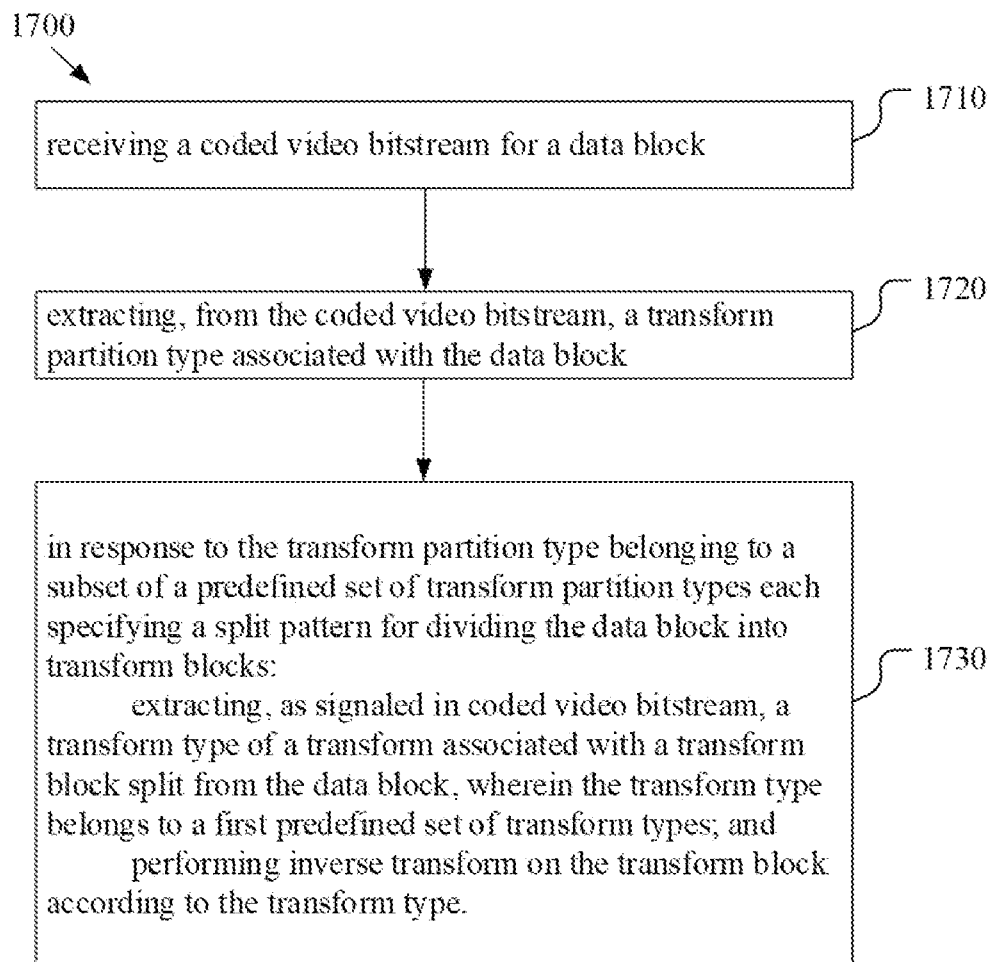


FIG. 16

**FIG. 17**

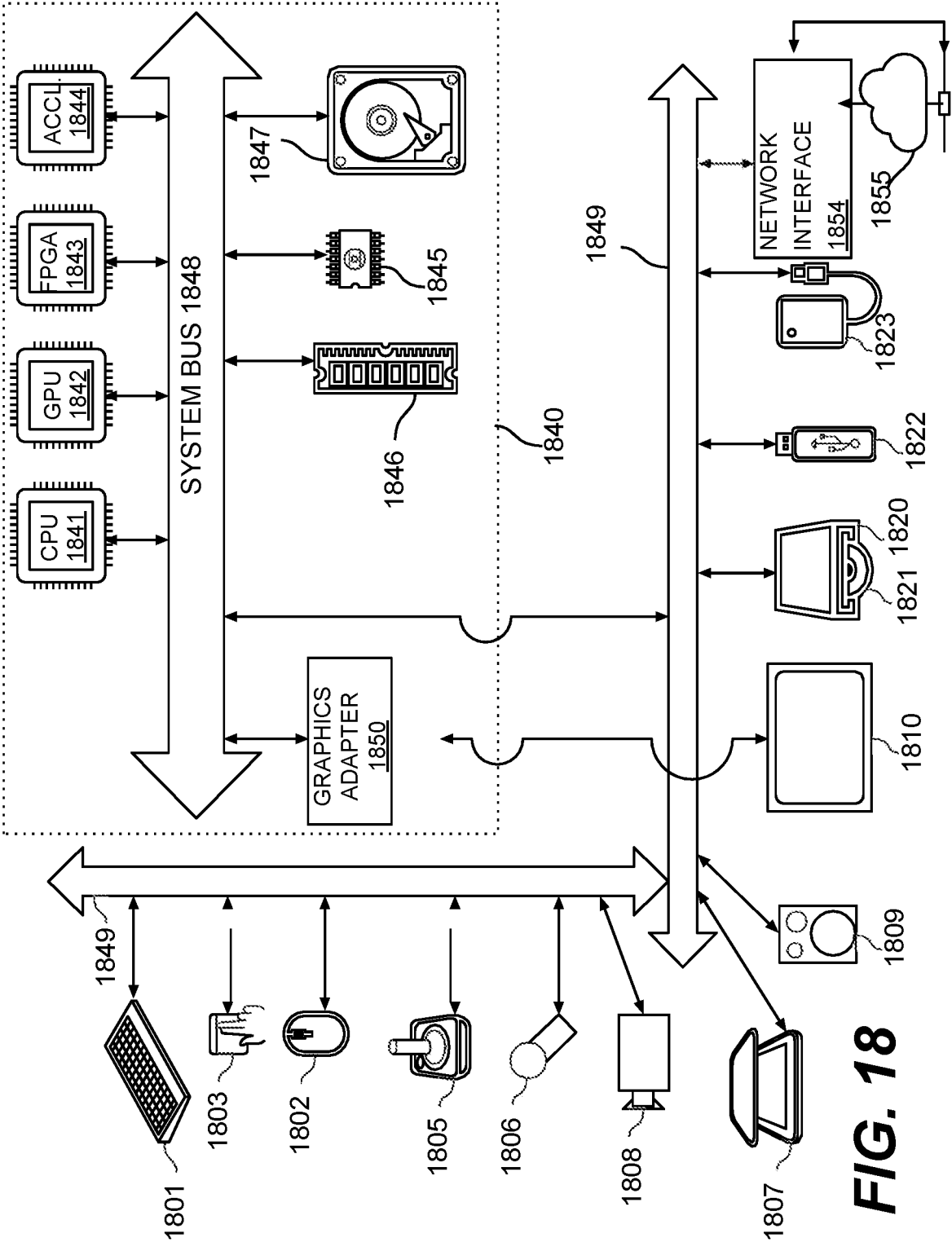


FIG. 18

INTERNATIONAL SEARCH REPORT

International application No.

PCT/US 22/14248

A. CLASSIFICATION OF SUBJECT MATTER

IPC - H04N 19/18 (2022.01)

CPC - H04N 19/18; H04N 19/48; H04N 19/70; H04N 19/186

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)

See Search History document

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

See Search History document

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

See Search History document

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2020/0092556 A1 (GOOGLE LLC), 19 March 2020 (19.03.2020), entire document, especially Abstract; para [0082], [0084]-[0085], [0092]-[0093], [0132], [0230], [0239]	1, 4, 15-17, 19/(1, 4, 15-17), 21/(1, 4, 15-17)
Y		2-3, 5-14, 18, 19/(2-3, 5-6, 12-14, 18), 20, 21/(2-3, 5-6, 12-14, 18)
Y	US 2020/0112734 A1 (TENCENT AMERICA LLC), 09 April 2020 (09.04.2020), entire document, especially Abstract; para [0010], [0075], [0228]-[0230], [0322]	2, 5-6, (7-8)/(2, 5-6), 9-14, 19/(2, 5-6, 12-14), 20, 21/(2, 5-6, 12-14), 22
Y	FRACASTORIA, GIULIA et al. Graph Transform Optimization with Application to Image Compression. 30 July 2019 (30.07.2019) [online], [retrieved on 2022-04-05]. Retrieved from the Internet < https://arxiv.org/pdf/1712.06393.pdf >, entire document, especially Abstract; pg 2	3, 7-11, 19/3, 20, 21/3, 22
Y	US 2020/0162736 A1 (ELECTRONICS AND TELECOMMUNICATIONS RESEARCH INSTITUTE), 21 May 2020 (21.05.2020), entire document, especially Abstract; para [0159], [0220]	18, 19/18, 21/18

☐ 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

05 April 2022

Date of mailing of the international search report

MAY 05 2022

Name and mailing address of the ISA/US

Mail Stop PCT, Attn: ISA/US, Commissioner for Patents
P.O. Box 1450, Alexandria, Virginia 22313-1450

Facsimile No. 571-273-8300

Authorized officer

Kari Rodriguez

Telephone No. PCT Helpdesk: 571-272-4300



(12) 发明专利申请

(10) 申请公布号 CN 115552902 A

(43) 申请公布日 2022. 12. 30

(21) 申请号 202280003871.1

刘杉

(22) 申请日 2022.01.28

(74) 专利代理机构 北京派特恩知识产权代理有限公司 11270

(30) 优先权数据

63/175,897 2021.04.16 US

17/568,275 2022.01.04 US

专利代理师 王花丽 张颖玲

(85) PCT国际申请进入国家阶段日
2022.10.31

(51) Int.Cl.
H04N 19/18 (2006.01)

(86) PCT国际申请的申请数据
PCT/US2022/014248 2022.01.28

(87) PCT国际申请的公布数据
W02022/220904 EN 2022.10.20

(71) 申请人 腾讯美国有限责任公司
地址 美国加利福尼亚州帕洛阿尔托公园大道2747号

(72) 发明人 马杜·柏林加色·克里什南 赵欣

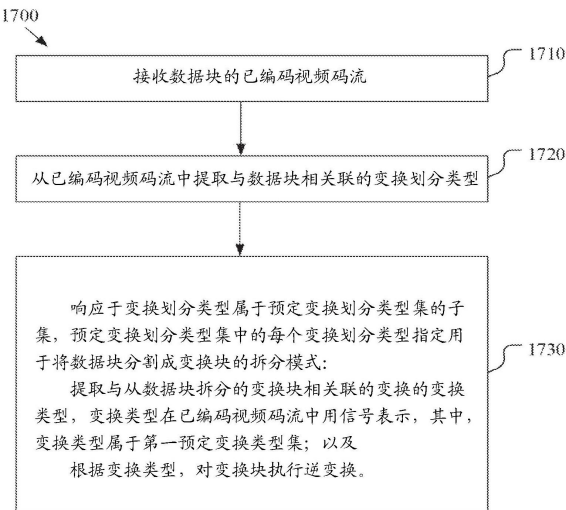
权利要求书2页 说明书29页 附图19页

(54) 发明名称

变换划分与主/二次变换类型选择之间的交互

(57) 摘要

用于在解码器中对视频数据进行编码/解码的方法、装置和计算机可读存储介质。该方法包括：接收数据块的已编码视频码流；从已编码视频码流中提取与数据块相关联的变换划分类型；以及响应于变换划分类型属于预定变换划分类型集的子集，预定变换划分类型集中的每个变换划分类型指定用于将数据块分割成变换块的拆分模式；提取与从数据块拆分的变换块相关联的变换的变换类型，如在已编码视频码流中用信号表示的，其中，变换类型属于第一预定变换类型集；以及根据变换类型，对变换块执行逆变换。



1. 一种用于在解码器中对视频数据进行解码的方法,所述方法包括:
接收数据块的已编码视频码流;
从所述已编码视频码流中提取与所述数据块相关联的变换划分类型;以及
响应于所述变换划分类型属于预定变换划分类型集的子集,所述预定变换划分类型集中的每个变换划分类型指定用于将所述数据块分割成变换块的拆分模式;
提取与从所述数据块拆分的变换块相关联的变换的变换类型,所述变换类型在已编码视频码流中用信号表示,其中,所述变换类型属于第一预定变换类型集;以及
根据所述变换类型,对所述变换块执行逆变换。
2. 根据权利要求1所述的方法,进一步包括:响应于所述变换划分类型不属于所述预定变换划分类型集的子集:
将与所述变换块相关联的变换类型确定为预定默认变换类型。
3. 根据权利要求1所述的方法,其中,所述变换是主变换,并且所述第一预定变换类型集包括:
离散余弦变换 (DCT) 类型1至DCT类型8;
非对称离散正弦变换 (ADST);
离散正弦变换 (DST) 类型1至DST类型8;
线图变换 (LGT);以及
卡洛南-洛伊变换 (KLT)。
4. 根据权利要求1所述的方法,其中,所述预定变换划分类型集的子集包括不进行变换块划分的PARTITION_NONE。
5. 根据权利要求1所述的方法,其中,与所述预定变换划分类型集的子集中的每个变换划分类型相关联的变换划分的数量小于或等于预定阈值。
6. 根据权利要求5所述的方法,其中,所述预定阈值包括从1到16的整数且包括1和16。
7. 根据权利要求1至6中任一项所述的方法,其中,所述变换是二次变换,所述第一预定变换类型集包括KLT。
8. 根据权利要求7所述的方法,其中,所述预定变换划分类型集的子集包括不进行变换块划分的PARTITION_NONE。
9. 根据权利要求7所述的方法,其中,与所述预定变换划分类型集的子集中的每个变换划分类型相关联的变换划分的数量小于或等于预定阈值。
10. 根据权利要求7所述的方法,其中,所述变换类型进一步指示:与所述变换块相关联的主变换的类型属于第二预定变换类型集。
11. 根据权利要求10所述的方法,其中,所述预定变换划分类型集的子集包括不进行变换块划分的PARTITION_NONE,并且其中,所述第二预定变换类型集包括DCT和ADST。
12. 一种用于对视频数据进行解码的方法,所述方法包括:
接收数据块的已编码视频码流;
从所述已编码视频码流中提取与所述视频数据的数据块相关联的变换划分类型;
响应于所述变换划分类型属于预定变换划分类型集的子集,从所述已编码视频码流中提取与所述数据块的变换块相关联的变换类型;以及
响应于所述变换划分类型不属于所述预定变换划分类型集,按照默认方式识别所述数

据块的变换类型。

13. 根据权利要求12所述的方法, 其中, 所述预定变换划分类型集的子集包括不进行变换块划分的PARTITION_NONE。

14. 根据权利要求12所述的方法, 其中, 所述变换类型包括主变换类型或二次变换类型。

15. 一种用于对视频数据进行解码的方法, 所述方法包括:

接收数据块的已编码视频码流;

从所述已编码视频码流中提取与所述数据块的变换块相关联的变换的变换类型; 以及
响应于所述变换类型属于预定变换类型集:

从所述已编码视频码流中提取与所述数据块相关联的变换划分类型。

16. 根据权利要求15所述的方法, 进一步包括:

响应于所述变换类型不属于所述预定变换类型集:

将与所述数据块相关联的变换划分类型确定为预定默认变换划分类型, 所述预定默认变换划分类型包括PARTITION_NONE。

17. 根据权利要求15所述的方法, 其中, 所述变换是主变换, 所述预定变换类型集包括:

DCT类型2;

ADST;

DCT类型1至DCT类型8;

DST类型1至DST类型8;

LGT; 或者

KLT。

18. 根据权利要求15所述的方法, 其中, 所述变换是二次变换, 所述预定变换类型集包括具有与预定KLT索引相关联的内核的KLT。

19. 一种设备, 包括电路, 所述电路配置成实现根据权利要求1至6中任一项所述的方法或者根据权利要求12至18中任一项所述的方法。

20. 一种设备, 包括电路, 所述电路配置成实现根据权利要求7所述的方法。

21. 一种计算机程序产品, 包括非暂时性计算机可读程序介质, 在所述非暂时性计算机可读程序介质上存储有计算机代码, 当所述计算机代码由一个或多个处理器执行时, 所述计算机代码使得所述一个或多个处理器实现根据权利要求1至6中任一项所述的方法或者根据权利要求12至18中任一项所述的方法。

22. 一种计算机程序产品, 包括非暂时性计算机可读程序介质, 在所述非暂时性计算机可读程序介质上存储有计算机代码, 当所述计算机代码由一个或多个处理器执行时, 所述计算机代码使得所述一个或多个处理器实现根据权利要求7所述的方法。

变换划分与主/二次变换类型选择之间的交互

援引并入

[0001] 本申请基于并要求于2021年4月16日提交的第63/175,897号美国临时申请和于2022年1月4日提交的第17/568,275号美国非临时申请的优先权权益,这两个申请通过引用整体并入本文中。

技术领域

[0002] 本公开描述了一组高级视频编码技术。更具体地,所公开的技术涉及视频编码和解码中的变换划分方案与主/二次变换类型选择之间的交互。

背景技术

[0003] 本文所提供的背景描述是出于总体上呈现本公开内容的目的。在该背景技术部分以及本说明书的各个方面中所描述的、目前已署名的发明人的工作所进行的程度,并不表明其在本申请提交时有资格作为现有技术,且从未明示或暗示其被承认为本公开内容的现有技术。

[0004] 可使用具有运动补偿的帧间图片预测来执行视频编码和解码。未压缩的数字视频可包括一系列图片,每个图片具有例如为 1920×1080 的亮度样本及相关联的完全或子采样的色度样本的空间大小。该一系列图片可具有例如每秒60幅图片或每秒60帧的固定或可变的图片速率(替代地,称为帧率)。对于流式传输或数据处理,未压缩的视频具有特定比特率要求。例如,具有像素分辨率为 1920×1080 、帧率为60帧/秒、色度子采样为4:2:0、每个像素8位/颜色通道的视频需要接近1.5Gbit/s的带宽。一小时的此类视频需要600GB以上的存储空间。

[0005] 视频编码和解码的一个目的可以是通过压缩来减少未压缩的输入视频信号中的冗余。压缩可有助于减小上述带宽和/或存储空间需求,在某些情况下可减小两个数量级或大于两个数量级。可采用无损压缩和有损压缩以及它们的组合。无损压缩指的是可通过解码过程从已压缩的原始信号中重建原始信号的精确副本的技术。有损压缩指的是以下编码/解码过程:原始视频信息在编码期间没有完全保留,且在解码期间不能完全恢复。当使用有损压缩时,已重建的信号可能与原始信号不同,但是原始信号和已重建的信号之间的失真足够小,以使已重建的信号可用于预期的应用,尽管会损失一些信息。在视频的情况下,在许多应用中广泛采用有损压缩。可容忍的失真量取决于应用。例如,某些消费视频流式应用的用户相比电影或电视广播应用的用户来说可以容忍更高的失真。可选择或调节可通过特定编码算法实现的压缩率,以反映各种失真容忍度:更高的可容忍的失真通常允许编码算法产生更高的损失和更高的压缩率。

[0006] 视频编码器和解码器可利用来自多个宽泛类别的技术和步骤,这些步骤包括例如运动补偿、傅里叶变换、量化和熵编码。

[0007] 视频编解码器技术可包括称为帧内编码的技术。在帧内编码中,在不参考来自先前重建的参考图片的样本或其它数据的情况下表示样本值。在一些视频编解码器中,图片

在空间上细分成样本块。当所有样本块都以帧内模式编码时,该图片可称为帧内图片。帧内图片及其派生物(例如,独立解码器刷新图片)可用于重置解码器状态,因此可用作已编码视频码流和视频会话中的第一张图片,或者用作静止图像。然后,帧内预测之后的块的样本可进行变换,而变换到频域中,且可以在熵编码之前对如此生成的变换系数进行量化。帧内预测表示一种使预变换域中的样本值最小化的技术。在一些情况下,变换之后的DC值越小,且AC系数越小,则在给定的量化步长下表示熵编码之后的块所需的位越少。

[0008] 例如从诸如MPEG-2代编码技术已知的传统帧内编码不使用帧内预测。然而,一些更新的视频压缩技术包括试图基于例如周围样本数据和/或元数据对块进行编码/解码的技术,该周围样本数据和/或元数据在空间相邻的编码和/或解码期间获得并且按照解码顺序排在正在被帧内编码或解码的数据块之前。在下文中这样的技术称为“帧内预测”技术。应注意,至少在一些情况下,帧内预测仅使用来自正在重建的当前图片的参考数据,而不使用来自参考图片的参考数据。

[0009] 帧内预测可以有許多不同形式。当在给定的视频编码技术中可使用不止一种这样的技术时,使用中的技术可称为帧内预测模式。可以在特定编解码器中提供一种或多种帧内预测模式。在一些情况下,模式可具有子模式和/或可与各种参数相关联,且模式/子模式信息和视频块的帧内编码参数可单独地编码或共同地包含在模式码字中。给定的模式、子模式和/或参数组合使用哪个码字,可能会影响通过帧内预测的编码效率增益,因此可能会影响用于将码字转换成码流的熵编码技术。

[0010] H.264引入了一种帧内预测模式,该帧内预测模式在H.265中得到完善,且在诸如联合探索模型(JEM)、下一代视频编码(VVC)和基准集(BMS)等更新的编码技术中进一步得到完善。通常,对于帧内预测,可使用属于已经变成可用的相邻样本值来形成预测值块(predictor block)。例如,特定相邻样本集的可用值可沿着一定方向和/或行复制到预测值块中。对使用方向的参考可以在码流中进行编码,或者其本身可被预测。

