



- (51) **International Patent Classification:**
H04N 19/52 (2014.01) *H04N 19/70* (2014.01)
- (21) **International Application Number:**
PCT/IB2019/055591
- (22) **International Filing Date:**
01 July 2019 (01.07.2019)
- (25) **Filing Language:** English
- (26) **Publication Language:** English
- (30) **Priority Data:**
PCT/CN2018/093663
29 June 2018 (29.06.2018) CN
PCT/CN2018/093987
02 July 2018 (02.07.2018) CN
- (71) **Applicants:** **BEIJING BYTEDANCE NETWORK TECHNOLOGY CO., LTD.** [CN/CN]; Room B-0035, 2/F, No.3 Building, No.30, Shixing Road, Shijingshan District, Beijing 100041 (CN). **BYTEDANCE INC.** [US/US]; 12655 West Jefferson Boulevard, Sixth Floor, Suite No. 137, Los Angeles, California 90066 (US).
- (72) **Inventors:** **ZHANG, Li**; 12655 West Jefferson Boulevard, Sixth Floor, Suite No. 137, Los Angeles, California 90066 (US). **ZHANG, Kai**; 12655 West Jefferson Boulevard, Sixth Floor, Suite No. 137, Los Angeles, California 90066 (US). **LIU, Hongbin**; Jinritoutiao Post Office, China Satellite Communications Tower, No.63, Zhichun Road, Haidian District, Beijing 100080 (CN). **WANG, Yue**; Jinritoutiao Post Office, China Satellite Communications Tower, No.63, Zhichun Road, Haidian District, Beijing 100080 (CN).
- (74) **Agent:** **LIU, SHEN & ASSOCIATES**; 10th Floor, Building 1, 10 Caihefang Road, Haidian District, Beijing 100080 (CN).
- (81) **Designated States** (*unless otherwise indicated, for every kind of national protection available*): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA,

(54) **Title:** MANAGING MOTION VECTOR PREDICTORS FOR VIDEO CODING

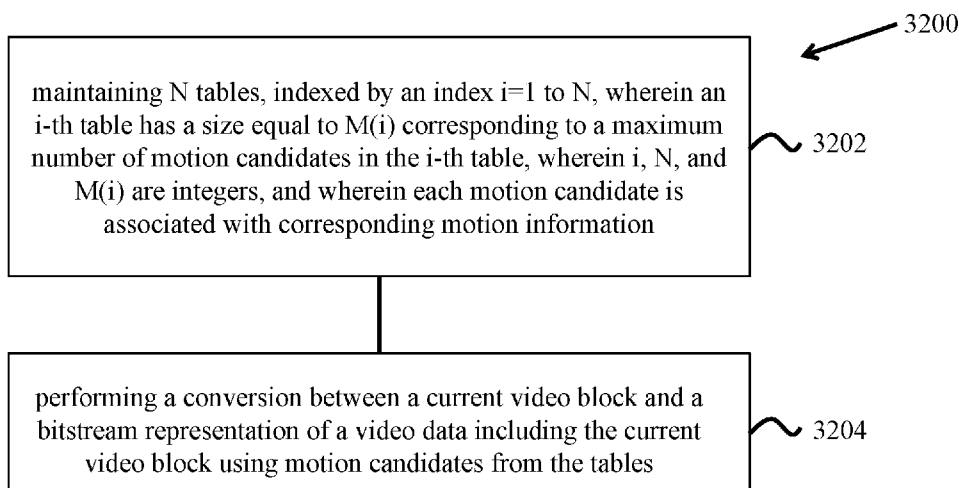


FIG. 32A

(57) **Abstract:** Devices, systems and methods for encoding and decoding digital video using tables containing coding candidates are described. In a representative aspect, a video processing method is provided to comprise: maintaining N tables, indexed by an index $i=1$ to N, wherein an i -th table has a size equal to $M(i)$ corresponding to a maximum number of motion candidates in the i -th table, wherein i , N, and $M(i)$ are integers, and wherein each motion candidate is associated with corresponding motion information; and performing a conversion between a current video block and a bitstream representation of a video data including the current video block using motion candidates from the tables.

SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN,
TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(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))*
- *in black and white; the international application as filed contained color or greyscale and is available for download from PATENTSCOPE*

MANAGING MOTION VECTOR PREDICTORS FOR VIDEO CODING

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] Under the applicable patent law and/or rules pursuant to the Paris Convention, this application is made to timely claim the priority to and benefits of International Patent Application No. PCT/CN2018/093663, filed on June 29, 2018, and International Patent Application No. PCT/CN2018/093987, filed on July 2, 2018. For all purposes under the U.S. law, the entire disclosures of International Patent Application No. PCT/CN2018/093663, and International Patent Application No. PCT/CN2018/093987 are incorporated by reference as part of the disclosure of this application.

TECHNICAL FIELD

[0002] This patent document relates to video coding and decoding techniques, devices and systems.

BACKGROUND

[0003] In spite of the advances in video compression, digital video still accounts for the largest bandwidth use on the internet and other digital communication networks. As the number of connected user devices capable of receiving and displaying video increases, it is expected that the bandwidth demand for digital video usage will continue to grow.

SUMMARY

[0004] Devices, systems and methods related to encoding and decoding digital video using a set of look-up tables (LUTs) containing coding candidates are described. The described methods may be applied to both the existing video coding standards (e.g., High Efficiency Video Coding (HEVC)) and future video coding standards or video codecs.

[0005] In one representative aspect, the disclosed technology may be used to provide a method for video processing method which includes maintaining N tables, indexed by an index $i=1$ to N , wherein an i -th table has a size equal to $M(i)$ corresponding to a maximum number of motion candidates in the i -th table, wherein i , N , and $M(i)$ are integers, and wherein each motion candidate is associated with corresponding motion information; and performing a conversion between a current video block and a bitstream representation of a video data

including the current video block using motion candidates from the tables.

[0006] In another aspect, a video processing method is provided to include: receiving a bitstream representation of a current video block in a video data; and processing the bitstream representation based on N tables, indexed by an index $i=1$ to N, wherein an i -th table has a size equal to $M(i)$ corresponding to a maximum number of motion candidates in the i -th table, wherein i , N, and $M(i)$ are integers, and wherein each motion candidate is associated with corresponding motion information.

[0007] In yet another representative aspect, the above-described method is embodied in the form of processor-executable code and stored in a computer-readable program medium.

[0008] In yet another representative aspect, a device that is configured or operable to perform the above-described method is disclosed. The device may include a processor that is programmed to implement this method.

[0009] In yet another representative aspect, a video decoder apparatus may implement a method as described herein.

[0010] The above and other aspects and features of the disclosed technology are described in greater detail in the drawings, the description and the claims.

BRIEF DESCRIPTION OF THE DRAWINGS

[0011] FIG. 1 shows an example block diagram of a typical High Efficiency Video Coding (HEVC) video encoder and decoder.

[0012] FIG. 2 shows examples of macroblock (MB) partitions in H.264/AVC.

[0013] FIG. 3 shows examples of splitting coding blocks (CBs) into prediction blocks (PBs).

[0014] FIGS. 4A and 4B show an example of the subdivision of a coding tree block (CTB) into CBs and transform blocks (TBs), and the corresponding quadtree, respectively.

[0015] FIGS. 5A and 5B show an example of the subdivisions and a corresponding QTBT (quadtree plus binary tree) for a largest coding unit (LCU).

[0016] FIGS. 6A–6E show examples of partitioning a coding block.

[0017] FIG. 7 shows an example subdivision of a CB based on a QTBT.

- [0018] FIGS. 8A–8I show examples of the partitions of a CB supported the multi-tree type (MTT), which is a generalization of the QTBT.
- [0019] FIG. 9A shows an example of tree-type signaling.
- [0020] FIG. 9B shows an example of constructing a merge candidate list.
- [0021] FIG. 10 shows an example of positions of spatial candidates.
- [0022] FIG. 11 shows an example of candidate pairs subject to a redundancy check of spatial merge candidates.
- [0023] FIGS. 12A and 12B show examples of the position of a second prediction unit (PU) based on the size and shape of the current block.
- [0024] FIG. 13 shows an example of motion vector scaling for temporal merge candidates.
- [0025] FIG. 14 shows an example of candidate positions for temporal merge candidates.
- [0026] FIG. 15 shows an example of generating a combined bi-predictive merge candidate.
- [0027] FIGS. 16A and 16B show examples of a derivation process for motion vector prediction candidates.
- [0028] FIG. 17 shows an example of motion vector scaling for spatial motion vector candidates.
- [0029] FIG. 18 shows an example of motion prediction using the alternative temporal motion vector prediction (ATMVP) algorithm for a coding unit (CU).
- [0030] FIG. 19 shows an example of the identification of a source block and source picture.
- [0031] FIG. 20 shows an example of a coding unit (CU) with sub-blocks and neighboring blocks used by the spatial-temporal motion vector prediction (STMVP) algorithm.
- [0032] FIG. 21 shows an example of bilateral matching in pattern matched motion vector derivation (PMMVD) mode, which is a special merge mode based on the frame-rate up conversion (FRUC) algorithm.
- [0033] FIG. 22 shows an example of template matching in the FRUC algorithm.
- [0034] FIG. 23 shows an example of unilateral motion estimation in the FRUC

algorithm.