[0011] 参考图1A,在右下方描绘了在H.265的33种可能的帧内预测方向(对应于H.265中指定的35个帧内模式的33个角度模式)中指定的9个预测方向的子集。箭头汇聚的点(101)表示正在被预测的样本。箭头表示使用相邻样本预测101处的样本所沿的方向。例如,箭头(102)指示根据在右上方、与水平方向成45度角的一个或多个相邻样本预测样本(101)。类似地,箭头(103)指示根据在样本(101)的左下方、与水平方向成22.5度角的一个或多个相邻样本预测样本(101)。

[0012] 仍然参考图1A,在左上方描绘了 4×4 个样本的正方形块(104)(由粗体虚线指示)。正方形块(104)包括16个样本,每个样本用“S”、其在Y维度上的位置(例如,行索引)和其在X维度上的位置(例如,列索引)来标记。例如,样本S21是在Y维度上(从顶部开始)的第二个样本和在X维度上(从左侧开始)的第一个样本。类似地,样本S44是在Y和X维度上、块(104)中的第四个样本。由于块的大小为 4×4 个样本,因此S44位于右下角。还示出了遵循类似编号方案的示例性参考样本。参考样本用R、其相对于块(104)的Y位置(例如,行索引)和X位置(列索引)来标记。在H.264和H.265中,使用与正在重建的块相邻的预测样本。

[0013] 块104的帧内图片预测可通过根据用信号表示的预测方向从相邻样本复制参考样本值来开始。例如,假设已编码视频码流包括信令,该信令针对该块104指示箭头(102)的预测方向,即,根据在右上方、与水平方向成45度角的一个或多个预测样本来预测样本。在这

种情况下,根据同一个参考样本R05预测样本S41、S32、S23和S14。然后根据参考样本R08预测样本S44。

[0014] 在一些情况下,可例如通过插值来组合多个参考样本的值,以计算参考样本;尤其是当方向不能被45度整除时。

[0015] 随着视频编码技术继续发展,可能的方向的数量增加。在H.264 (2003年)中,例如九个不同的方向可用于帧内预测。在H.265 (2013年)中,增加到33个方向,以及在本公开时,JEM/VVC/BMS可支持多达65个方向。已进行实验研究来帮助识别最合适的帧内预测方向,且熵编码中的一些技术可用于以小的比特对那些最合适的方向进行编码,对于方向,接受一定的比特代价。此外,有时可以根据已解码的相邻块的帧内预测中所使用的相邻方向来预测方向本身。

[0016] 图1B示出了示意图(180),其描绘了根据JEM的65个帧内预测方向,以说明在随着时间的推移而开发的各種编码技术中,预测方向的数量增加。

[0017] 在已编码视频码流中,将表示帧内预测方向的比特映射到该预测方向的方式,可能因视频编码技术的不同而不同;例如,其范围可以从预测方向简单直接映射到帧内预测模式,变化为预测方向映射到码字,映射到涉及最可能模式的复杂自适应方案,以及类似技术。然而,在所有情况下,可存在某些帧内预测方向,这些方向与某些其它方向相比,在统计上出现在视频内容中的可能性较小。由于视频压缩的目标是减少冗余,因此在一种设计良好的视频编码技术中,相比于更可能的方向,那些不太可能的方向可由更多的位数表示。

[0018] 帧间图片预测或帧间预测可基于运动补偿。在运动补偿中,来自先前重建的图片或其部分(参考图片)的样本数据在沿着由运动矢量(此后称为MV)指示的方向在空间上偏移之后,可用于预测新重建的图片或图片部分(例如,块)。在一些情况下,参考图片可与当前正在重建的图片相同。MV可具有X和Y两个维度,或具有三个维度,第三个维度为正在使用的参考图片的指示(类似于时间维度)。

[0019] 在一些视频压缩技术中,可根据其它MV,例如根据在空间上与正在重建的区域相邻的样本数据的另一区域相关的、且按解码次序处于当前MV之前的其它MV来预测适用于样本数据的某个区域的该当前MV。这样做可通过消除相关MV中的冗余来大大减少对MV进行编码所需的数据总量,从而增加压缩效率。MV预测可有效地工作,例如,由于在对从相机获得的输入视频信号(称为自然视频)进行编码时,存在以下统计可能性:比适用单个MV的区域更大的区域在视频序列中沿着相似的方向移动,因此在某些情况下,可使用从相邻区域的MV导出的相似运动矢量来预测该更大的区域。这使得用于给定区域的实际MV与根据周围MV所预测的MV相似或相同。进而在熵编码之后,该MV可以采用比直接对MV进行编码(而非根据相邻MV来预测该MV)时使用的位数更小的位数来进行表示。在一些情况下,MV预测可以是无损压缩从原始信号(即:样本流)中导出的信号(即:MV)的示例。在其它情况下,例如由于根据多个周围MV计算预测值时出现舍入误差,使得MV预测本身可以是有损的。

[0020] H.265/HEVC (ITU-T H.265建议书,“High Efficiency Video Coding (高效视频编码)”,2016年12月)中描述了各种MV预测机制。在H.265指定的多种MV预测机制中,下面描述的是下文称为“空间合并(spatial merge)”的技术。

[0021] 具体地,参考图2,当前块(201)包括在运动搜索过程期间已由编码器找到的样本,可根据已产生空间偏移的相同大小的先前块来预测该样本。可以从与一个或多个参考图片

相关联的元数据中导出MV,而非直接对该MV进行编码,例如使用与被标记为A0、A1和B0、B1、B2(分别对应202到206)的五个周围样本中的任一样本相关联的MV,(按解码次序)从最近的参考图片中导出该MV。在H.265中,MV预测可使用来自相邻块正在使用的同一参考图片的预测值。

发明内容

[0022] 本公开描述了用于视频编码和/或解码的方法、装置和计算机可读存储介质的各种实施例。

[0023] 根据一方面,本公开的一个实施例提供一种用于在解码器中对视频数据进行编码/解码的方法。该方法包括:接收数据块的已编码视频码流;从已编码视频码流中提取与数据块相关联的变换划分类型;以及响应于变换划分类型属于预定变换划分类型集的子集,预定变换划分类型集中的每个变换划分类型指定用于将数据块分割成变换块的拆分模式:提取与从数据块拆分的变换块相关联的变换的变换类型,该变换类型在已编码视频码流中用信号表示,其中,变换类型属于第一预定变换类型集;以及根据变换类型,对变换块执行逆变换。

[0024] 根据另一方面,本公开的一个实施例提供一种用于对视频数据进行编码/解码的方法。该方法包括:接收数据块的已编码视频码流;从已编码视频码流中提取与视频数据的数据块相关联的变换划分类型;响应于变换划分类型属于预定变换划分类型集的子集,从已编码视频码流中提取与数据块的变换块相关联的变换类型;以及响应于变换划分类型不属于预定变换划分类型集,按照默认方式识别数据块的变换类型。

[0025] 根据另一方面,本公开的一个实施例提供一种用于对视频数据进行编码/解码的方法。该方法包括:接收数据块的已编码视频码流;从已编码视频码流中提取与数据块的变换块相关联的变换的变换类型;以及响应于变换类型属于预定变换类型集:从已编码视频码流中提取与数据块相关联的变换划分类型。

[0026] 根据另一方面,本公开的一个实施例提供一种用于视频编码和/或解码的装置。该装置包括:存储指令的存储器;和与存储器通信的处理器。当处理器执行指令时,处理器配置成使得该装置执行上述用于视频解码和/或编码的方法。

[0027] 根据又一方面,本公开的一个实施例提供一种非暂时性计算机可读介质,非暂时性计算机可读介质存储有指令,当该指令由用于视频解码和/或编码的计算机执行时,该指令使得该计算机执行上述用于视频解码和/或编码的方法。

[0028] 上述和其它方面及其实现方式在附图、说明书和权利要求中更详细地描述。

附图说明

[0029] 根据以下详细描述和附图,所公开的主题的进一步的特征、性质和各种优点将更加明显,在附图中:

[0030] 图1A示出了帧内预测定向模式的示例性子集的示意性图示。

[0031] 图1B示出了示例性帧内预测方向的图示。

[0032] 图2示出了在一个示例中,用于运动矢量预测的当前块及其周围空间合并候选的示意性图示。

- [0033] 图3示出了根据一个示例性实施例的通信系统(300)的简化框图的示意性图示。
- [0034] 图4示出了根据一个示例性实施例的通信系统(400)的简化框图的示意性图示。
- [0035] 图5示出了根据一个示例性实施例的视频解码器的简化框图的示意性图示。
- [0036] 图6示出了根据一个示例性实施例的视频编码器的简化框图的示意性图示。
- [0037] 图7示出了根据另一示例性实施例的视频编码器的框图。
- [0038] 图8示出了根据另一示例性实施例的视频解码器的框图。
- [0039] 图9示出了根据本公开的示例性实施例的定向(directional)帧内预测模式。
- [0040] 图10示出了根据本公开的示例性实施例的非定向帧内预测模式。
- [0041] 图11示出了根据本公开的示例性实施例的递归帧内预测模式。
- [0042] 图12示出了根据本公开的示例性实施例的帧内预测块的变换块划分和扫描。
- [0043] 图13示出了根据本公开的示例性实施例的帧间预测块的变换块划分和扫描。
- [0044] 图14示出了根据本公开的示例性实施例的低频不可分离变换过程。
- [0045] 图15示出了根据本公开的示例性实施例的基于各个参考行(reference line)的帧内预测方案。
- [0046] 图16示出了根据本公开的示例性实施例的非递归块划分方案。
- [0047] 图17示出了根据本公开的示例性实施例的流程图。
- [0048] 图18示出了根据本公开的示例性实施例的计算机系统的示意性图示。

具体实施方式

[0049] 图3示出了根据本公开的一个实施例的通信系统(300)的简化框图。通信系统(300)包括多个终端设备,该多个终端设备可通过例如网络(350)彼此通信。例如,通信系统(300)包括通过网络(350)互连的第一终端设备对(310)和(320)。在图3的示例中,第一终端设备对(310)和(320)可执行单向数据传输。例如,终端设备(310)可以对视频数据(例如,由终端设备(310)采集的视频图片流的视频数据)进行编码,以通过网络(350)传输到另一终端设备(320)。已编码视频数据可以以一个或多个已编码视频码流的形式传输。终端设备(320)可以从网络(350)接收已编码视频数据,对已编码视频数据进行解码以恢复视频图片,以及根据所恢复的视频数据显示视频图片。单向数据传输可以在媒体服务等应用中实现。

[0050] 在另一示例中,通信系统(300)包括执行已编码视频数据的双向传输的第二终端设备对(330)和(340),该双向传输可例如在视频会议应用期间实现。对于双向数据传输,在一个示例中,终端设备(330)和(340)中的每个终端设备可以对视频数据(例如,由终端设备采集的视频图片流的视频数据)进行编码,以通过网络(350)传输到终端设备(330)和(340)中的另一终端设备。终端设备(330)和(340)中的每个终端设备还可接收由终端设备(330)和(340)中的另一终端设备传输的已编码视频数据,且可以对已编码视频数据进行解码以恢复视频图片,以及可根据所恢复的视频数据在可访问的显示设备上显示视频图片。

[0051] 在图3的示例中,终端设备(310)、终端设备(320)、终端设备(330)和终端设备(340)可实现为服务器、个人计算机和智能电话,但是本公开的潜在原理的适用性可不限于此。本公开的实施例可以在台式计算机、膝上型计算机、平板电脑、媒体播放器、可穿戴计算机、专用视频会议设备等中实现。网络(350)表示在终端设备(310)、终端设备(320)、终端设

备(330)和终端设备(340)之间传送已编码视频数据的任何数量的网络,包括例如有线(连线的)和/或无线通信网络。通信网络(350)9可以在电路交换信道、分组交换信道和/或其它类型的信道中交换数据。代表性的网络包括电信网络、局域网、广域网和/或互联网。出于本讨论的目的,除非在本文中予以明确说明,否则网络(350)的架构和拓扑对于本公开的操作来说可能是无关紧要的。

[0052] 作为用于所公开的主题的应用的示例,图4示出了视频编码器和视频解码器在视频流式传输环境中的放置方式。所公开的主题可同等地适用于其它视频应用,包括例如视频会议、数字TV、广播、游戏、虚拟现实、在包括CD、DVD、存储棒等的数字介质上存储压缩视频等等。

[0053] 视频流式传输系统可包括视频采集子系统(413),采集子系统(413)可包括诸如数码相机(401),视频源(401)用于创建未压缩的视频图片或图像流(402)。在一个示例中,视频图片流(402)包括由视频源401的数码相机记录的样本。相较于已编码视频数据(404)(或已编码视频码流),被描绘为粗线以强调高数据量的视频图片流(402)可由电子设备(420)处理,电子设备(420)包括耦接到视频源(401)的视频编码器(403)。视频编码器(403)可包括硬件、软件或软硬件组合,以实现或实施如下文更详细地描述的所公开的主题的各方面。相较于未压缩的视频图片流(402),被描绘为细线以强调较低数据量的已编码视频数据(404)(或已编码视频码流(404))可存储在流式传输服务器(405)上以供将来使用,或者直接存储于下游视频设备(未示出)。一个或多个流式传输客户端子系统,例如图4中的客户端子系统(406)和客户端子系统(408)可访问流式传输服务器(405)以检索已编码视频数据(404)的副本(407)和副本(409)。客户端子系统(406)可包括例如电子设备(430)中的视频解码器(410)。视频解码器(410)对已编码视频数据的传入副本(407)进行解码,且产生可以在显示器(412)(例如,显示屏)或其它呈现设备(未描绘)上呈现的未压缩的输出视频图片流(411)。视频解码器410可配置成执行本公开所描述的各种功能中的一些或所有功能。在一些流式传输系统中,可根据某些视频编码/压缩标准对已编码视频数据(404)、(407)和(409)(例如,视频码流)进行编码。这些标准的示例包括ITU-T H.265建议书。在一个示例中,正在开发的视频编码标准非正式地称为下一代视频编码(VVC)。所公开的主题可用于VVC的上下文和其它视频编码标准中。

[0054] 应注意,电子设备(420)和电子设备(430)可包括其它组件(未示出)。例如,电子设备(420)可包括视频解码器(未示出),且电子设备(430)还可包括视频编码器(未示出)。

[0055] 在下文中,图5示出了根据本公开的任意实施例的视频解码器(510)的框图。视频解码器(510)可包括在电子设备(530)中。电子设备(530)可包括接收器(531)(例如,接收电路)。视频解码器(510)可用于代替图4的示例中的视频解码器(410)。

[0056] 接收器(531)可接收将由视频解码器(510)解码的一个或多个已编码视频序列。在同一实施例或另一实施例中,一次可以对一个已编码视频序列进行解码,其中每个已编码视频序列的解码独立于其它已编码视频序列。每个视频序列可与多个视频帧或图像相关联。可以从信道(501)接收已编码视频序列,信道(501)可以是硬件/软件链路,其通向存储已编码视频数据的存储设备或者传输已编码视频数据的流式传输源。接收器(531)可接收可转发到它们各自的处理电路(未描绘)的已编码视频数据和其它数据,例如已编码音频数据和/或辅助数据流。接收器(531)可以将已编码视频序列与其它数据分开。为了防止网络

抖动,缓冲存储器(515)可设置在接收器(531)和熵解码器/解析器(520)(此后称为“解析器(520)”)之间。在某些应用中,缓冲存储器(515)可实现为视频解码器(510)的一部分。在其它应用中,缓冲存储器(515)可位于视频解码器(510)的外部且与视频解码器(510)分离(未描绘)。而在另一些其它应用中,在视频解码器(510)的外部可设置缓冲存储器(未描绘),其目的是例如防止网络抖动,且在视频解码器(510)的内部可设置另一附加的缓冲存储器(515)以例如处理播放定时。当接收器(531)从具有足够带宽和可控性的存储/转发设备或从等时同步网络接收数据时,可能不需要缓冲存储器(515),或可以将缓冲存储器做得较小。为了在诸如互联网等业务分组网络上使用,可能需要足够大小的缓冲存储器(515),缓冲存储器(515)的大小可相对较大。这种缓冲存储器可实现为具有自适应大小,且可至少部分地在操作系统或视频解码器(510)外部的类似元件(未描绘)中实现。

[0057] 视频解码器(510)可包括解析器(520),以根据已编码视频序列重建符号(521)。这些符号的类别包括用于管理视频解码器(510)的操作的信息,以及用于控制诸如显示器(512)(例如,显示屏)之类的呈现设备的潜在信息,该呈现设备可以是或不是电子设备(530)的整体部分,但是可耦接到电子设备(530),如图5所示。用于呈现设备的控制信息可以是辅助增强信息(SEI消息)或视频可用性信息(VUI)参数集片段(未描绘)的形式。解析器(520)可以对解析器(520)所接收的已编码视频序列进行解析/熵解码。已编码视频序列的熵编码可根据视频编码技术或标准进行,且可遵循各种原理,包括可变长度编码、霍夫曼(Huffman)编码、具有或不具有上下文敏感度的算术编码等。解析器(520)可基于对应于子群的至少一个参数,从已编码视频序列提取用于视频解码器中的像素的子群中的至少一个子群的子群参数集。子群可包括图片群组(GOP)、图片、图块、切片、宏块、编码单元(CU)、块、变换单元(TU)、预测单元(PU)等。解析器(520)还可以从已编码视频序列提取信息,例如变换系数(例如,傅里叶变换系数),量化器参数值,运动矢量等。

[0058] 解析器(520)可以对从缓冲存储器(515)接收的视频序列执行熵解码/解析操作,从而创建符号(521)。

[0059] 取决于已编码视频图片或一部分已编码视频图片(例如:帧间图片和帧内图片,帧间块和帧内块)的类型以及其它因素,符号(521)的重建可涉及多个不同的处理或功能单元。涉及到的单元以及单元所涉及的方式可由解析器(520)通过从已编码视频序列解析的子群控制信息来控制。为了简单起见,未描绘解析器(520)与下文的多个处理或功能单元之间的此类子群控制信息流。

[0060] 除了已提及的功能块之外,视频解码器(510)可以在概念上细分成如下文所描述的多个功能单元。在商业约束下运行的实际实现方式中,这些功能单元中的许多功能单元彼此紧密交互且可至少部分地彼此集成。然而,出于清楚地描述所公开的主题的各种功能的目的,在本公开的下文中采用概念上细分成的多个功能。

[0061] 第一单元可包括缩放器/逆变换单元(551)。缩放器/逆变换单元(551)可以从解析器(520)接收作为符号(521)的量化变换系数以及控制信息,包括指示要使用哪种类型的逆变换、块大小、量化因子/参数、量化缩放矩阵等信息。缩放器/逆变换单元(551)可输出包括样本值的块,样本值可输入到聚合器(555)中。

[0062] 在一些情况下,缩放器/逆变换(551)的输出样本可属于帧内编码块;即,不使用来自先前重建的图片的预测信息,但是可使用来自当前图片的先前重建部分的预测信息的

块。此类预测信息可由帧内图片预测单元 (552) 提供。在一些情况下, 帧内图片预测单元 (552) 可使用已重建且存储在当前图片缓冲器 (558) 中的周围块信息来生成大小和形状与正在重建的块相同的块。例如, 当前图片缓冲器 (558) 缓冲部分重建的当前图片和/或完全重建的当前图片。在一些实现方式中, 聚合器 (555) 可基于每个样本, 将帧内预测单元 (552) 生成的预测信息添加到由缩放器/逆变换单元 (551) 提供的输出样本信息中。

[0063] 在其它情况下, 缩放器/逆变换单元 (551) 的输出样本可属于帧间编码和潜在运动补偿的块。在这种情况下, 运动补偿预测单元 (553) 可访问参考图片存储器 (557) 以提取用于帧间图片预测的样本。在根据属于块的符号 (521) 对所提取的样本进行运动补偿之后, 这些样本可由聚合器 (555) 添加到缩放器/逆变换单元 (551) 的输出 (单元 551 的输出可称为残差样本或残差信号), 从而生成输出样本信息。运动补偿预测单元 (553) 从参考图片存储器 (557) 内的地址提取预测样本可受到运动矢量控制, 该运动矢量可以以符号 (521) 的形式提供给运动补偿预测单元 (553) 使用, 符号 (521) 可具有例如 X 分量、Y 分量 (偏移) 和参考图片分量 (时间)。运动补偿还可包括在使用子样本精确运动矢量时, 从参考图片存储器 (557) 提取的样本值的内插, 运动补偿还可与运动矢量预测机制等相关联。

[0064] 聚合器 (555) 的输出样本可经受环路滤波器单元 (556) 中的各种环路滤波技术。视频压缩技术可包括环路内滤波器技术, 该环路内滤波器技术受控于包括在已编码视频序列 (还称为已编码视频码流) 中且可作为来自解析器 (520) 的符号 (521) 可用于环路滤波器单元 (556) 的参数, 然而, 视频压缩技术还可响应于在对已编码图片或已编码视频序列的先前 (按解码次序) 部分进行解码期间获得的元信息, 以及响应于先前重建且经过环路滤波的样本值。可以以各种顺序包括多种类型的环路滤波器, 作为环路滤波器单元 556 的一部分, 如下文将进一步详细描述。

[0065] 环路滤波器单元 (556) 的输出可以是样本流, 该样本流可输出到呈现设备 (512) 以及存储在参考图片存储器 (557) 中以用于将来的帧间图片预测。

[0066] 一旦完全重建, 某些已编码图片就可用作参考图片以用于将来的帧间图片预测。例如, 一旦对应于当前图片的已编码图片被完全重建, 且已编码图片 (通过例如解析器 (520)) 被识别为参考图片, 则当前图片缓冲器 (558) 可变成参考图片存储器 (557) 的一部分, 且可以在开始重建后续已编码图片之前重新分配新的当前图片缓冲器。

[0067] 视频解码器 (510) 可根据诸如 ITU-T H.265 建议书的标准中采用的预定视频压缩技术执行解码操作。在已编码视频序列遵循视频压缩技术或标准的语法以及视频压缩技术或标准中记录的配置文件的意义下, 已编码视频序列可符合所使用的视频压缩技术或标准指定的语法。具体而言, 配置文件可以从视频压缩技术或标准中可用的所有工具中选择某些工具, 作为在该配置文件下可供使用的仅有工具。为了符合标准, 已编码视频序列的复杂度可处于视频压缩技术或标准的层级所限定的范围内。在一些情况下, 层级限制最大图片大小、最大帧率、最大重建采样率 (以例如每秒兆 (mega) 个样本为单位进行测量)、最大参考图片大小等。在一些情况下, 由层级设定的限制可通过假想参考解码器 (HRD) 规范和在已编码视频序列中用信号表示的 HRD 缓冲器管理的元数据来进一步限定。

[0068] 在一些示例性实施例中, 接收器 (531) 可以在接收已编码视频时接收附加 (冗余) 数据。附加数据可以被包括作为已编码视频序列的一部分。附加数据可由视频解码器 (510) 使用来对数据进行适当解码和/或更准确地重建原始视频数据。附加数据可采用例如时间、

空间或信噪比 (SNR) 增强层、冗余切片、冗余图片、前向纠错码等形式。

[0069] 图6示出了根据本公开的一个示例性实施例的视频编码器 (603) 的框图。视频编码器 (603) 可包括在电子设备 (620) 中。电子设备 (620) 可进一步包括传输器 (640) (例如, 传输电路)。视频编码器 (603) 可用于代替图4的示例中的视频编码器 (403)。

[0070] 视频编码器 (603) 可以从视频源 (601) (并非图6的示例中的电子设备 (620) 的一部分) 接收视频样本, 视频源 (601) 可采集将由视频编码器 (603) 编码的视频图像。在另一示例中, 视频源 (601) 可实现为电子设备 (620) 的一部分。

[0071] 视频源 (601) 可提供将由视频编码器 (603) 编码的呈数字视频样本流形式的源视频序列, 该数字视频样本流可具有任何合适的位深度 (例如: 8位、10位、12位.....)、任何色彩空间 (例如BT.601YCrCb, RGB, XYZ.....) 和任何合适的采样结构 (例如YCrCb 4:2:0, YCrCb 4:4:4)。在媒体服务系统中, 视频源 (601) 可以是能够存储先前已准备的视频的存储设备。在视频会议系统中, 视频源 (601) 可以是采集本地图像信息作为视频序列的相机。视频数据可作为多个单独的图片或图像来提供, 当按顺序观看时, 这些图片或图像被赋予运动。图片本身可构建为空间像素阵列, 其中取决于所使用的采样结构、色彩空间等, 每个像素可包括一个或多个样本。本领域的普通技术人员可容易地理解像素和样本之间的关系。下文侧重于描述样本。

[0072] 根据一些示例性实施例, 视频编码器 (603) 可实时地或在应用所要求的任何其它时间约束下, 将源视频序列的图片编码并压缩成已编码视频序列 (643)。施行适当的编码速度构成了控制器 (650) 的一个功能。在一些实施例中, 控制器 (650) 可以在功能上耦接到如下文所描述的其它功能单元且控制所述的其它功能单元。为了简单起见, 图中未描绘耦接。由控制器 (650) 设置的参数可包括速率控制相关参数 (图片跳过、量化器、率失真优化技术的 λ 值.....)、图片大小、图片群组 (GOP) 布局、最大运动矢量搜索范围等。控制器 (650) 可配置成具有其它合适的功能, 这些功能涉及针对某一系统设计优化的视频编码器 (603)。

[0073] 在一些示例性实施例中, 视频编码器 (603) 可配置成在编码环路中进行操作。作为过于简化的描述, 在一个示例中, 编码环路可包括源编码器 (630) (例如, 负责基于待编码的输入图片和参考图片来创建符号, 例如符号流) 和嵌入于视频编码器 (603) 中的 (本地) 解码器 (633)。解码器 (633) 重建符号以用类似于 (远程) 解码器可创建样本数据的方式创建样本数据, 即使所嵌入的解码器633处理在不进行熵编码的情况下由源编码器630编码的视频流 (因为在所公开的主题中考虑的视频压缩技术中, 熵编码中的符号与已编码视频码流之间的任何压缩可以是无损的)。重建的样本流 (样本数据) 输入到参考图片存储器 (634)。由于符号流的解码产生与解码器位置 (本地或远程) 无关的位精确结果, 因此参考图片存储器 (634) 中的内容在本地编码器和远程编码器之间也是按比特位精确对应的。换句话说, 编码器的预测部分“看到”的参考图片样本与解码器将在解码期间使用预测时所“看到”的样本值完全相同。这种参考图片同步性基本原理 (以及在例如因信道误差而无法维持同步性的情况下产生的漂移) 用于改善编码质量。

[0074] “本地”解码器 (633) 的操作可与诸如已在上文结合图5详细描述的视频解码器 (510) 的“远程”解码器相同。然而, 另外简要参考图5, 由于符号可用且熵编码器 (645) 和解析器 (520) 能够无损地将符号编码/解码成已编码视频序列, 因此包括缓冲存储器 (515) 和解析器 (520) 的视频解码器 (510) 的熵解码部分可能无法完全在编码器中的本地解码器

(633) 中实现。

[0075] 此时可以观察到,除了仅可存在于解码器中的解析/熵解码之外的任何解码器技术,还可必定需要以基本上相同的功能形式存在于对应的编码器中。出于此原因,所公开的主题有时可侧重于解码器操作,解码器操作与编码器的解码部分结合。因此,可简化编码器技术的描述,因为编码器技术与全面地描述的解码器技术互逆。在下文中,仅在某些区域或方面,提供编码器的更详细的描述。

[0076] 在操作期间,在一些示例性实现方式中,源编码器(630)可执行运动补偿预测编码,通过参考来自视频序列中被指定为“参考图片”的一个或多个先前已编码图片,该运动补偿预测编码对输入图片进行预测性编码。以这种方式,编码引擎(632)对输入图片的像素块和参考图片的像素块之间的颜色通道中的差异(或残差)进行编码,该参考图片可被选作该输入图片的预测参考。术语“残差”及其形容词形式“残差的”可互换地使用。

[0077] 本地视频解码器(633)可基于源编码器(630)创建的符号,对可指定为参考图片的图片的已编码视频数据进行解码。编码引擎(632)的操作有利地可以是有损过程。当已编码视频数据可以在视频解码器(图6未示出)中被解码时,已重建视频序列通常可以是带有一些误差的源视频序列的副本。本地视频解码器(633)复制解码过程,该解码过程可由视频解码器对参考图片执行,且可使已重建参考图片存储在参考图片高速缓存(634)中。以这种方式,视频编码器(603)可以在本地存储已重建参考图片的副本,该副本与将由远端(远程)视频解码器获得的已重建参考图片具有共同内容(不存在传输误差)。

[0078] 预测器(635)可针对编码引擎(632)执行预测搜索。即,对于待编码的新图片,预测器(635)可以在参考图片存储器(634)中搜索可用作新图片的适当预测参考的样本数据(作为候选参考像素块)或某些元数据,例如参考图片运动矢量、块形状等。预测器(635)可基于样本块逐像素块操作,以找到合适的预测参考。在一些情况下,如由预测器(635)获得的搜索结果所确定的,输入图片可具有从参考图片存储器(634)中存储的多个参考图片取得的预测参考。

[0079] 控制器(650)可管理源编码器(630)的编码操作,包括例如设置用于对视频数据进行编码的参数和子群参数。

[0080] 可以在熵编码器(645)中对所有上述功能单元的输出进行熵编码。熵编码器(645)根据诸如霍夫曼编码、可变长度编码、算术编码等的技术来对各种功能单元生成的符号进行无损压缩,从而将该符号转换成已编码视频序列。

[0081] 传输器(640)可缓冲由熵编码器(645)创建的已编码视频序列,从而为通过通信信道(660)进行传输做准备,通信信道(660)可以是通向可存储已编码视频数据的存储设备的硬件/软件链路。传输器(640)可以将来自视频编码器(603)的已编码视频数据与待传输的其它数据合并,其它数据例如是已编码音频数据和/或辅助数据流(未示出来源)。

[0082] 控制器(650)可管理视频编码器(603)的操作。在编码期间,控制器(650)可以向每个已编码图片分配某一已编码图片类型,但这可能影响可应用于相应的图片的编码技术。例如,通常可以将图片分配为以下任一种图片类型:

[0083] 帧内图片(I图片),其可以是不将序列中的任何其它图片用作预测源就可被编码和解码的图片。一些视频编解码器容许不同类型的帧内图片,包括例如独立解码器刷新(“IDR”)图片。本领域的普通技术人员了解I图片的这些变体及其相应的应用和特征。

[0084] 预测性图片(P图片),其可以是可使用帧内预测或帧间预测进行编码和解码的图片,该帧内预测或帧间预测使用至多一个运动矢量和参考索引来预测每个块的样本值。

[0085] 双向预测性图片(B图片),其可以是可使用帧内预测或帧间预测进行编码和解码的图片,该帧内预测或帧间预测使用至多两个运动矢量和参考索引来预测每个块的样本值。类似地,多个预测性图片可使用多于两个参考图片和相关联的元数据来用于重建单个块。

[0086] 源图片通常可以在空间上细分成多个样本编码块(例如, 4×4 、 8×8 、 4×8 或 16×16 个样本的块),且逐块进行编码。这些块可参考其它(已编码)块进行预测性编码,其它(已编码)块由应用于块的相应图片的编码分配来确定。例如,I图片的块可进行非预测性编码,或者I图片的块可参考同一图片的已编码块来进行预测性编码(空间预测或帧内预测)。P图片的像素块可参考一个先前编码的参考图片通过空间预测或通过时间预测来进行预测性编码。B图片的块可参考一个或两个先前编码的参考图片通过空间预测或通过时间预测来进行预测性编码。出于其它目的,源图片或中间处理的图片可细分成其它类型的块。编码块和其它类型的块的划分可以或可以不遵循相同的方式,如下文进一步详细描述。

[0087] 视频编码器(603)可根据诸如ITU-T H.265建议书的预定视频编码技术或标准执行编码操作。在操作中,视频编码器(603)可执行各种压缩操作,包括利用输入视频序列中的时间和空间冗余的预测性编码操作。因此,已编码视频数据可符合所使用的视频编码技术或标准指定的语法。

[0088] 在一些示例性实施例中,传输器(640)可以在传输已编码视频时传输附加数据。源编码器(630)可包括此类数据作为已编码视频序列的一部分。附加数据可包括时间/空间/SNR增强层、诸如冗余图片和切片的其它形式的冗余数据、SEI消息、VUI参数集片段等。

[0089] 所采集的视频可作为呈时间序列的多个源图片(视频图片)。帧内图片预测(通常简化为帧内预测)利用给定图片中的空间相关性,而帧间图片预测则利用图片之间的时间或其它相关性。例如,可以将正在编码/解码的特定图片分成块,正在编码/解码的特定图片称为当前图片。在当前图片中的块类似于视频中先前已编码且仍被缓冲的参考图片中的参考块时,可通过称为运动矢量的矢量对当前图片中的块进行编码。运动矢量指向参考图片中的参考块,且在使用多个参考图片的情况下,运动矢量可具有识别参考图片的第三个维度。

[0090] 在一些示例性实施例中,双向预测技术可用于帧间图片预测。根据这种双向预测技术,使用两个参考图片,例如按解码次序在视频中的当前图片之前(但是按显示次序可能分别是过去或将来)的第一参考图片和第二参考图片。可通过指向第一参考图片中的第一参考块的第一运动矢量和指向第二参考图片中的第二参考块的第二运动矢量来对当前图片中的块进行编码。可通过第一参考块和第二参考块的组合来联合地预测该块。

[0091] 此外,合并模式技术可用于帧间图片预测以提高编码效率。

[0092] 根据本公开的一些示例性实施例,诸如帧间图片预测和帧内图片预测的预测以块为单位来执行。例如,将视频图片序列中的图片划分成编码树单元(CTU)以用于压缩,图片中的CTU可具有相同大小,例如 64×64 像素、 32×32 像素或 16×16 像素。通常,CTU可包括三个平行的编码树块(CTB):一个亮度CTB和两个色度CTB。可以将每个CTU递归地以二叉树拆分成一个或多个编码单元(CU)。例如,可以将 64×64 像素的CTU拆分成一个 64×64 像素的

CU,或4个 32×32 像素的CU。一个或多个 32×32 块中的每一个块可进一步拆分成4个 16×16 像素的CU。在一些示例性实施例中,可以在编码期间分析每个CU以确定多种预测类型之中用于CU的预测类型,例如帧间预测类型或帧内预测类型。根据时间和/或空间可预测性,可以将CU拆分成一个或多个预测单元(PU)。通常,每个PU包括亮度预测块(PB)和两个色度PB。在一个实施例中,编码(编码/解码)中的预测操作以预测块为单位来执行。可以以各种空间模式执行将CU拆分成PU(或不同颜色通道的PB)。例如,亮度或色度PB可包括针对样本的值(例如,亮度值)的矩阵,所述的样本例如是 8×8 像素、 16×16 像素、 8×16 像素、 16×8 像素等。

[0093] 图7示出了根据本公开的另一示例性实施例的视频编码器(703)的图。视频编码器(703)配置成接收视频图片序列中的当前视频图片内的样本值的处理块(例如,预测块),并将处理块编码到作为已编码视频序列的一部分的已编码图片中。在示例中,视频编码器(703)可用于代替图4的示例中的视频编码器(403)。

[0094] 例如,视频编码器(703)接收用于处理块的样本值的矩阵,该处理块例如是 8×8 样本的预测块等。然后,视频编码器(703)使用例如率失真优化(RDO)来确定是否使用帧内模式、帧间模式或双向预测模式来最佳地对处理块进行编码。当确定在帧内模式中对处理块进行编码时,视频编码器(703)可使用帧内预测技术以将处理块编码到已编码图片中;以及当确定在帧间模式或双向预测模式中对处理块进行编码时,视频编码器(703)可分别使用帧间预测或双向预测技术以将处理块编码到已编码图片中。在一些示例性实施例中,合并模式可用作帧间图片预测的子模式,其中,在不借助预测器外部的已编码运动矢量分量的情况下,从一个或多个运动矢量预测器导出运动矢量。在一些其它示例性实施例中,可存在适用于主题块的运动矢量分量。因此,视频编码器(703)可包括未在图7中明确示出的组件,例如用于确定处理块的预测模式的模式决策模块。

[0095] 在图7的示例中,视频编码器(703)包括如图7中的示例性布置所示的耦接在一起的帧间编码器(730)、帧内编码器(722)、残差计算器(723)、开关(726)、残差编码器(724)、通用控制器(721)和熵编码器(725)。

[0096] 帧间编码器(730)配置成接收当前块(例如,处理块)的样本,将该块与参考图片中的一个或多个参考块(例如,按显示次序是先前图片和后来图片中的块)进行比较,生成帧间预测信息(例如,根据帧间编码技术的冗余信息描述、运动矢量、合并模式信息),以及基于帧间预测信息使用任何合适的技术来计算帧间预测结果(例如,已预测块)。在一些示例中,参考图片是基于已编码视频信息,使用嵌入在图6的示例性编码器620中的解码单元633(如图7的残差解码器728所示,如下文进一步详细描述)来解码的已解码参考图片。

[0097] 帧内编码器(722)配置成接收当前块(例如,处理块)的样本,将该块与同一图片中已编码的块进行比较,以及在变换之后生成量化系数,且在一些情况下还生成帧内预测信息(例如,根据一个或多个帧内编码技术的帧内预测方向信息)。帧内编码器(722)可基于帧内预测信息和同一图片中的参考块计算帧内预测结果(例如,已预测块)。

[0098] 通用控制器(721)可配置成确定通用控制数据,且基于该通用控制数据控制视频编码器(703)的其它组件。在一个示例中,通用控制器(721)确定块的预测模式,且基于该预测模式将控制信号提供给开关(726)。例如,当该预测模式是帧内模式时,通用控制器(721)控制开关(726)以选择供残差计算器(723)使用的帧内模式结果,且控制熵编码器(725)以

选择帧内预测信息并将帧内预测信息包括在码流中;以及当该块的预测模式是帧间模式时,通用控制器(721)控制开关(726)以选择供残差计算器(723)使用的帧间预测结果,且控制熵编码器(725)以选择帧间预测信息并将帧间预测信息包括在码流中。

[0099] 残差计算器(723)可配置成计算所接收的块与从帧内编码器(722)或帧间编码器(730)选择的块的预测结果之间的差(残差数据)。残差编码器(724)可配置成对残差数据进行编码以生成变换系数。例如,残差编码器(724)可配置成将残差数据从空间域变换到频域,以生成变换系数。变换系数随后经受量化处理以获得量化变换系数。在各种示例性实施例中,视频编码器(703)还包括残差解码器(728)。残差解码器(728)配置成执行逆变换,且生成已解码残差数据。已解码残差数据可适当地由帧内编码器(722)和帧间编码器(730)使用。例如,帧间编码器(730)可基于已解码残差数据和帧间预测信息生成已解码块,且帧内编码器(722)可基于已解码残差数据和帧内预测信息生成已解码块。适当地处理已解码块以生成已解码图片,且已解码图片可以在存储器电路(未示出)中缓冲并用作参考图片。

[0100] 熵编码器(725)可配置成将码流格式化以包括已编码块,以及执行熵编码。熵编码器(725)配置成在码流中包括各种信息。例如,熵编码器(725)可配置成将通用控制数据、所选预测信息(例如,帧内预测信息或帧间预测信息)、残差信息和其它合适的信息包括在码流中。当在帧间模式或双向预测模式的合并子模式中对块进行编码时,可以不存在残差信息。

[0101] 图8示出了根据本公开的另一实施例的示例性视频解码器(810)的图。视频解码器(810)配置成接收作为已编码视频序列的一部分的已编码图片,且对已编码图片进行解码以生成已重建图片。在一个示例中,视频解码器(810)可用于代替图4的示例中的视频解码器(410)。

[0102] 在图8的示例中,视频解码器(810)包括如图8中的示例性布置所示的耦接在一起的熵解码器(871)、帧间解码器(880)、残差解码器(873)、重建模块(874)和帧内解码器(872)。

[0103] 熵解码器(871)可配置成根据已编码图片来重建某些符号,这些符号表示构成已编码图片的语法元素。此类符号可包括例如对块进行编码的模式(例如,帧内模式、帧间模式、双向预测模式、合并子模式或另一子模式)、可识别供帧内解码器(872)或帧间解码器(880)使用以进行预测的某些样本或元数据的预测信息(例如,帧内预测信息或帧间预测信息)、呈例如量化变换系数形式的残差信息等。在一个示例中,当预测模式是帧间或双向预测模式时,将帧间预测信息提供给帧间解码器(880);以及当预测类型是帧内预测类型时,将帧内预测信息提供给帧内解码器(872)。残差信息可经受逆量化并提供给残差解码器(873)。

[0104] 帧间解码器(880)可配置成接收帧间预测信息,且基于该帧间预测信息生成帧间预测结果。

[0105] 帧内解码器(872)可配置成接收帧内预测信息,且基于该帧内预测信息生成预测结果。

[0106] 残差解码器(873)可配置成执行逆量化以提取解量化的变换系数,且处理该解量化的变换系数,以将残差从频域变换到空间域。残差解码器(873)还可利用某些控制信息(用以包括量化器参数(QP)),这些控制信息可由熵解码器(871)提供(未描绘数据路径,因

为这仅仅是低数据量控制信息)。

[0107] 重建模块(874)可配置成在空间域中组合由残差解码器(873)输出的残差与预测结果(可由帧间预测模块或帧内预测模块输出,视情况而定)以形成已重建块,已重建块形成已重建图片的一部分,而已重建图片是已重建视频的一部分。应注意,还可执行诸如去块操作等其它合适的操作来改善视觉质量。

[0108] 应注意,可使用任何合适的技术来实现视频编码器(403)、视频编码器(603)和视频编码器(703)以及视频解码器(410)、视频解码器(510)和视频解码器(810)。在一些示例性实施例中,可使用一个或多个集成电路来实现视频编码器(403)、视频编码器(603)和视频编码器(703)以及视频解码器(410)、视频解码器(510)和视频解码器(810)。在另一实施例中,可使用执行软件指令的一个或多个处理器来实现视频编码器(403)、视频编码器(603)和视频编码器(603)以及视频解码器(410)、视频解码器(510)和视频解码器(810)。