[0035] FIG. 24 shows an example of the decoder-side motion vector refinement (DMVR) algorithm based on bilateral template matching.

[0036] FIG. 25 shows an example of neighboring samples used for deriving illumination compensation (IC) parameters.

[0037] FIG. 26 shows an example of neighboring blocks used to derive the spatial merge candidates.

[0038] FIG. 27 shows an example of the proposed 67 intra prediction modes.

[0039] FIG. 28 shows an example of neighboring blocks for Most Probable Mode derivation.

[0040] FIGS. 29A and 29B show the corresponding luma and chroma sub-blocks in an I-slice with the QTBT structure.

[0041] FIGS. 30 and 31 show examples of process flow charts for a LUT (Look-Up Table) based prediction method.

[0042] FIG. 31 shows an example of priority-based insertion of candidates from non-adjacent blocks and LUTs.

[0043] FIG. 32A shows a flowchart of an example method for video processing in accordance with the presently disclosed technology.

[0044] FIG. 32B shows a flowchart of another example method for video processing in accordance with the presently disclosed technology.

[0045] FIG. 33 is a block diagram of an example of a hardware platform for implementing a visual media decoding or a visual media encoding technique described in the present document.

DETAILED DESCRIPTION

[0046] Due to the increasing demand of higher resolution video, video coding methods and techniques are ubiquitous in modern technology. Video codecs typically include an electronic circuit or software that compresses or decompresses digital video, and are continually being improved to provide higher coding efficiency. A video codec converts uncompressed video to a compressed format or vice versa. There are complex relationships between the video quality, the amount of data used to represent the video (determined by the

bit rate), the complexity of the encoding and decoding algorithms, sensitivity to data losses and errors, ease of editing, random access, and end-to-end delay (latency). The compressed format usually conforms to a standard video compression specification, e.g., the High Efficiency Video Coding (HEVC) standard (also known as H.265 or MPEG-H Part 2), the Versatile Video Coding standard to be finalized, or other current and/or future video coding standards.

[0047] Embodiments of the disclosed technology may be applied to existing video coding standards (e.g., HEVC, H.265) and future standards to improve compression performance. Section headings are used in the present document to improve readability of the description and do not in any way limit the discussion or the embodiments (and/or implementations) to the respective sections only.

1. Example embodiments of video coding

[0048] FIG. 1 shows an example block diagram of a typical HEVC video encoder and decoder. An encoding algorithm producing an HEVC compliant bitstream would typically proceed as follows. Each picture is split into block-shaped regions, with the exact block partitioning being conveyed to the decoder. The first picture of a video sequence (and the first picture at each clean random access point into a video sequence) is coded using only intra-picture prediction (that uses some prediction of data spatially from region-to-region within the same picture, but has no dependence on other pictures). For all remaining pictures of a sequence or between random access points, inter-picture temporally predictive coding modes are typically used for most blocks. The encoding process for inter-picture prediction consists of choosing motion data comprising the selected reference picture and motion vector (MV) to be applied for predicting the samples of each block. The encoder and decoder generate identical inter-picture prediction signals by applying motion compensation (MC) using the MV and mode decision data, which are transmitted as side information.

[0049] The residual signal of the intra- or inter-picture prediction, which is the difference between the original block and its prediction, is transformed by a linear spatial transform. The transform coefficients are then scaled, quantized, entropy coded, and transmitted together with the prediction information.

[0050] The encoder duplicates the decoder processing loop (see gray-shaded boxes in FIG. 1) such that both will generate identical predictions for subsequent data.