[0109] 返回到帧内预测过程,在帧内预测过程中,通过相邻、下一个相邻或者一个或多个其它行的样本、或者它们的组合来预测块(例如,亮度预测块或色度预测块,或者如果没有进一步拆分成预测块,则是编码块)中的样本,以生成预测块。然后,可通过变换来处理正在编码的实际块和预测块之间的残差,之后进行量化。可使得各种帧内预测模式可用,且可以在码流中用信号表示与帧内模式选择相关的参数和其它参数。例如,各种帧内预测模式可涉及用于预测样本的一个或多个行位置、从一个或多个预测行中选择预测样本所沿的方向、以及其它特殊的帧内预测模式。

[0110] 例如,一组帧内预测模式(可互换地称为“帧内模式”)可包括预定数量的定向帧内预测模式。如上文与图1的示例性实现方式相关地描述的,这些帧内预测模式可对应于预定数量的方向,沿着这些方向选择块外部的样本(out-of-block samples)作为在特定块中预测的样本的预测。在另一特定示例性实现方式中,可支持和预先定义八(8)个主方向模式,这8个主方向模式对应于与水平轴线成45度至207度的角度。

[0111] 在帧内预测的一些其它实现方式中,为了进一步利用定向纹理中更多种类的空间冗余,定向帧内模式可进一步扩展到具有更细粒度的角度集。例如,上述8角度实现方式可配置成提供8个标称角度(nominal angles),如图9所示,这8个标称角度称为V_PRED, H_PRED, D45_PRED, D135_PRED, D113_PRED, D157_PRED, D203_PRED和D67_PRED,且对于每个标称角度,可添加预定数量的(例如,7个)更细角度。通过这种扩展,更大总数的(例如本示例中,56个)方向角度可用于帧内预测,对应于相同数量的预定定向帧内模式。预测角度可由标称帧内角度加上角度增量来表示。对于上文针对每个标称角度具有7个更细的角度方向的特定示例,角度增量可以是-3至3乘以步长,该步长为3度。

[0112] 在一些实现方式中,作为上述定向帧内模式的替代或者除了上述定向帧内模式之外,还可预先定义预定数量的非定向帧内预测模式,以及使得预定数量的非定向帧内预测模式可用。例如,可指定5个称为平滑帧内预测模式的非定向帧内模式。这些非定向帧内模式预测模式可具体地称为DC帧内模式,PAETH帧内模式,SMOOTH帧内模式,SMOOTH_V帧内模式和SMOOTH_H帧内模式。图10示出了在这些示例性非定向模式下对特定块的样本的预测。作为示例,图10示出了通过来自顶部相邻行和/或左侧相邻行的样本预测的4×4块1002。块1002中的特定样本1010可对应于位于块1002的顶部相邻行中的样本1010的正上方样本1004、作为顶部相邻行和左侧相邻行的交集的样本1010的左上样本1006、以及位于块1002

的左侧相邻行中的样本1010的正左侧样本1008。对于示例性DC帧内预测模式，左侧相邻样本1008和上方相邻样本1004的平均值可用作样本1010的预测值。对于示例性PAETH帧内预测模式，可提取顶部参考样本1004、左侧参考样本1008和左上方参考样本1006，然后，这三个参考样本中最接近(顶部+左侧-左上方)的那个值可设置为样本1010的预测值。对于示例性SMOOTH_V帧内预测模式，样本1010可通过沿着垂直方向对左上方相邻样本1006和左侧相邻样本1008进行二次插值来预测。对于示例性SMOOTH_H帧内预测模式，样本1010可通过沿着水平方向对左上方相邻样本1006和顶部相邻样本1004进行二次插值来预测。对于示例性SMOOTH帧内预测模式，样本1010可通过沿着垂直方向和水平方向的二次插值的平均值来预测。上述非定向帧内模式实现方式仅作为非限制性示例来说明。还考虑其它相邻行、样本的其它非定向选择、以及组合预测样本以预测预测块中的特定样本的方式。

[0113] 在不同编码层级(图片、切片、块、单元等)下，由编码器从上述定向或非定向模式中选择特定帧内预测模式，这可以在码流中用信号表示。在一些示例性实现方式中，首先可以用信号表示示例性8个标称定向模式和5个非角度平滑模式(总共13个选项)。然后，如果用信号表示的模式是8个标称角度帧内模式之一，则进一步用信号表示索引，以向对应的用信号表示的标称角度指示所选择的角度的增量。在一些其它示例性实现方式中，所有帧内预测模式可一起索引(例如，56个定向模式加上5个非定向模式，以产生61个帧内预测模式)以用于用信号表示。

[0114] 在一些示例性实现方式中，示例性56个或其它数量的定向帧内预测模式可利用统一定向预测器来实现，统一定向预测器将块的每个样本投影到参考子样本位置，且通过2抽头双线性滤波器对参考样本进行插值。

[0115] 在一些实现方式中，为了利用边缘上的参考捕获衰减空间相关性，可设计称为FILTER INTRA模式的附加滤波器模式。对于这些模式，除了块外部的样本之外，块内的预测样本可用作块内的一些片区的帧内预测参考样本。例如，可预先定义这些模式，并使得这些模式可用于至少亮度块(或仅亮度块)的帧内预测。可预先设计预定数量的(例如，五个)滤波器帧内模式，每个滤波器帧内模式由n抽头滤波器(例如，7抽头滤波器)的集合表示，从而反映例如4×2片区中的样本与邻近该片区的n个近邻之间的相关性。换句话说，n抽头滤波器的权重因子可依赖于位置。以8×8块、4×2片区和7抽头滤波器为例，如图11所示，8×8块1102可拆分成8个4×2片区。在图11中，这些片区由B0, B1, B2, B3, B4, B5, B6和B7表示。对于每个片区，其7个近邻(在图11中，由R0至R7指示)可用于预测当前片区中的样本。对于片区B0，所有近邻可能已重建。但是对于其它片区，一些近邻位于当前块中，因此可能未重建，然后紧邻的近邻的预测值用作参考。例如，不重建如图11指示的片区B7的所有近邻，因此替代地，使用近邻的预测样本。

[0116] 在帧内预测的一些实现方式中，一个颜色分量可使用一个或多个其它颜色分量来预测。颜色分量可以是YCrCb、RGB、XYZ颜色空间等中的任意一个分量。例如，可实现从亮度分量(例如，亮度参考样本)预测色度分量(例如，色度块)，该预测称为从亮度而来的色度或CfL。在一些示例性实现方式中，跨颜色预测仅可允许从亮度到色度。例如，色度块中的色度样本可建模为同时重建的亮度样本(coincident reconstructed luma samples)的线性函数。CfL预测可如下实现：

$$\text{CfL}(\alpha) = \alpha \times L^{\text{AC}} + \text{DC} \quad (1)$$

[0117] 其中, L^{AC} 指示亮度分量的AC贡献, α 指示线性模型的参数, DC指示色度分量的DC贡献。例如, AC分量是针对块的每个样本获得, 而DC分量是针对整个块获得。具体而言, 可以将已重建亮度样本子采样(subsampled)到色度分辨率, 然后可以从每个亮度值中减去平均亮度值(亮度的DC), 以形成亮度的AC贡献。然后将亮度的AC贡献用于等式(1)的线性模式, 以预测色度分量的AC值。为了根据亮度AC贡献估算或预测色度AC分量(而不是要求解码器计算缩放参数), 示例性CfL实现方式可基于原始色度样本确定参数 α , 并在码流中用信号表示参数 α 。这降低了解码器的复杂度且产生更精确的预测。至于色度分量的DC贡献, 在一些示例性实现方式中, 可使用色度分量内的帧内DC模式来计算色度分量的DC贡献。

[0118] 在参考行的一些示例性实现方式中, 可使用多线帧内预测。在这些实现方式中, 多于一个参考行可供在帧内预测中选择, 编码器决定并用信号表示使用哪个参考行来生成帧内预测。参考行索引可以在帧内预测模式之前用信号表示, 在用信号表示非零参考行索引的情况下, 只允许最可能的预测模式。参考图15, 描绘了4个参考行以及左上参考样本的示例(从参考行0到参考行3), 其中每个参考行由六个部段组成, 这六个部段即部段A至部段F(如1502-1512所指示的)。此外, 部段A和部段F可分别通过来自部段B和部段E的最接近样本来填充。

[0119] 然后, 可实现帧内预测块或帧间预测块的残差的变换, 之后进行变换系数的量化。为了执行变换, 在变换之前, 帧内编码块和帧间编码块可进一步分成多个变换块(有时可互换地用作“变换单元”, 即使术语“单元”通常用于表示三个颜色通道的集合, 例如, “编码单元”可包括亮度编码块和色度编码块)。在一些实现方式中, 可指定编码块(或预测块)的最大划分深度(术语“已编码块”可与“编码块”互换地使用)。例如, 这种划分不会超过2个层级。在帧内预测块和帧间预测块之间, 可以不同地处理将预测块划分成变换块的操作。然而, 在一些实现方式中, 在帧内预测块和帧间预测块之间, 这种划分可以相似。

[0120] 在一些示例性实现方式中, 对于帧内编码块, 可以以所有变换块具有相同大小的方式进行变换划分, 并以光栅扫描顺序对变换块进行编码。图12示出了帧内编码块的这种变换块划分的示例。具体而言, 图12示出了编码块1202通过中间层级二叉树拆分1204而划分成16个具有相同块大小的变换块, 如1206所示。用于编码的示例性光栅扫描顺序由图12中的顺序箭头示出。

[0121] 在一些示例性实现方式中, 对于帧间编码块, 可以以递归方式进行变换单元划分, 其中划分深度可达预定数量的层级(例如, 2个层级)。如图13所示, 拆分可针对任何子分区且在任何层级停止或递归地继续。具体地, 图13示出了一个示例, 其中块1302拆分成四个二叉树子块1304, 其中一个子块进一步拆分成四个二级变换块, 而其它子块的划分在第一层级之后停止, 从而产生总共7个具有两种不同大小的变换块。用于编码的示例性光栅扫描顺序进一步由图13中的顺序箭头示出。虽然图13示出了多达两级正方形变换块的四叉树拆分的示例性实现方式, 但是在一些生成实现方式中, 变换划分可支持1:1(正方形)、1:2/2:1和1:4/4:1变换块形状和大小, 其范围从 4×4 到 64×64 。在一些示例性实现方式中, 如果编码块小于或等于 64×64 , 则变换块划分仅可应用于亮度分量(换句话说, 在该条件下, 色度变换块可与编码块相同)。否则, 如果编码块宽度或高度大于64, 则亮度编码块和色度编码块分别可隐式地拆分成多个 $\min(W, 64) \times \min(H, 64)$ 变换块和 $\min(W, 32) \times \min(H, 32)$ 变换块。

[0122] 在一些示例性实现方式中, 如图16所示, 提供将编码块或预测块划分成变换块的

另一替代示例性方案。如图16所示,作为使用递归变换划分的替代,可根据编码块的变换类型将预定划分类型集应用于编码块。在图16所示的特定示例中,可应用6种示例性划分类型之一,以将编码块拆分成各种数量的变换块。这种方案可适用于编码块或预测块。在本公开中,术语“划分类型”通常可指的是对块(例如,预测块或编码块)进行划分的方式,其可指的是“变换划分类型”、“预测块划分类型”或“编码块划分类型”。此外,对于“变换划分类型”下的描述,相同的构思还可适用于“编码块划分类型”,反之亦然。

[0123] 更详细地说,如图16所示,图16的划分方案对任何给定的变换类型提供多达6种划分类型。在该方案中,可基于例如率失真成本对每个编码块或预测块分配变换类型。在一个示例中,可基于编码块或预测块的变换类型来确定分配给编码块或预测块的划分类型。特定划分类型可对应于变换块拆分大小和模式(或划分类型),如图16所示的4种划分类型所示。可预先定义各种变换类型和各种划分类型之间的对应关系。在下文中示出了示例性对应关系,其中大写字母标签指示可基于率失真成本分配给编码块或预测块的变换类型:

[0124] • PARTITION_NONE (划分_无):分配等于块大小的变换大小。

[0125] • PARTITION_SPLIT (划分_拆分):分配如下变换大小,该变换大小是块大小的宽度的1/2和块大小的高度的1/2。

[0126] • PARTITION_HORZ (划分_HORZ):分配如下变换大小,该变换大小与块大小具有相同宽度且是块大小的高度的1/2。

[0127] • PARTITION_VERT (划分_VERT):分配如下变换大小,该变换大小是块大小的宽度的1/2且与块大小具有相同高度。

[0128] • PARTITION_HORZ4 (划分_HORZ4):分配如下变换大小,该变换大小与块大小具有相同宽度且是块大小的高度的1/4。

[0129] • PARTITION_VERT4 (划分_VERT4):分配如下变换大小,该变换大小是块大小的宽度的1/4且与块大小具有相同高度。

[0130] 在上述示例中,对于划分的变换块,如图16所示的划分类型均包含统一变换大小。这仅仅是示例,而非限制。在一些其它实现方式中,在特定划分类型(或模式)中,混合变换块大小可用于划分的变换块。

[0131] 然后,可以对如上获得的每个变换块进行主变换。主变换本质上是将变换块中的残差从空间域移动到频域。在实际主变换的一些实现方式中,为了支持上述示例性扩展编码块划分,可允许多种变换大小(对于两个维度中的每个维度,其范围从4点到64点)和变换形状(正方形;宽高比为2:1/1:2和4:1/1:4的矩形)。

[0132] 转到实际主变换,在一些示例性实现方式中,2-D变换过程可涉及使用混合变换核(hybrid transform kernel)(例如,混合变换核可由用于已编码残差变换块的每个维度的不同1-D变换组成)。示例性1-D变换核可包括但不限于:a)4点DCT-2、8点DCT-2、16点DCT-2、32点DCT-2、64点DCT-2;b)4点不对称DST、8点不对称DST、16点不对称DST(DST-4,DST-7)及其翻转版本;c)4点恒等变换(identity transform)、8点恒等变换、16点恒等变换、32点恒等变换。对用于每个维度的变换核的选择可基于率失真(RD)准则。例如,在表1中列出了可实现的DCT-2和不对称DST的基函数。

表1:示例性主变换基函数(针对N点输入的DCT-2、DST-4和DST-7)

变换类型	基函数 $T_i(j)$, $i, j=0, 1, \dots, N-1$
DCT-2	$T_i(j) = \omega_0 \cdot \sqrt{\frac{2}{N}} \cdot \cos\left(\frac{\pi \cdot i \cdot (2j+1)}{2N}\right)$ $\text{其中 } \omega_0 = \begin{cases} \sqrt{\frac{2}{N}} & i = 0 \\ 1 & i \neq 0 \end{cases}$
DST-4	$T_i(j) = \sqrt{\frac{2}{N}} \cdot \sin\left(\frac{\pi \cdot (2i+1) \cdot (2j+1)}{4N}\right)$
DST-7	$T_i(j) = \sqrt{\frac{4}{2N+1}} \cdot \sin\left(\frac{\pi \cdot (2i+1) \cdot (j+1)}{2N+1}\right)$

[0133] 在一些示例性实现方式中,用于特定主变换实现方式的混合变换核的可用性可基于变换块大小和预测模式。在表2中列出了示例性依赖关系。对于色度分量,可以以隐式方式执行变换类型选择。例如,对于帧内预测残差,可根据帧内预测模式来选择变换类型,如表3所指定的。对于帧间预测残差,可根据共定位亮度块(co-located luma block)的变换类型选择来选择色度块的变换类型。因此,对于色度分量,在码流中不存在变换类型信令。

表2:AV1混合变换核及其基于预测模式和块大小的可用性。这里,→和↓指示水平维度和垂直维度;√和×指示针对该块大小和预测模式的内核可用性

变换类型	描述	预测模式	
		帧内	帧间
DCT_DCT	DCT → 和 ↓	✓ (所有块大小)	✓ (所有块大小)
ADST_DCT	ADST ↓ ; DCT →	✓ (块大小 ≤ 16 × 16)	✓ (块大小 ≤ 16 × 16)
DCT_ADST	DCT ↓ ; ADST →		
ADST_ADST	ADST ↓ 和 →		
FLIPADST_DCT	FLIPADST ↓ ; DCT →	×	✓ (块大小 ≤ 16 × 16)
DCT_FLIPADST	DCT ↓ ; FLIPADST →		
FLIPADST_FLIPADST	FLIPADST ↓ 和 →		
ADST_FLIPADST	ADST ↓ ; FLIPADST →		
FLIPADST_ADST	FLIPADST ↓ ; ADST →		
IDTX	IDTX ↓ 和 →	✓ (块大小 ≤ 16 × 16)	✓ (块大小 ≤ 32 × 32)
V_DCT	DCT ↓ ; IDTX →	✓ (块大小 < 16 × 16)	✓ (块大小 ≤ 16 × 16)
H_DCT	IDTX ↓ ; DCT →		
V_ADST	ADST ↓ ; IDTX →	×	✓ (块大小 < 16 × 16)
H_ADST	IDTX ↓ ; ADST		
	→		
V_FLIPADST	FLIPADST ↓ ; IDTX →	×	✓ (块大小 < 16 × 16)
H_FLIPADST	IDTX ↓ ; FLIPADST →		

表3:色度分量帧内预测残差的变换类型选择

帧内预测	垂直变换	水平变换
DC_PRED	DCT	DCT
V_PRED	ADST	DCT
H_PRED	DCT	ADST
D45_PRED	DCT	DCT
D135_PRED	ADST	ADST
D113_PRED	ADST	DCT
D157_PRED	DCT	ADST
D203_PRED	DCT	ADST
D67_PRED	ADST	DCT
SMOOTH_PRED	ADST	ADST
SMOOTH_V_PRED	ADST	DCT
SMOOTH_H_PRED	DCT	ADST
PAETH_PRED	ADST	ADST

[0134] 在一些实现方式中,可以对主变换系数执行二次变换。例如,如图14所示,可以在正向主变换和量化(在编码器处)之间以及在解量化和逆向主变换(在解码器侧)之间应用LFNST(低频不可分离变换),以进一步对主变换系数去相关(decorrelate),LFNST称为简化二次变换。本质上,LFNST可以取一部分主变换系数,例如低频部分(因此,从变换块的主变换系数的完整集“简化”),以进行二次变换。在示例性LFNST中,根据变换块大小,可应用 4×4 不可分离变换或 8×8 不可分离变换。例如, 4×4 LFNST可应用于较小的变换块(例如, $\min(\text{宽度}, \text{高度}) < 8$),而 8×8 LFNST可应用于较大的变换块(例如, $\min(\text{宽度}, \text{高度}) > 8$)。例如,如果对 8×8 变换块进行 4×4 LFNST,则只有 8×8 主变换系数的低频 4×4 部分会进一步经历二次变换。

[0135] 具体地如图14所示,变换块可以是 8×8 (或 16×16)。因此,变换块的正向主变换1402生成 8×8 (或 16×16)主变换系数矩阵1404,其中每个正方形单元表示 2×2 (或 4×4)部分。例如,正向LFNST的输入可以不是完整的 8×8 (或 16×16)主变换系数。例如, 4×4 (或 8×8)LFNST可用于二次变换。因此,只有主变换系数矩阵1404的 4×4 (或 8×8)低频主变换系数可用作LFNST的输入,如阴影部分(左上角)1406所指示。可以不对主变换系数矩阵的其余部分进行二次变换。因此,在二次变换之后,主变换系数的、受到LFNST的那部分变成二次变换系数,而未受到LFNST的其余部分(例如,矩阵1404的无阴影部分)保持对应的主变换系数。在一些示例性实现方式中,未受到二次变换的其余部分可全部设置为零系数。

[0136] 在下文中描述了LFNST中使用的不可分离变换的应用示例。为了应用示例性 4×4 LFNST, 4×4 输入块X(例如,表示主变换系数块的 4×4 低频部分,例如图14的主变换矩阵1404的阴影部分1406)可表示成:

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

[0137] 该2-D输入矩阵首先可以以示例性顺序线性化或扫描成矢量 $\vec{x}$:

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

[0138] 然后, 4×4 LFNST的不可分离变换可计算成 $\vec{F} = T \cdot \vec{X}$, 其中 $\vec{F}$ 指示输出的变换系数矢量, T 是 16×16 变换矩阵。然后, 使用该块的扫描顺序 (例如, 水平、垂直或对角线) 将所得到的 16×1 系数矢量 $\vec{F}$ 反向扫描成 4×4 块。索引较小的系数可与较小的扫描索引一起放置在 4×4 系数块中。以这样的方式, 可通过第二变换 T 进一步利用主变换系数 X 中的冗余, 从而提供额外的压缩增强。

[0139] 上述示例性LFNST基于直接矩阵乘法方法, 以应用不可分离变换, 使得在单次过程 (single pass) 中实现LFNST, 而不需要多次迭代。在一些进一步的示例性实现方式中, 可进一步降低示例性 4×4 LFNST的不可分离变换矩阵 (T) 的维度, 以最小化计算复杂度和存储变换系数的存储器空间需求。这种实现方式可称为简化非分离变换 (RST)。更详细地说, RST的主要思想是将 N (在上述示例中, N 为 $4 \times 4 = 16$, 但是对于 8×8 块, N 可等于64) 维矢量映射到不同空间中的 R 维矢量, 其中 N/R ($R < N$) 表示降维因子 (dimension reduction factor)。因此, 作为 $N \times N$ 变换矩阵的替代, RST矩阵变成 $R \times N$ 矩阵, 如下:

$$T_{R \times N} = \begin{bmatrix} t_{11} & t_{12} & t_{13} & \dots & t_{1N} \\ t_{21} & t_{22} & t_{23} & \dots & t_{2N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ t_{R1} & t_{R2} & t_{R3} & \dots & t_{RN} \end{bmatrix} \quad (4)$$

[0140] 其中, 变换矩阵的 R 行是 N 维空间的减小的 R 基。因此, 该变换将输入矢量或 N 维转换成减小的 R 维的输出矢量。因此, 如图14所示, 从主系数1406变换的二次变换系数1408在维度上以因子或 N/R 减小。图14中1408周围的三个正方形可以用零来填充。

[0141] RTS的逆变换矩阵可以是其正向变换的转置。对于示例性 8×8 LFNST (与上述 4×4 LFNST相比, 这里的描述更加多样化), 可应用示例性减小因子4, 因此 64×64 直接不可分离变换矩阵相应地简化成 16×64 直接矩阵。此外, 在一些实现方式中, 输入主系数的一部分 (而不是全部) 可线性化成为LFNST的输入矢量。例如, 只有示例性 8×8 输入主变换系数的一部分可线性化为上述 X 矢量。对于一个特定示例, 在 8×8 主变换系数矩阵的四个 4×4 象限之中, 可忽略右下角 (高频系数), 只有其它三个象限使用预定扫描顺序来线性化为 48×1 矢量, 而非 64×1 矢量。在这样的实现方式中, 不可分离变换矩阵可进一步从 16×64 简化成 16×48 。

[0142] 因此, 可以在解码器侧使用示例性简化的 48×16 逆RST矩阵, 以生成 8×8 核 (主) 变换系数的左上、右上和左下 4×4 象限。具体而言, 当对相同的变换集配置应用进一步简化的 16×48 RST矩阵 (而不是 16×64 RST) 时, 不可分离二次变换可以从 8×8 主系数块的三个 4×4 象限块 (排除了右下 4×4 块) 中提取矢量化的48个矩阵元素作为输入。在这样的实现方式中, 在二次变换中可忽略已删掉的右下 4×4 主变换系数。这种进一步简化的变换可以将 48×1 矢量转换成 16×1 输出矢量, 将 16×1 输出矢量反向扫描成 4×4 矩阵以填充图14的1408。1408周围的二次变换系数的三个正方形可以用零来填充。

[0143] 借助于RST的这种降维, 减少了存储所有LFNST矩阵的存储器使用量。例如, 在上述示例中, 与不存在降维的实现方式相比, 存储器使用量可以从10KB减小到8KB, 而性能的降低相当小。

[0144] 在一些实现方式中, 为了降低复杂度, 可进一步限制LFNST, 以仅当位于要受到

LFNST的主变换系数部分之外(例如,位于图14中1404的部分1406之外)的所有系数都不显著时才应用LFNST。因此,当应用LFNST时,所有的仅主变换系数(primary-only transform coefficients)(例如,图14的主系数矩阵1404的无阴影部分)可接近零。这种限制允许在最后显著位置上调节LFNST索引信令,因此避免一些额外的系数扫描,而当不应用该限制时,检查特定位置的显著系数可能需要一些额外的系数扫描。在一些实现方式中,LFNST的最坏情况处理(就每像素的乘法而言)可分别将 4×4 块和 8×8 块的不可分离变换限制为 8×16 变换和 8×48 变换。在这些情况下,当应用LFNST时,最后显著扫描位置(last-significant scan position)必须小于8,而其它尺寸则小于16。对于形状为 $4 \times N$ 和 $N \times 4$ 且 $N > 8$ 的块,上述限制意味着现在仅对左上 4×4 区域应用一次LFNST。由于在应用LFNST时,所有的仅主系数(primary-only coefficients)均为零,因此在这种情况下,主变换所需的操作数量减少。从编码器的角度来看,当测试LFNST变换时,可简化系数的量化。对于前16个系数(按扫描次序),必须最大限度地进行率失真优化量化(RDO),可强制其余系数为零。

[0145] 在一些示例性实现方式中,可用的RST内核可指定为多个变换集,其中每个变换集包括多个不可分离变换矩阵。例如,总共可存在4个变换集,每个变换集有2个不可分离变换矩阵(内核)供LFNST使用。这些内核可以预先离线训练,因此它们是数据驱动的。离线训练的变换核可存储在存储器中,或者可硬编码在编码或解码设备中,以在编码/解码过程期间使用。在编码或解码过程期间对变换集的选择可由帧内预测模式确定。可预先定义从帧内预测模式到变换集的映射。在表4中示出了这种预定映射的示例。例如,如表4所示,如果三个跨分量线性模型(CCLM)模式之一(INTRA_LT_CCLM、INTRA_T_CCLM或INTRA_L_CCLM)用于当前块(即 $81 \leq \text{predModeIntra} \leq 83$),则可以对当前色度块选择变换集0。对于每个变换集,可通过显式用信号表示的LFNST索引进一步指定所选择的不可分离二次变换候选。例如,在变换系数之后,索引可以在码流中对于每个帧内CU用信号表示一次。

表4:变换选择表

IntraPredMode	变换集索引
$\text{IntraPredMode} < 0$	1
$0 \leq \text{IntraPredMode} \leq 1$	0
$2 \leq \text{IntraPredMode} \leq 12$	1
$13 \leq \text{IntraPredMode} \leq 23$	2
$24 \leq \text{IntraPredMode} \leq 44$	3
$45 \leq \text{IntraPredMode} \leq 55$	2
$56 \leq \text{IntraPredMode} \leq 80$	1
$81 \leq \text{IntraPredMode} \leq 83$	0

[0146] 因为在上述示例性实现方式中,仅当位于第一系数子群或部分之外的所有系数都不有效时才限制LFNST可应用,所以LFNST索引编码取决于最后显著系数(last significant coefficient)的位置。此外,LFNST索引可被上下文编码,但是不依赖于帧内预测模式,且只有第一个bin可被上下文编码。此外,LFNST可应用于帧内切片和帧间切片中的帧内CU,以及亮度和色度。如果启用双树,则可分别用信号表示亮度和色度的LFNST索引。对于帧间切片(禁用双树),可以用信号表示单个LFNST索引,单个LFNST索引用于亮度和色度两者。

[0147] 在一些示例性实现方式中,当选择帧内子划分 (ISP) 模式时,可禁用LFNST且可以不用信号表示RST索引,原因是即使RST应用于每个可行的划分块,性能的改进也可能微不足道。此外,对ISP预测的残差禁用RST可降低编码复杂度。在一些进一步的实现方式中,当选择多元线性回归帧内预测 (MIP) 模式时,也可禁用LFNST且可以不用信号表示RST索引。

[0148] 考虑到由于现有的最大变换大小限制 (例如, 64×64) 而导致大于 64×64 (或代表最大变换块大小的任何其它预定大小) 的大CU被隐式地拆分 (例如, TU平铺 (TU tiling)), 因此LFNST索引搜索针对特定数量的解码流水线阶段可将数据缓冲增加四倍。因此,在一些实现方式中,允许LFNST的最大大小可限制为例如 64×64 。在一些实现方式中,仅在DCT2作为主变换时才启用LFNST。

[0149] 在一些其它实现方式中,通过定义例如12个二次变换集,给亮度分量提供帧内二次变换 (IST), 其中每个集中存在例如3个内核。帧内模式相关索引 (intra mode dependent index) 可用于选择变换集。一个集中的内核选择可基于用信号表示的语法元素。当DCT2或ADST用作水平变换和垂直主变换两者时,可启用IST。在一些实现方式中,根据块大小,可选择 4×4 不可分离变换或 8×8 不可分离变换。如果 $\min(\text{tx_宽度}, \text{tx_高度}) < 8$, 则可选择 4×4 IST。对于较大的块,可使用 8×8 IST。这里, tx_宽度 (tx_width) 和 tx_高度 (tx_height) 分别对应于变换块的宽度和高度。IST的输入可以是处于Z字形扫描顺序的低频主变换系数。

[0150] 视频编码或解码过程中的各种变换 (例如, 残差块中的样本的主变换或者主变换系数过程中的块的二次变换) 在仅使用可分离变换方案时,在捕获定向纹理图案 (例如, 45° 方向 (例如,基本上远离水平或垂直方向的方向) 的边缘) 方面可能不是十分有效。如上所述,在一些示例性实现方式中,一个或多个不可分离变换设计可用于主变换系数的二次变换。

[0151] 变换块划分和应用用于所划分的变换块的变换类型可相互关联。例如,某些变换类型可更适合于特定划分类型。例如,与递归划分 (例如之前在图13中描述的划分) 相比,在图16中示出且在上文描述的变换划分方案提供非递归划分类型。如果针对所有可用的划分模式 (例如,图16中的变换划分类型) 下划分的变换块,允许所有可用的变换类型,则在确定哪个变换划分类型用于获得变换块以及哪个变换类型用于每个划分的变换块时,编码器需要在大参数空间中执行优化。实际上,与其它变换类型相比,某个变换类型集通常更适合于变换划分类型的一个特定类型。在下文中描述的各种实现方式中,可考虑变换划分类型和变换类型之间的交互,并使用这种交互来获得方案以限制用于特定划分类型的允许变换类型,并且类似地限制用于特定变换类型的允许划分类型。这样的实现方式可以在对每个所划分的变换块确定变换划分模式和变换类型选择时,特别是在使用非递归变换划分的情况下,减少编码器的优化空间。

[0152] 这些示例性实现方式可单独地使用或者以任何顺序或任何方式组合。在上文和下文的公开内容中,术语“已编码块”、“编码块”等可用于指代执行预测或变换的图片单元。编码块可以是亮度编码块,或者可以是色度编码块。在一些情况下,已编码/编码块可指的是预测块。术语“块大小”用于指代编码块的宽度或高度,或者宽度和高度的最大值,或者宽度和高度的最小值,或者面积大小 (宽度*高度),或者宽高比 (宽度:高度,或高度:宽度)。

[0153] 多个候选主变换类型

[0154] 在一个实施例中,块可存在多个候选主变换类型。块可包括由划分产生的变换块。

主变换类型的选择和/或用信号表示可限制为预定变换划分类型集。预定变换划分类型集可以是更大的可用变换划分类型集(例如,图16中的完整的划分类型集)的子集。换句话说,仅当块的变换划分类型属于预定变换划分类型集时,才做出并用信号表示对主变换的选择。否则,例如,如果变换划分类型是其它类型,则可使用默认变换类型,而不是选择并用信号表示主变换类型。

[0155] 在一个实现方式中,主变换类型可包括以下至少之一:离散余弦变换(DCT)类型1至DCT类型8;非对称离散正弦变换(ADST);离散正弦变换(DST)类型1至DST类型8;线图变换(LGT);或者卡洛南-洛伊变换(Karhunen-Loeve Transform,KLT)。

[0156] 在一个实现方式中,预定变换划分类型集仅包括例如图16的各种变换划分类型中的PARTITION_NONE,即,变换块大小等于预测块(或编码块)大小。因此,仅当变换类型属于该预定集时,才可能选择和/或用信号表示主变换类型。

[0157] 在一个实现方式中,还可考虑划分的数量来确定预定变换划分类型集。例如,仅当特定变换划分类型的划分的数量小于或等于预定阈值时,才可能将特定变换划分类型视为预定变换划分类型集的一部分,可对该部分选择和/或用信号表示主变换类型。在一个实现方式中,预定阈值可以是1到16的整数。

[0158] 多个候选二次变换类型

[0159] 在一个实施例中,块可存在多个候选二次变换类型。块可包括由划分产生的变换块。二次变换类型的选择和/或用信号表示仅可应用于预定变换划分类型集。预定变换划分类型集可以是更大的可用变换划分类型集(例如,图16中的完整的变换划分类型集)的子集。换句话说,仅当块的变换划分类型属于预定变换划分类型集时,才做出并用信号表示对二次变换的选择。否则,例如,如果变换划分类型是其它类型,则可使用默认二次变换类型,而不是选择并用信号表示二次变换类型,或者可以不执行二次变换。

[0160] 在一个实现方式中,二次变换类型可包括KLT。KLT可配置成具有不同的内核。

[0161] 在一个实现方式中,预定变换划分类型集仅可包括例如图16的各种变换划分类型中的PARTITION_NONE,即,变换块大小等于预测块(或编码块)大小。因此,仅当变换类型属于该预定集时,才可选择和/或用信号表示二次变换类型。

[0162] 在一个实现方式中,还可考虑划分的数量来确定预定变换划分类型集。例如,仅当特定变换划分类型的划分的数量小于或等于预定阈值时,才可能将特定变换划分类型视为预定变换划分类型集的一部分,可对该部分选择和/或用信号表示二次变换类型。在一个实现方式中,预定阈值可以是1到16的整数。

[0163] 在一个实现方式中,二次变换类型的选择和/或用信号表示可基于变换划分类型和主变换类型的组合。组合可包括预定变换划分类型集中的变换划分类型和预定变换类型集中的主变换类型。例如,仅当变换划分类型为PARTITION_NONE且块所使用的主变换类型为DCT或ADST时,才可能需要选择和/或用信号表示二次变换类型。否则,例如,如果变换划分类型是其它类型,则可使用默认二次变换类型,而不是选择并用信号表示二次变换类型,或者可以不执行二次变换。

[0164] 变换相关信令

[0165] 在本公开中,公开了各种信令机制,目的是提高信令效率,并考虑了变换相关语法元素/参数的顺序。

[0166] 在一个实施例中,变换划分类型信息可以在主/二次变换类型选择信息之前用信号表示。仅当变换划分属于预定变换划分类型集(例如PARTITION_NONE)时,才需要用信号表示主/二次变换类型选择。否则,如果变换划分不属于预定变换划分类型集时,则可能不需要用信号表示主/二次变换类型选择。取而代之的是,主/二次变换类型可被导出为预定默认变换类型。

[0167] 在一个实施例中,主/二次变换类型选择信息可以在变换划分类型信息之前用信号表示。在这种情况下,变换划分类型的选择和/或用信号表示可取决于主/二次变换类型选择信息。

[0168] 在一个实现方式中,仅当主变换类型属于预定变换类型集时,才可能需要用信号表示变换划分类型信息。否则,可能不需要用信号表示变换划分类型信息。例如,预定变换类型集可包括但不限于DCT类型1至DCT类型8、ADST、DST类型1至DST类型8,LGT和KLT。

[0169] 在一个实现方式中,仅当二次变换类型属于预定变换类型集时,才可能需要用信号表示变换划分类型信息。作为示例,预定变换类型集可包括但不限于具有与预定KLT索引相关联的内核的特定KLT。否则,如果二次变换类型不属于预定变换类型集时,则可能不需要用信号表示变换划分类型信息。取而代之的是,变换划分类型信息可被导出为预定默认变换划分类型(例如PARTITION_NONE)。

[0170] 图17示出了用于对视频数据进行解码的示例性方法1700。方法1700可包括以下步骤中的一部分或全部步骤:步骤1710,接收数据块的已编码视频码流;步骤1720,从已编码视频码流中提取与数据块相关联的变换划分类型;步骤1730,响应于变换划分类型属于预定变换划分类型集的子集,预定变换划分类型集中的每个变换划分类型指定用于将数据块分割成变换块的拆分模式;提取与从数据块拆分的变换块相关联的变换的变换类型,该变换类型在已编码视频码流中用信号表示,其中,变换类型属于第一预定变换类型集;以及根据变换类型,对变换块执行逆变换。

[0171] 在本公开的实施例中,可根据需要以任何数量或顺序组合或排列任何步骤和/或操作。两个或更多个步骤和/或操作可并行地执行。

[0172] 本公开的实施例可单独地使用或者以任何顺序组合。此外,每种方法(或实施例)、编码器和解码器可通过处理电路(例如,一个或多个处理器或者一个或多个集成电路)来实现。在一个示例中,一个或多个处理器执行存储在非暂时性计算机可读介质中的程序。本公开的实施例可适用于亮度块或色度块。

[0173] 上述技术可实现为计算机软件,该计算机软件使用计算机可读指令,且物理地存储在一个或多个计算机可读介质中。例如,图18示出了适于实施所公开的主题的某些实施例的计算机系统(1800)。

[0174] 可使用任何合适的机器代码或计算机语言对计算机软件进行编码,任何合适的机器代码或计算机语言可经受汇编、编译、链接或类似的机制以创建包括指令的代码,指令可由一个或多个计算机中央处理单元(CPU)、图形处理单元(GPU)等直接执行,或者通过解释、微代码执行等执行。

[0175] 指令可以在各种类型的计算机或其组件上执行,计算机或其组件包括例如个人计算机、平板计算机、服务器、智能电话、游戏设备、物联网设备等。

[0176] 图18所示的计算机系统(1800)的组件本质上是示例性的,并不旨在对实施本公开

的实施例的计算机软件的用途或功能的范围提出任何限制。组件的配置也不应解释为具有与计算机系统(1800)的示例性实施例中所示的组件中的任何一个组件或组件的组合相关的任何依赖或要求。

[0177] 计算机系统(1800)可包括某些人机接口输入设备。此类人机接口输入设备可响应于一个或多个人类用户通过例如下述的输入:触觉输入(例如:击键、划动,数据手套移动)、音频输入(例如:语音、拍手)、视觉输入(例如:手势)、嗅觉输入(未描绘)。人机接口设备还可用于捕获不一定与人的意识输入直接相关的某些媒介,例如音频(例如:语音、音乐、环境声音)、图像(例如:扫描图像、从静止图像相机获取的拍摄图像)、视频(例如,二维视频、包括立体视频的三维视频)。

[0178] 人机接口输入设备可包括下述中的一项或多项(每种中仅示出一个):键盘(1801)、鼠标(1802)、触控板(1803)、触摸屏(1810)、数据手套(未示出)、操纵杆(1805)、麦克风(1806)、扫描仪(1807)、相机(1808)。

[0179] 计算机系统(1800)还可包括某些人机接口输出设备。此类人机接口输出设备可通过例如触觉输出、声音、光和气味/味道来刺激一个或多个人类用户的感官。此类人机接口输出设备可包括触觉输出设备(例如,触摸屏(1810)的触觉反馈、数据手套(未示出)或操纵杆(1805),但还可以是不作为输入设备的触觉反馈设备)、音频输出设备(例如:扬声器(1809)、耳机(未描绘))、视觉输出设备(例如,包括CRT屏幕、LCD屏幕、等离子屏幕、OLED屏幕的屏幕(1810),每种屏幕具有或没有触摸屏输入功能,每种屏幕具有或没有触觉反馈功能,其中的一些屏幕能够通过诸如立体图像输出之类的装置、虚拟现实眼镜(未描绘)、全息显示器和烟箱(未描绘)以及打印机(未描绘)来输出二维视觉输出或超过三维的输出。

[0180] 计算机系统(1800)还可包括人类可访问存储设备及其关联介质,例如包括具有CD/DVD等介质(1821)的CD/DVD ROM/RW(1820)的光学介质、指状驱动器(1822)、可拆卸硬盘驱动器或固态驱动器(1823)、诸如磁带和软盘之类的传统磁性介质(未描绘)、诸如安全软件狗之类的基于专用ROM/ASIC/PLD的设备(未描绘)等。

[0181] 本领域技术人员还应该理解,结合当前公开的主题所使用的术语“计算机可读介质”不涵盖传输介质、载波或其它暂时性信号。

[0182] 计算机系统(1800)还可包括通向一个或多个通信网络(1855)的接口(1854)。网络可例如是无线网络、有线网络、光网络。网络可进一步是本地网络、广域网络、城域网、车辆和工业网络、实时网络、延迟容忍网络等。网络的示例包括诸如以太网之类的局域网、无线LAN、包括GSM、3G、4G、5G、LTE等的蜂窝网络、包括有线电视、卫星电视和地面广播电视的电视有线或无线广域数字网络、包括CAN总线的车辆和工业网络等。某些网络通常需要附接到某些通用数据端口或外围总线(1849)的外部网络接口适配器(例如,计算机系统(1800)的USB端口);如下所述,其它网络接口通常通过附接到系统总线而集成到计算机系统(1800)的内核中(例如,连接到PC计算机系统以太网接口或连接到智能手机计算机系统蜂窝网络接口)。计算机系统(1800)可使用这些网络中的任何网络与其它实体通信。此类通信可以是仅单向接收的(例如,广播电视)、仅单向发送的(例如,连接到某些CANBus设备的CANBus)或双向的,例如,使用局域网或广域网数字网络连接到其它计算机系统。如上所述,可以在那些网络和网络接口中的每一个上使用某些协议和协议栈。

[0183] 上述人机接口设备、人机可访问的存储设备和网络接口可附接到计算机系统

(1800)的内核(1840)。

[0184] 内核(1840)可包括一个或多个中央处理单元(CPU)(1841)、图形处理单元(GPU)(1842)、现场可编程门区域(FPGA)(1843)形式的专用可编程处理单元、用于某些任务的硬件加速器(1844)、图形适配器(1850)等。这些设备以及只读存储器(ROM)(1845)、随机存取存储器(1846)、诸如内部非用户可访问的硬盘驱动器、SSD等之类的内部大容量存储器(1847)可通过系统总线(1848)连接。在一些计算机系统中,可以以一个或多个物理插头的形式访问系统总线(1848),以能够通过附加的CPU、GPU等进行扩展。外围设备可直接附接到内核的系统总线(1848)或通过外围总线(1849)附接到内核的系统总线(1848)。在一个示例中,屏幕(1810)可连接到图形适配器(1850)。外围总线的架构包括PCI、USB等。

[0185] CPU(1841)、GPU(1842)、FPGA(1843)和加速器(1844)可执行某些指令,这些指令可组合来构成上述计算机代码。该计算机代码可存储在ROM(1845)或RAM(1846)中。过渡数据还可存储在RAM(1846)中,而永久数据可例如存储在内部大容量存储器(1847)中。可通过使用高速缓存来进行通向任何存储设备的快速存储及检索,该高速缓存可与下述紧密关联:一个或多个CPU(1841)、GPU(1842)、大容量存储(1847)、ROM(1845)、RAM(1846)等。

[0186] 计算机可读介质可以在其上具有执行各种由计算机实现的操作的计算机代码。介质和计算机代码可以是出于本公开的目的而专门设计和构造的介质和计算机代码,或者介质和计算机代码可以是计算机软件领域的技术人员公知且可用的类型。

[0187] 作为非限制性示例,可由于一个或多个处理器(包括CPU、GPU、FPGA、加速器等)执行包含在一种或多种有形的计算机可读介质中的软件而使得具有架构(1800),特别是内核(1840)的计算机系统提供功能。此类计算机可读介质可以是与如上所介绍的用户可访问的大容量存储相关联的介质,以及某些非暂时性内核(1840)的存储器,例如内核内部大容量存储器(1847)或ROM(1845)。实施本公开的各个实施例的软件可存储在此类设备中并由内核(1840)执行。根据特定需要,计算机可读介质可包括一个或多个存储设备或芯片。软件可使得内核(1840),特别是其中的处理器(包括CPU、GPU、FPGA等)执行本文所描述的特定过程或特定过程的特定部分,包括定义存储在RAM(1846)中的数据结构以及根据由软件定义的过程来修改此类数据结构。除此之外或作为替代,可由于硬连线或以其它方式体现在电路(例如:加速器(1844))中的逻辑而使得计算机系统提供功能,该电路可替代软件或与软件一起运行以执行本文描述的特定过程或特定过程的特定部分。在适当的情况下,提及软件的部分可包含逻辑,反之亦然。在适当的情况下,提及计算机可读介质的部分可包括存储用于执行的软件的电路(例如,集成电路(IC))、体现用于执行的逻辑的电路或包括两者。本公开包括硬件和软件的任何合适的组合。

[0188] 虽然本公开已描述多个示例性实施例,但是存在落入本公开的范围内的修改、置换和各种替换等效物。因此,应理解,本领域技术人员将能够设计出许多虽然未在本文中明确示出或描述,但是体现了本公开的原理,因此落入本公开的精神和范围内的系统和方法。

附录A:首字母缩写词

JEM:联合探索模型

VVC:下一代视频编码

BMS:基准集

MV:运动矢量

HEVC: 高效视频编码
SEI: 辅助增强信息
VUI: 视频可用性信息
GOP: 图片群组
TU: 变换单元
PU: 预测单元
CTU: 编码树单元
CTB: 编码树块
PB: 预测块
HRD: 假想参考解码器
SNR: 信噪比
CPU: 中央处理单元
GPU: 图形处理单元
CRT: 阴极射线管
LCD: 液晶显示器
OLED: 有机发光二极管
CD: 光盘
DVD: 数字视频光盘
ROM: 只读存储器
RAM: 随机存取存储器
ASIC: 专用集成电路
PLD: 可编程逻辑器件
LAN: 局域网
GSM: 全球移动通信系统
LTE: 长期演进
CANBus: 控制器局域网络总线
USB: 通用串行总线
PCI: 互连外围组件
FPGA: 现场可编程门区域
SSD: 固态驱动器
IC: 集成电路
HDR: 高动态范围
SDR: 标准动态范围
JVET: 联合视频探索团队
MPM: 最可能模式
WAIP: 广角帧内预测
CU: 编码单元
PU: 预测单元
TU: 变换单元

CTU:编码树单元
PDPC:位置相关预测组合
ISP:帧内子分区
SPS:序列参数设置
PPS:图片参数集
APS:自适应参数集
VPS:视频参数集
DPS:解码参数集
ALF:自适应环路滤波器
SAO:采样自适应偏移
CC-ALF:跨分量自适应环路滤波器
CDEF:约束定向增强过滤器
CCS0:跨分量采样偏移
LS0:局部采样偏移
LR:环路恢复过滤器
AV1:AOMedia视频1
AV2:AOMedia视频2。

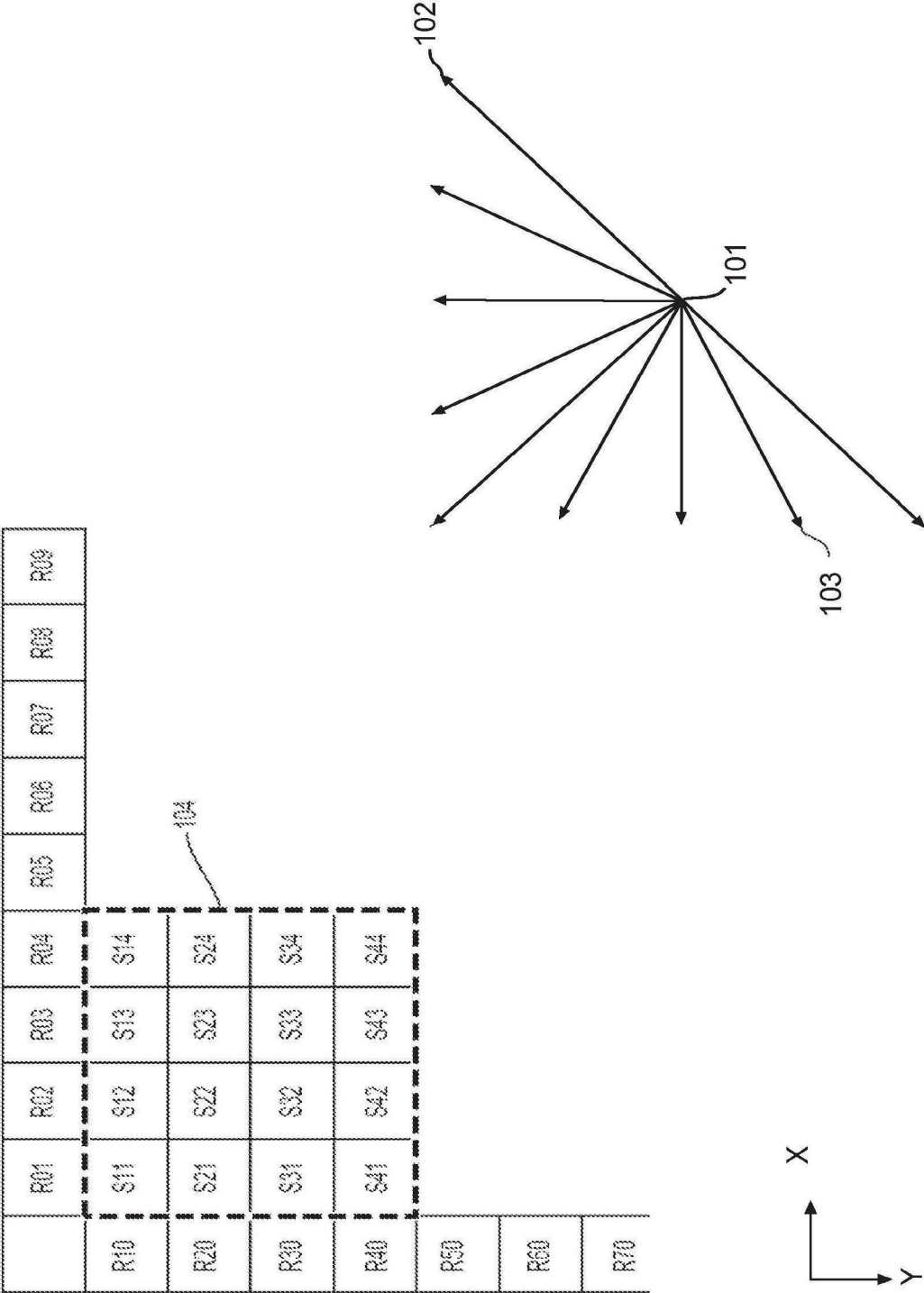


图1A

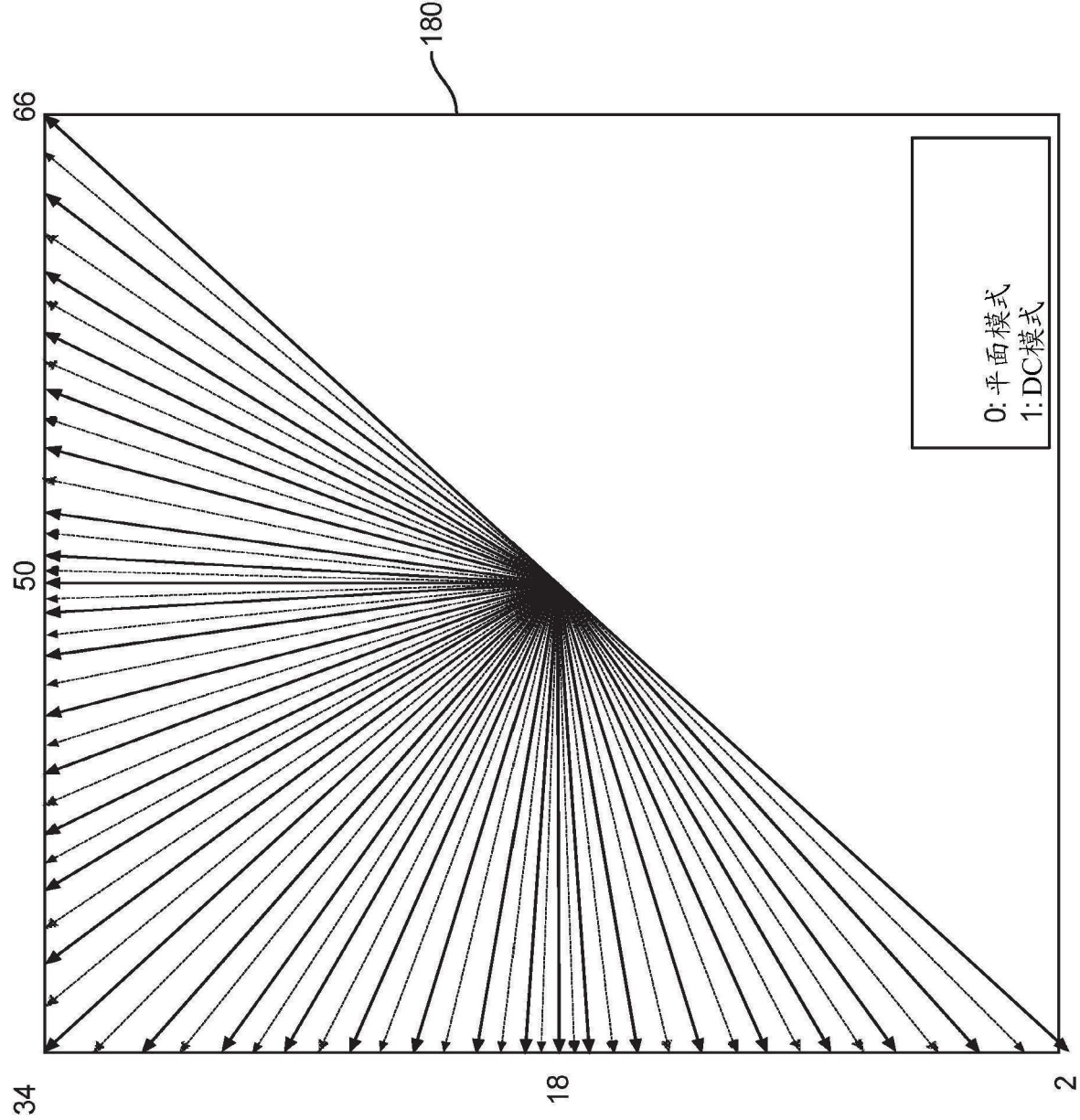


图1B

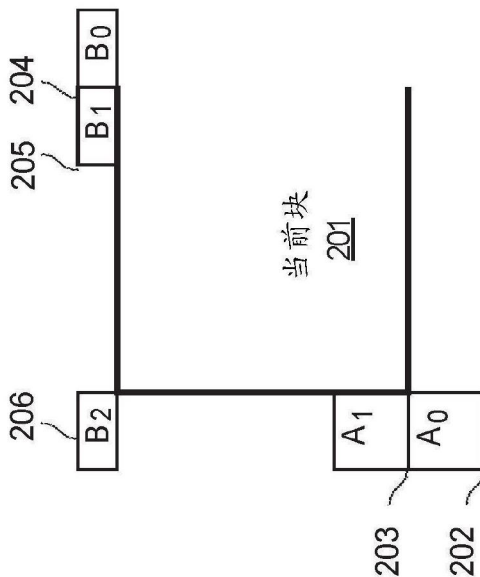


图2

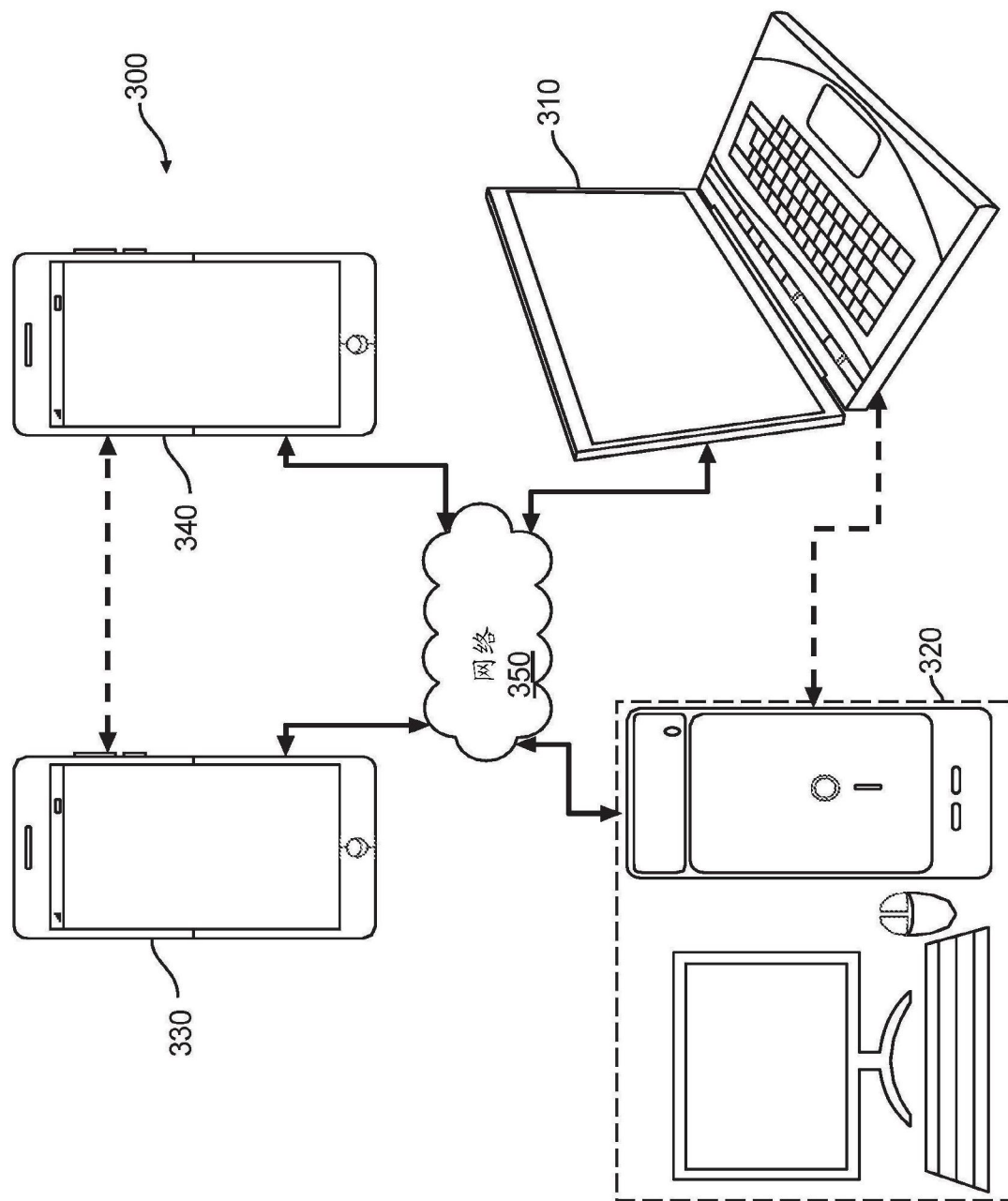


图3

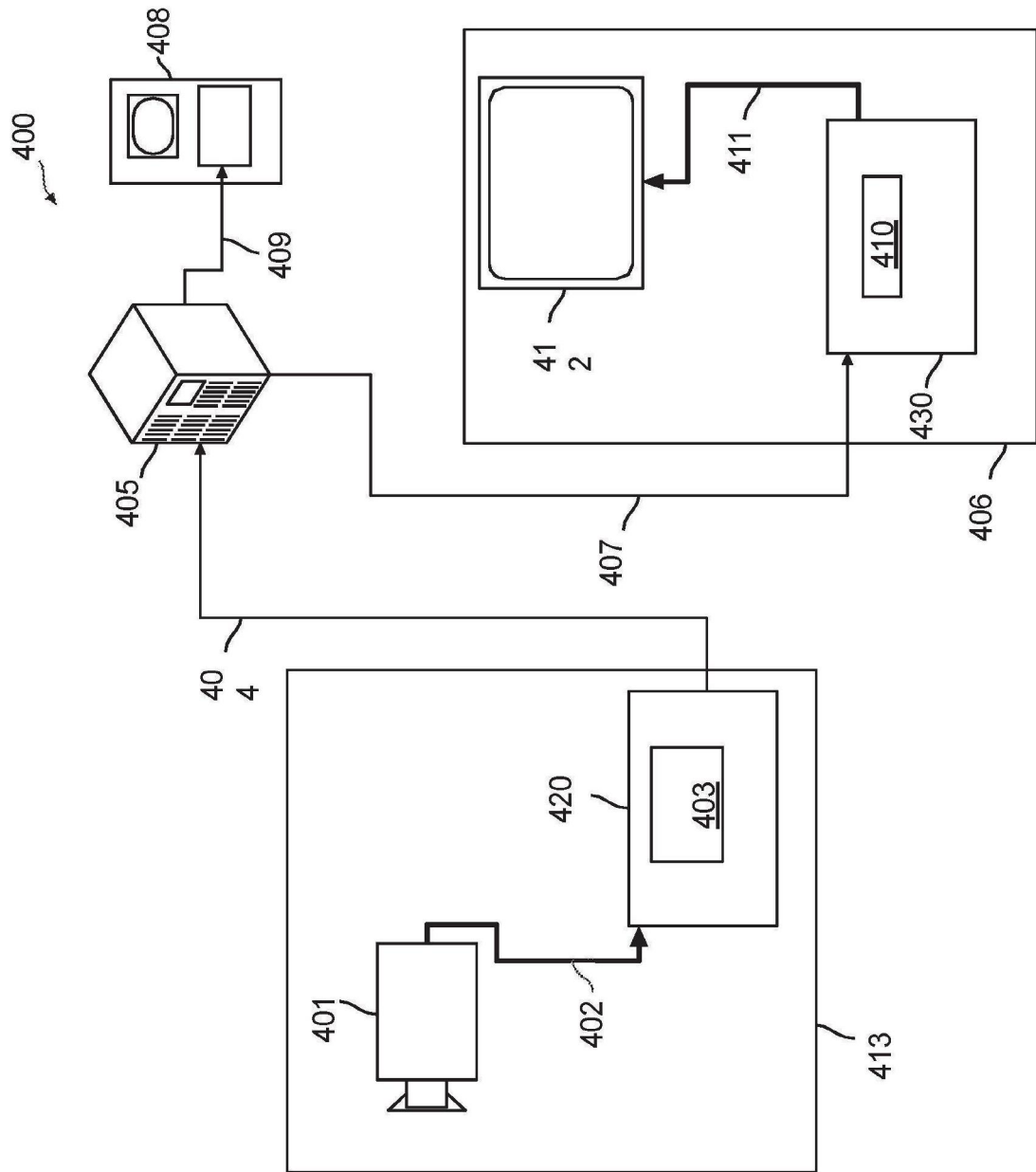


图4

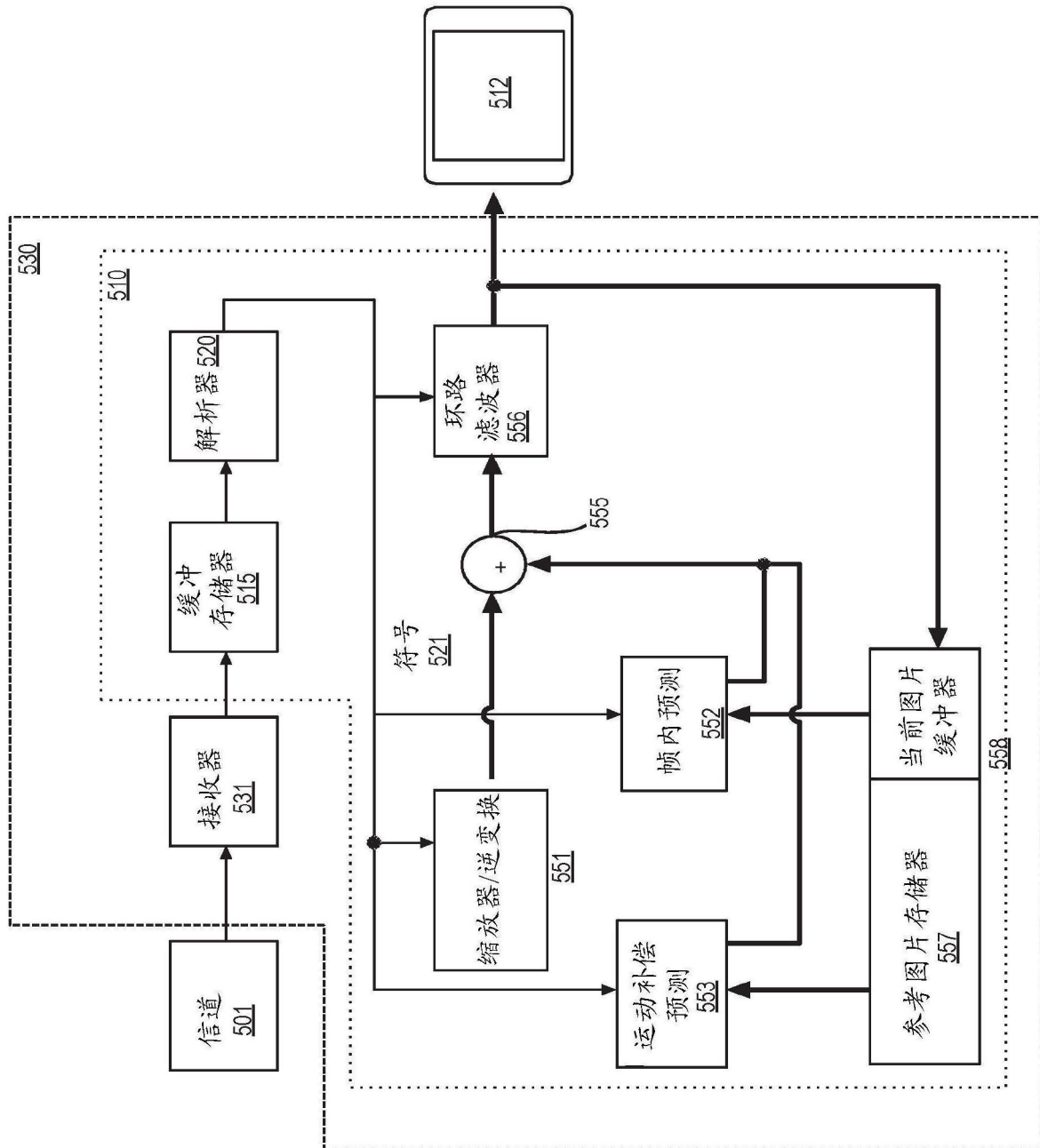


图5

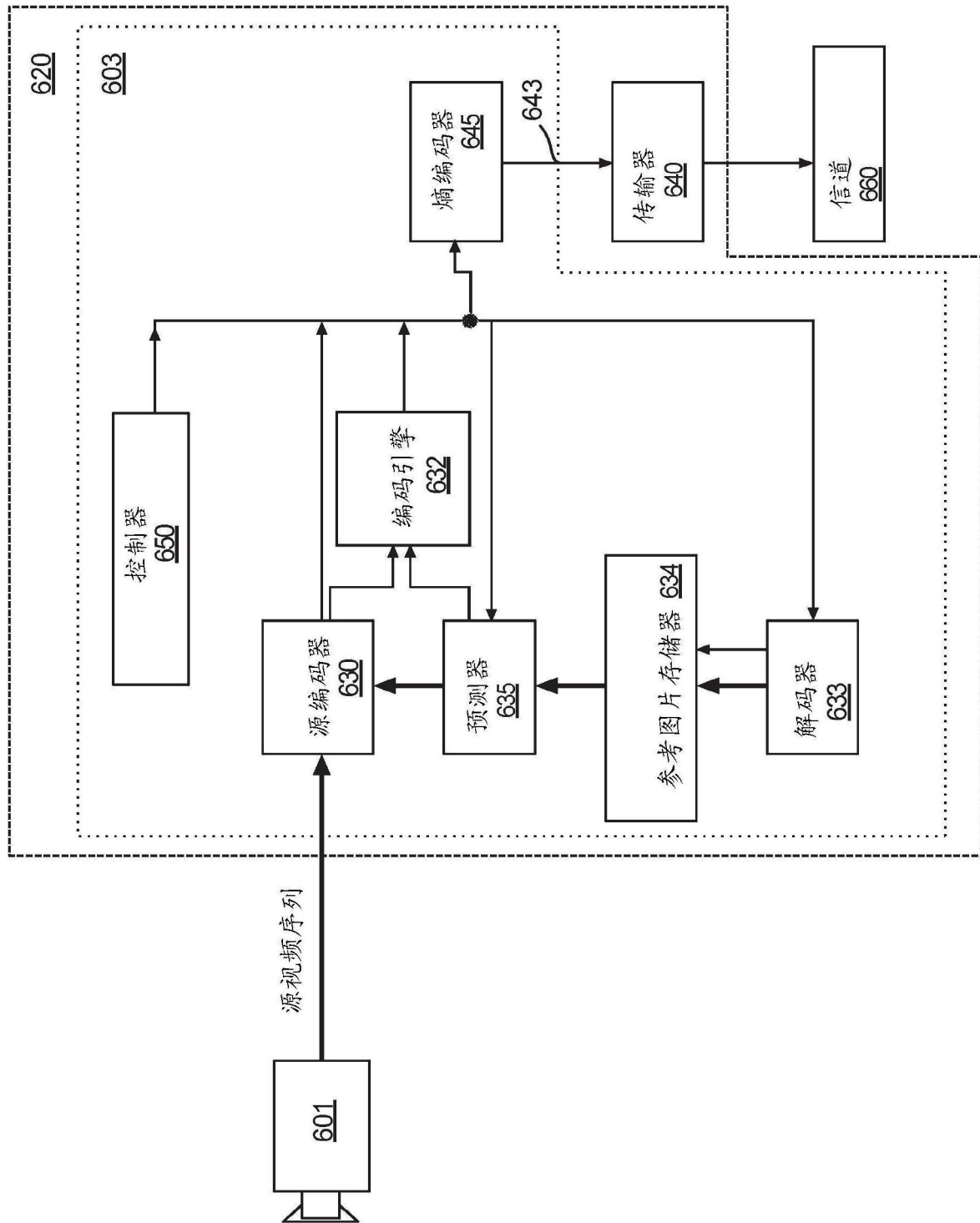


图6

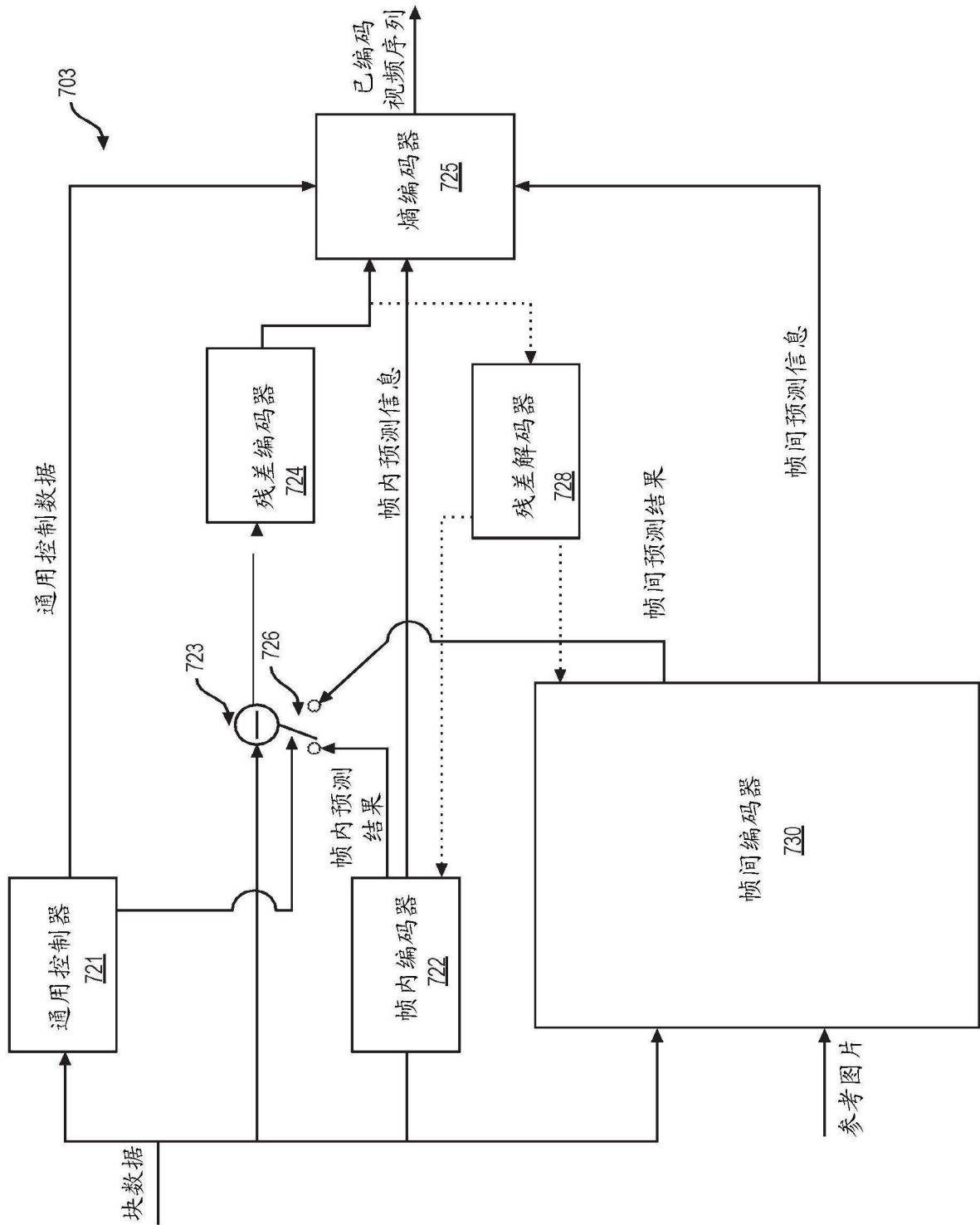


图7

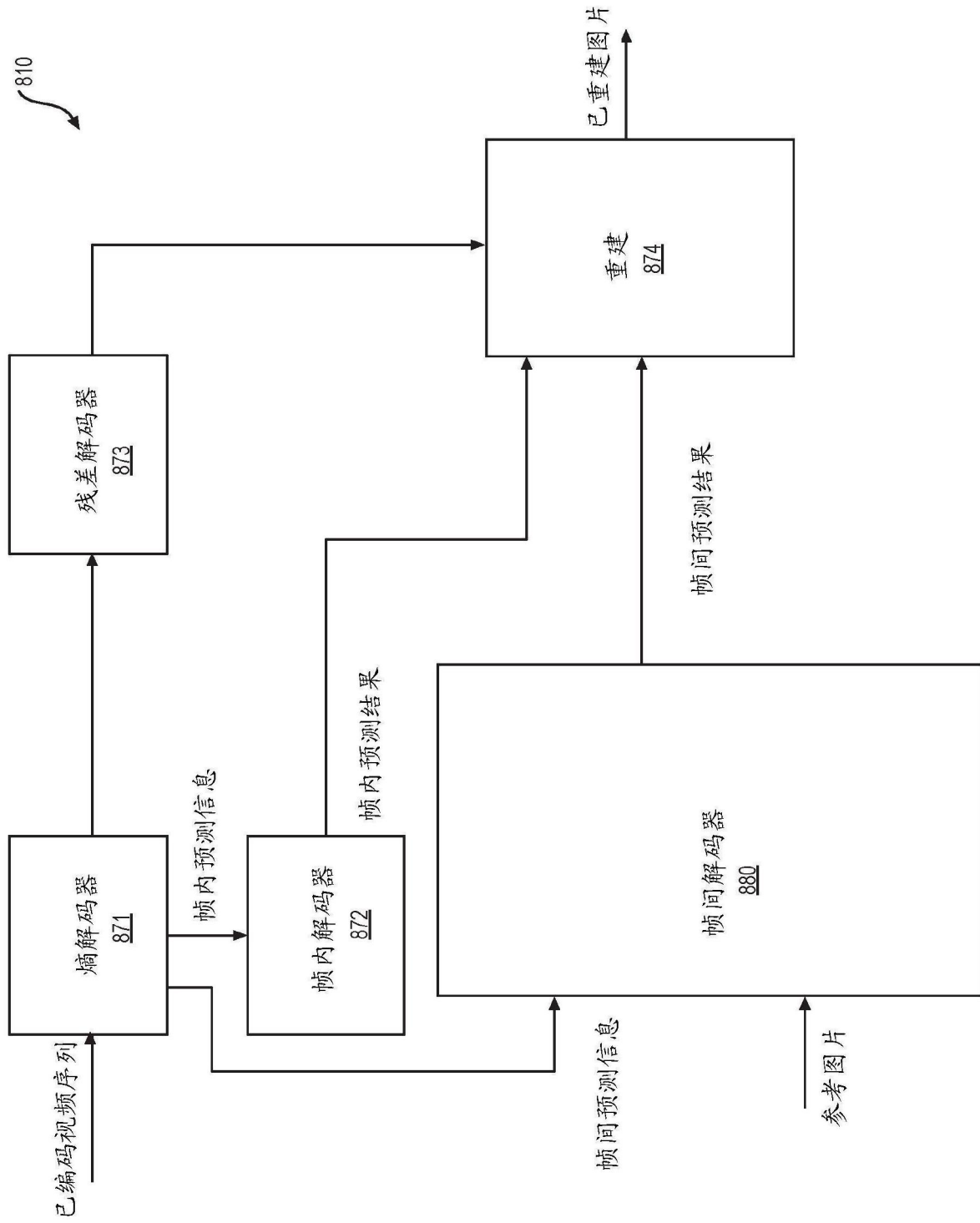


图8

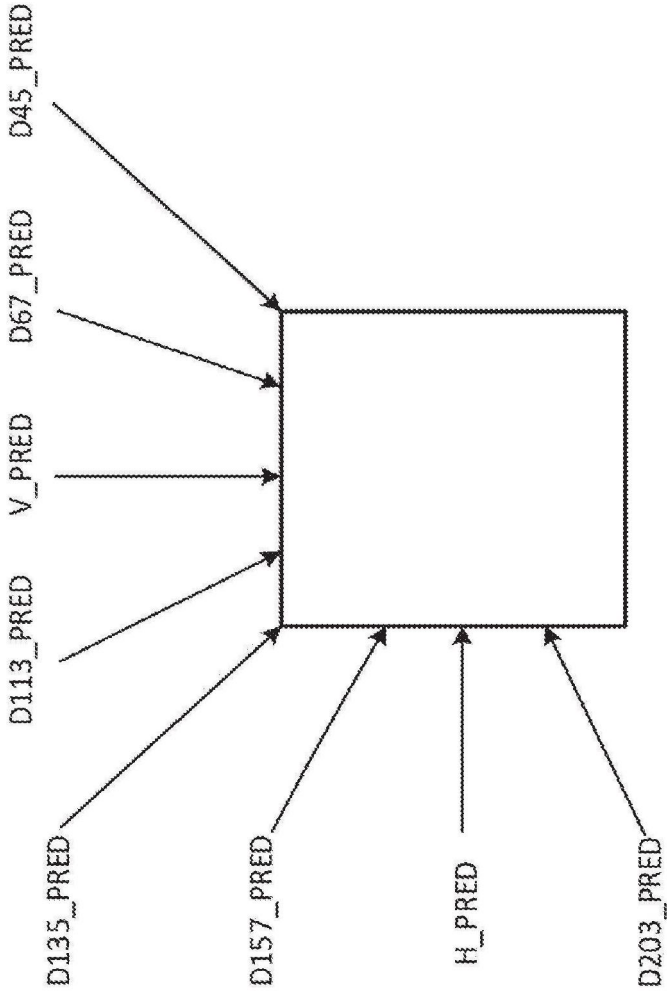


图9

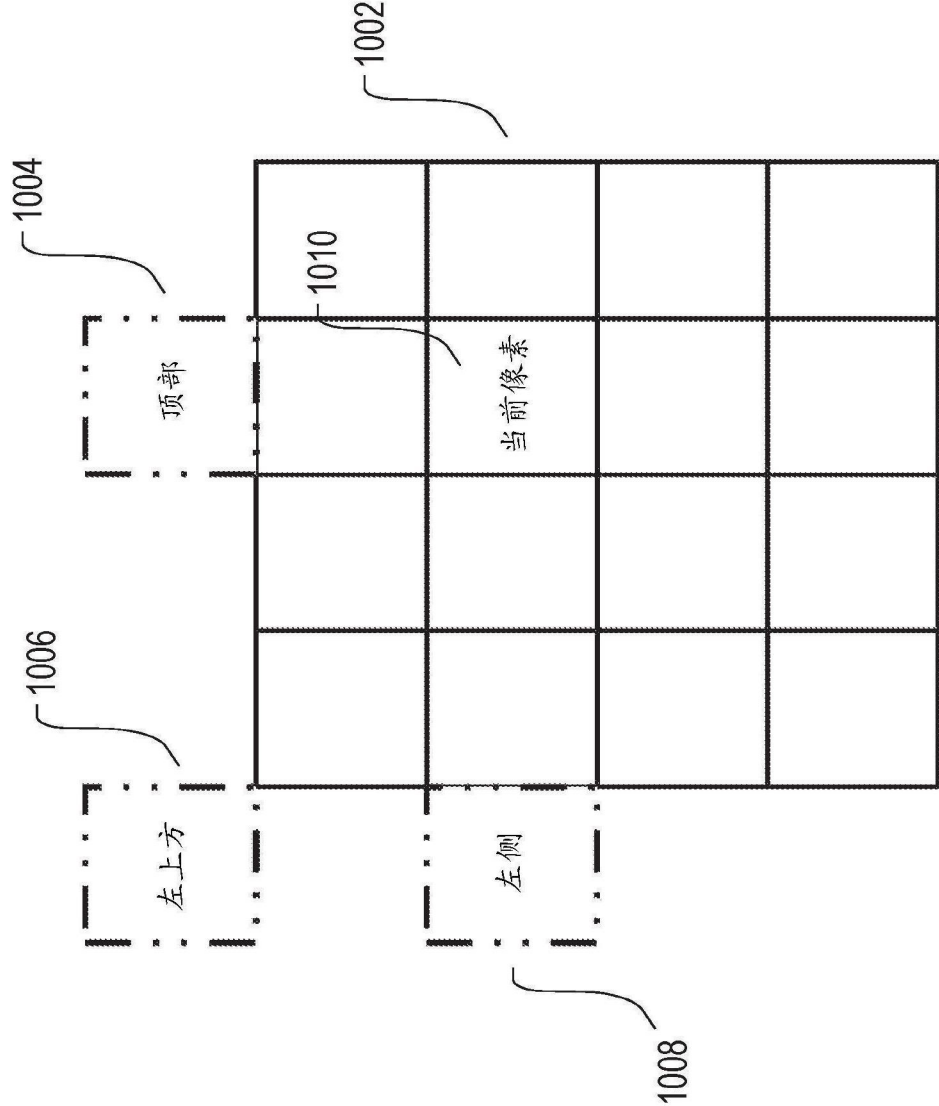


图10

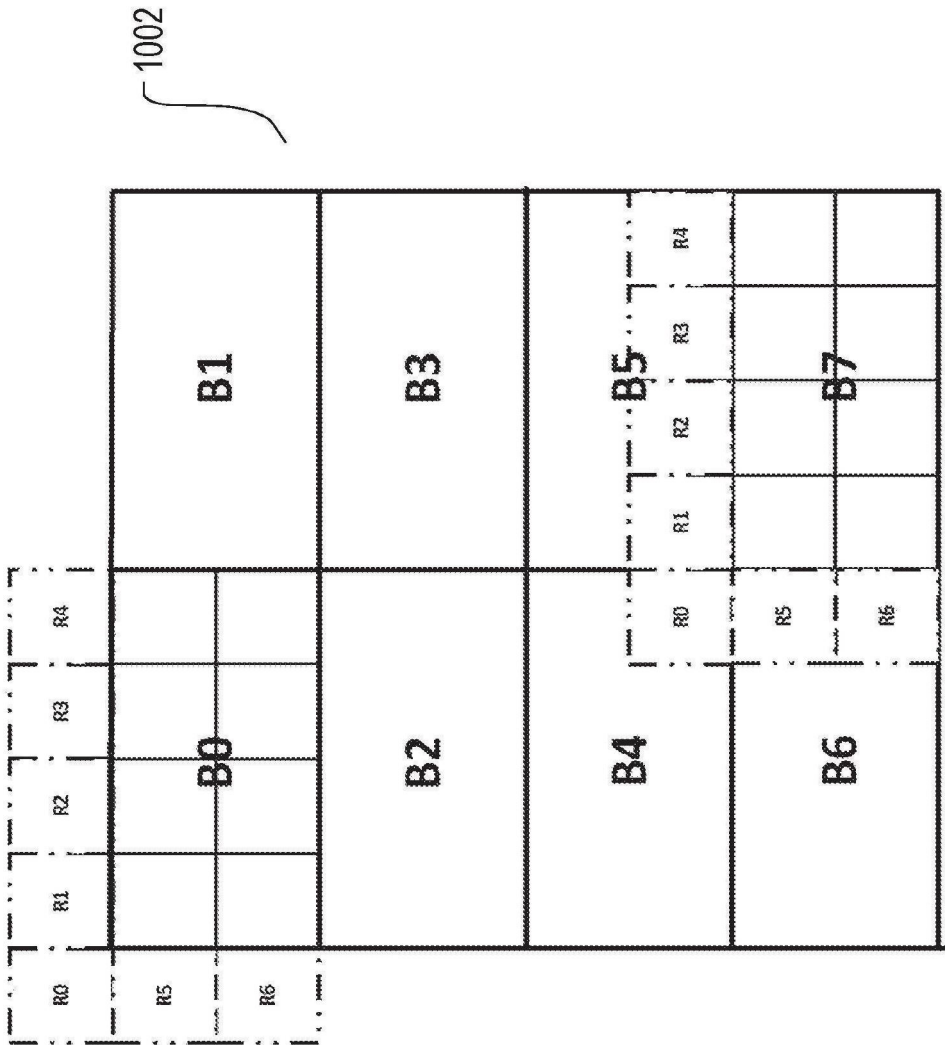


图11

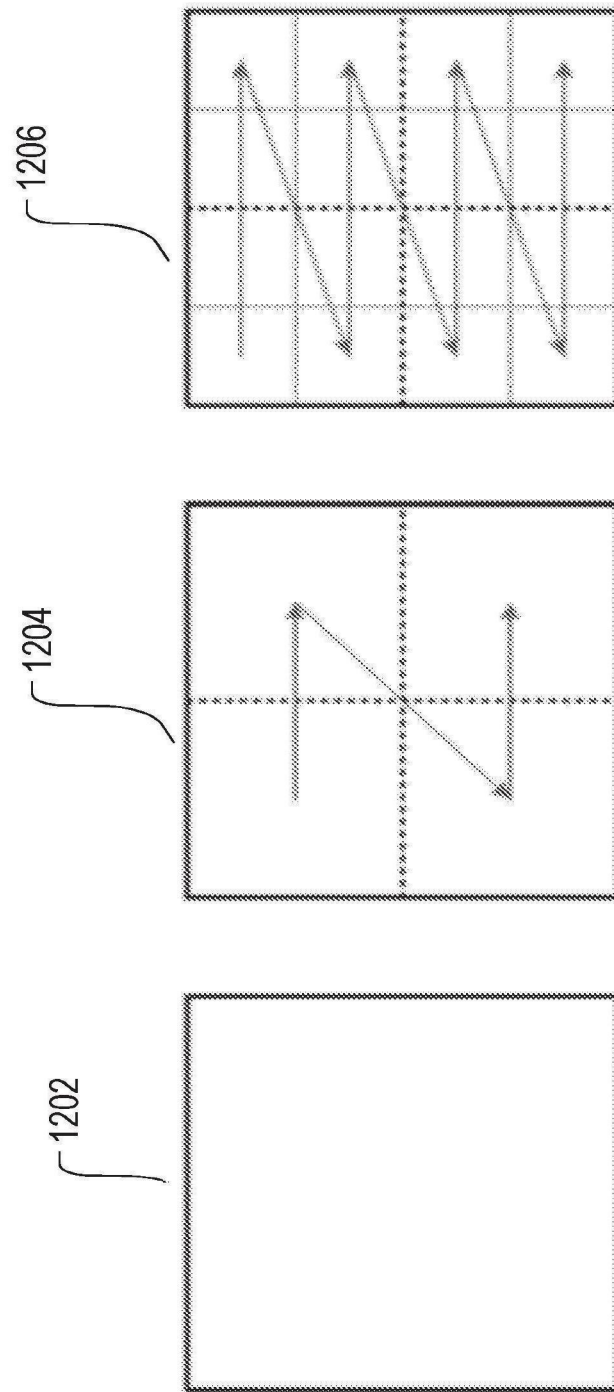


图12

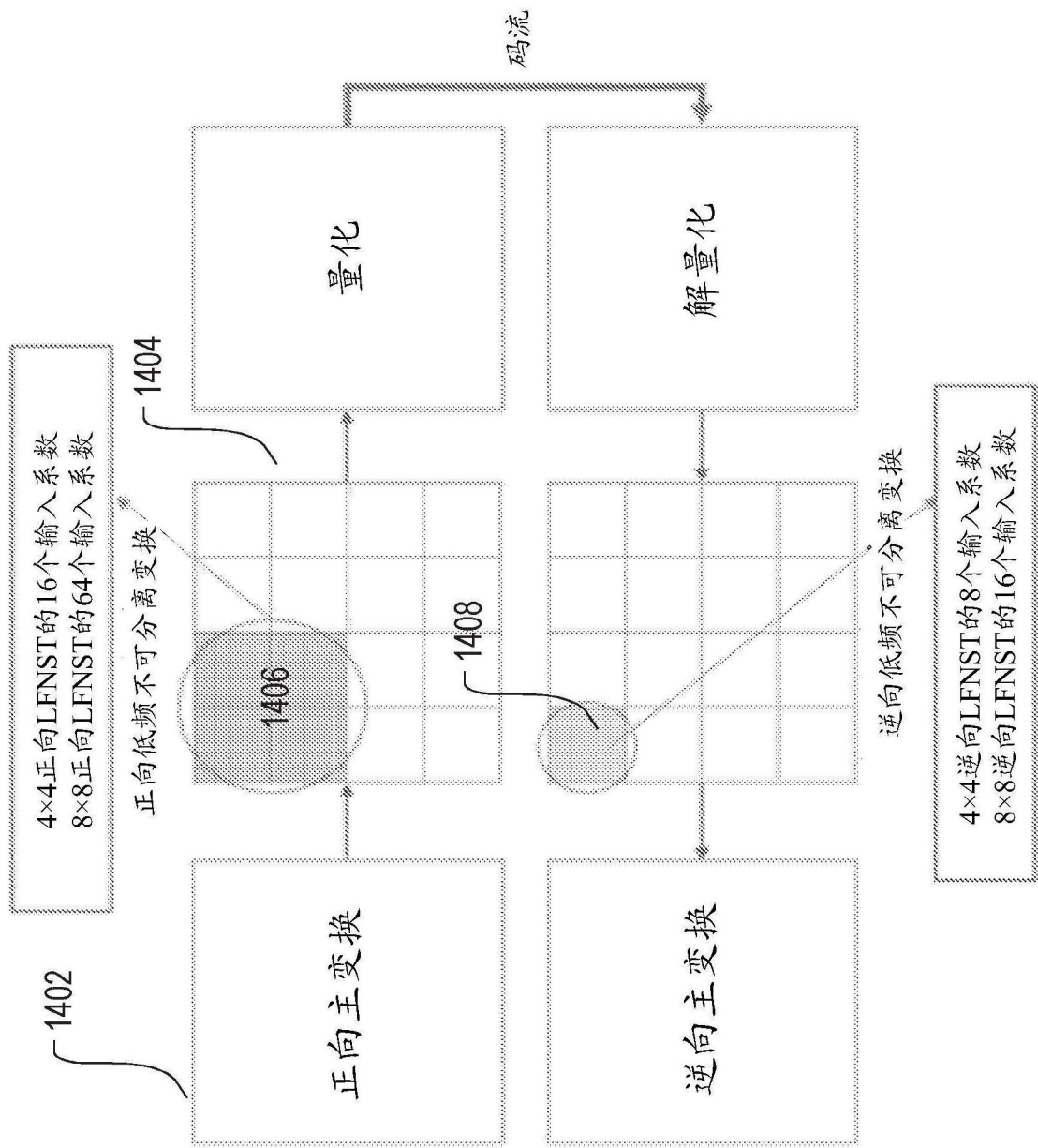


图14

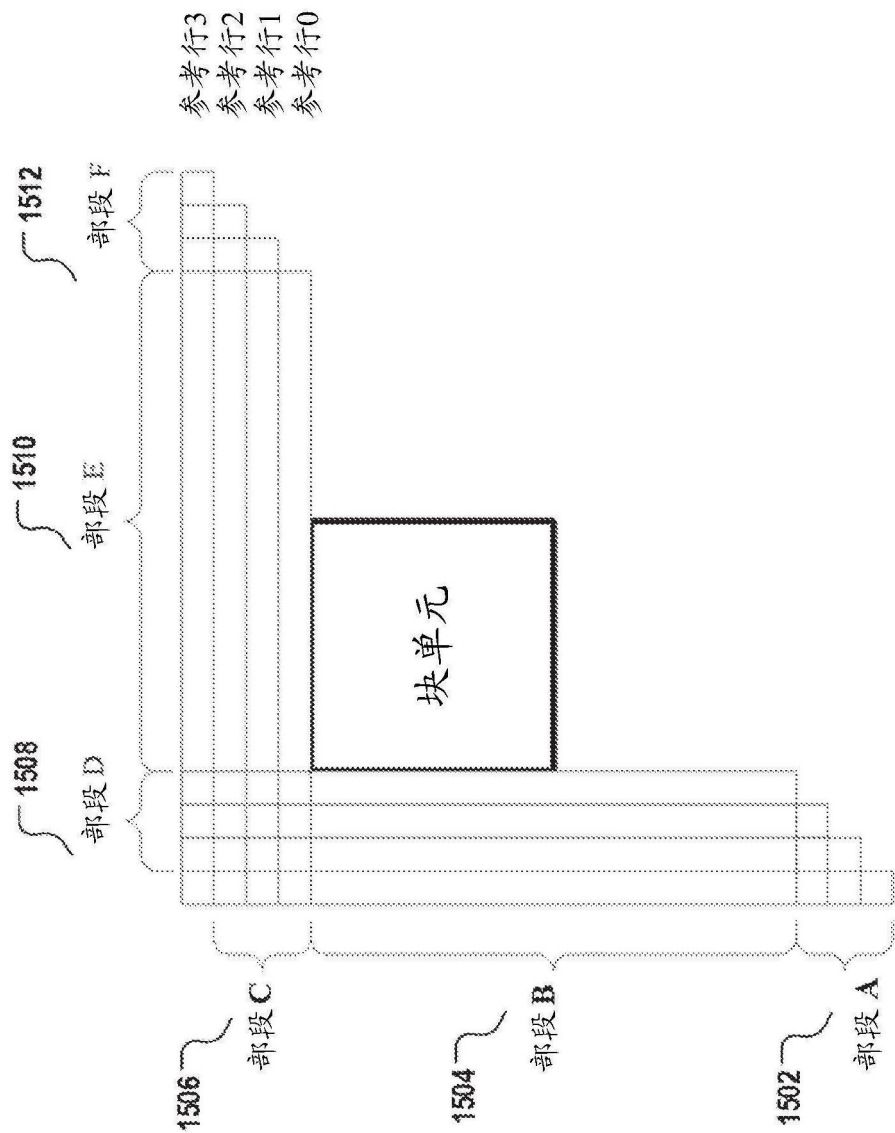


图15

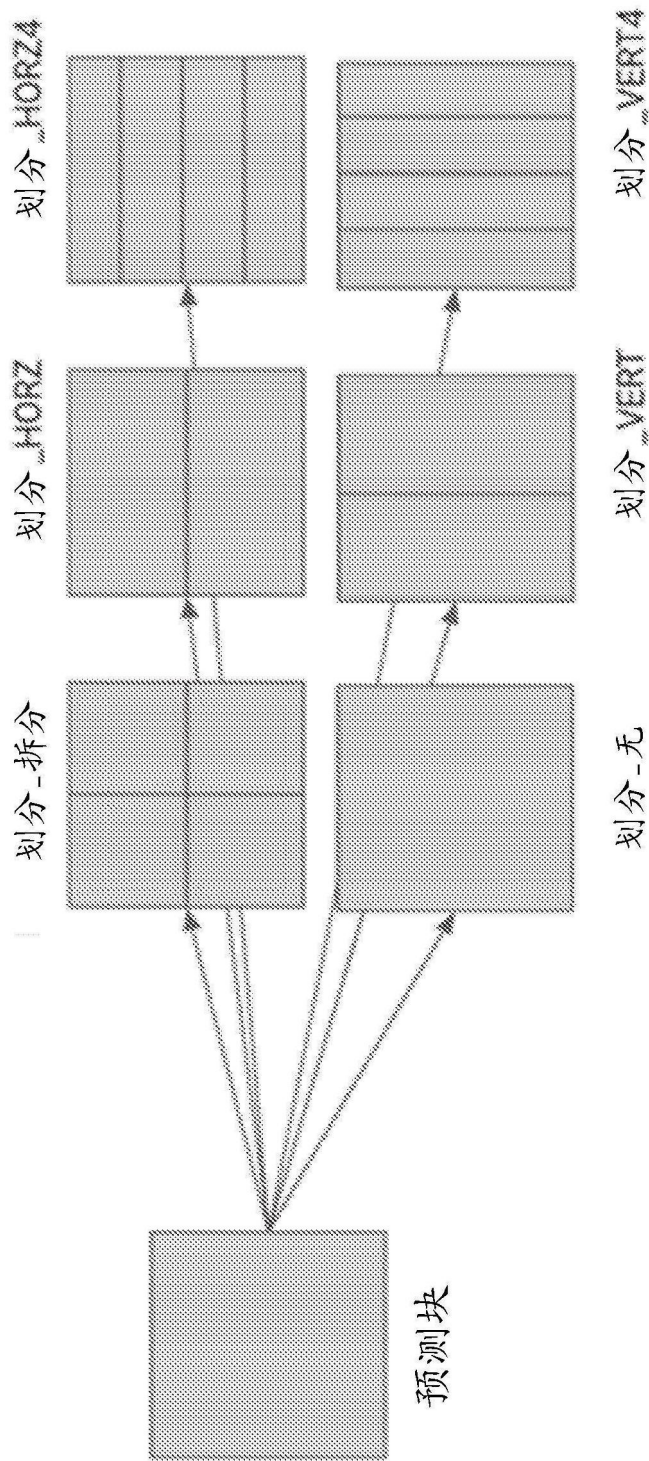


图16

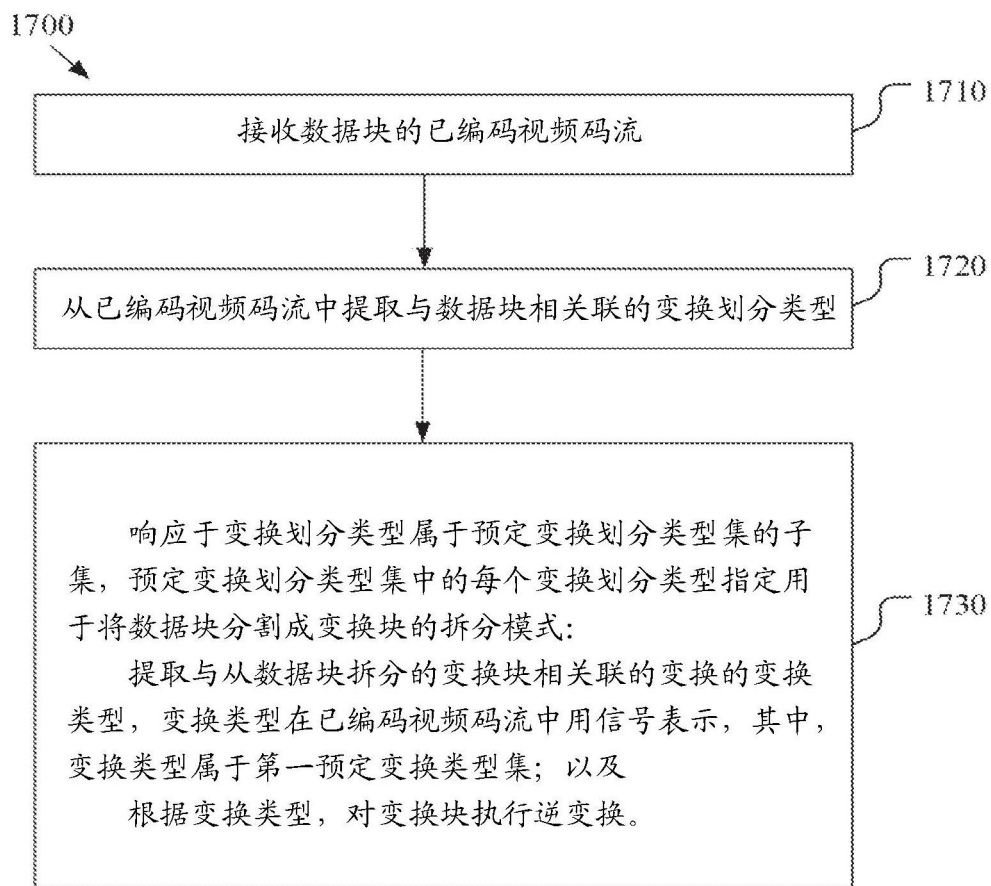


图17

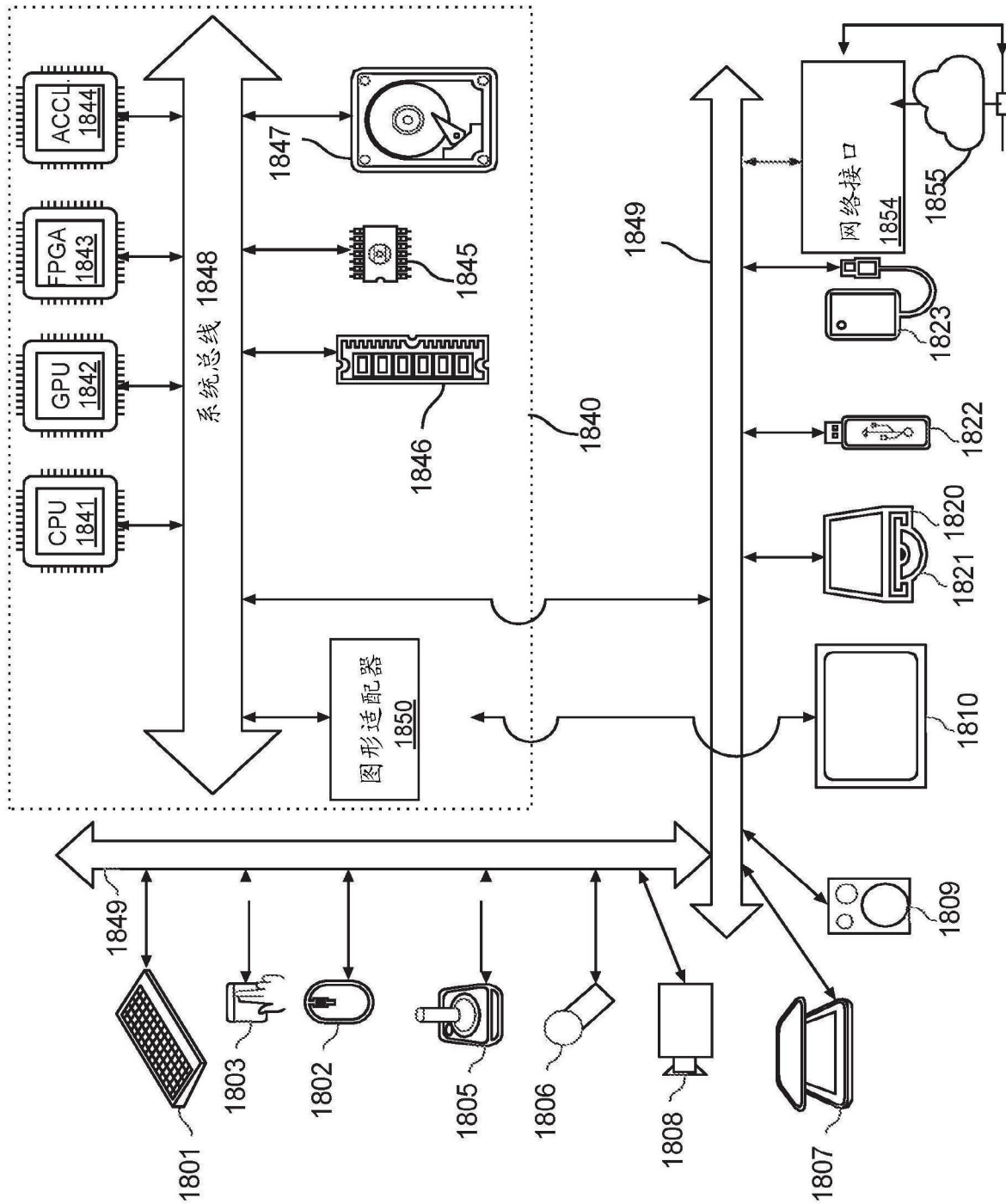


图18

Abstract

Methods, apparatus, and computer readable storage medium for for encoding/decoding video data in a decoder. The method includes receiving a coded video bitstream for a data block; extracting, from the coded video bitstream, a transform partition type associated with the data block; and in response to the transform partition type belonging to a subset of a predefined set of transform partition types each specifying a split pattern for dividing the data block into transform blocks: extracting, as signaled in coded video bitstream, a transform type of a transform associated with a transform block split from the data block, wherein the transform type belongs to a first predefined set of transform types; and performing inverse transform on the transform block according to the transform type.