Therefore, the quantized transform coefficients are constructed by inverse scaling and are then inverse transformed to duplicate the decoded approximation of the residual signal. The residual is then added to the prediction, and the result of that addition may then be fed into one or two loop filters to smooth out artifacts induced by block-wise processing and quantization. The final picture representation (that is a duplicate of the output of the decoder) is stored in a decoded picture buffer to be used for the prediction of subsequent pictures. In general, the order of encoding or decoding processing of pictures often differs from the order in which they arrive from the source; necessitating a distinction between the decoding order (i.e., bitstream order) and the output order (i.e., display order) for a decoder.

[0051] Video material to be encoded by HEVC is generally expected to be input as progressive scan imagery (either due to the source video originating in that format or resulting from deinterlacing prior to encoding). No explicit coding features are present in the HEVC design to support the use of interlaced scanning, as interlaced scanning is no longer used for displays and is becoming substantially less common for distribution. However, a metadata syntax has been provided in HEVC to allow an encoder to indicate that interlace-scanned video has been sent by coding each field (i.e., the even or odd numbered lines of each video frame) of interlaced video as a separate picture or that it has been sent by coding each interlaced frame as an HEVC coded picture. This provides an efficient method of coding interlaced video without burdening decoders with a need to support a special decoding process for it.

1.1. Examples of partition tree structures in H.264/AVC

[0052] The core of the coding layer in previous standards was the macroblock, containing a 16×16 block of luma samples and, in the usual case of 4:2:0 color sampling, two corresponding 8×8 blocks of chroma samples.

[0053] An intra-coded block uses spatial prediction to exploit spatial correlation among pixels. Two partitions are defined: 16×16 and 4×4 .

[0054] An inter-coded block uses temporal prediction, instead of spatial prediction, by estimating motion among pictures. Motion can be estimated independently for either 16×16 macroblock or any of its sub-macroblock partitions: 16×8 , 8×16 , 8×8 , 8×4 , 4×8 , 4×4 , as shown in FIG. 2. Only one motion vector (MV) per sub-macroblock partition is allowed.

1.2 Examples of partition tree structures in HEVC

[0055] In HEVC, a coding tree unit (CTU) is split into coding units (CUs) by using a quadtree structure denoted as coding tree to adapt to various local characteristics. The decision whether to code a picture area using inter-picture (temporal) or intra-picture (spatial) prediction is made at the CU level. Each CU can be further split into one, two or four prediction units (PUs) according to the PU splitting type. Inside one PU, the same prediction process is applied and the relevant information is transmitted to the decoder on a PU basis. After obtaining the residual block by applying the prediction process based on the PU splitting type, a CU can be partitioned into transform units (TUs) according to another quadtree structure similar to the coding tree for the CU. One of key feature of the HEVC structure is that it has the multiple partition conceptions including CU, PU, and TU.

[0056] Certain features involved in hybrid video coding using HEVC include:

[0057] (1) Coding tree units (CTUs) and coding tree block (CTB) structure: The analogous structure in HEVC is the coding tree unit (CTU), which has a size selected by the encoder and can be larger than a traditional macroblock. The CTU consists of a luma CTB and the corresponding chroma CTBs and syntax elements. The size $L \times L$ of a luma CTB can be chosen as $L = 16, 32$, or 64 samples, with the larger sizes typically enabling better compression. HEVC then supports a partitioning of the CTBs into smaller blocks using a tree structure and quadtree-like signaling.

[0058] (2) Coding units (CUs) and coding blocks (CBs): The quadtree syntax of the CTU specifies the size and positions of its luma and chroma CBs. The root of the quadtree is associated with the CTU. Hence, the size of the luma CTB is the largest supported size for a luma CB. The splitting of a CTU into luma and chroma CBs is signaled jointly. One luma CB and ordinarily two chroma CBs, together with associated syntax, form a coding unit (CU). A CTB may contain only one CU or may be split to form multiple CUs, and each CU has an associated partitioning into prediction units (PUs) and a tree of transform units (TUs).

[0059] (3) Prediction units and prediction blocks (PBs): The decision whether to code a picture area using inter picture or intra picture prediction is made at the CU level. A PU partitioning structure has its root at the CU level. Depending on the basic prediction-type decision, the luma and chroma CBs can then be further split in size and predicted from luma and chroma prediction blocks (PBs). HEVC supports variable PB sizes from 64×64 down to

4×4 samples. FIG. 3 shows examples of allowed PBs for an M×M CU.

[0060] (4) Transform units (Tus) and transform blocks: The prediction residual is coded using block transforms. A TU tree structure has its root at the CU level. The luma CB residual may be identical to the luma transform block (TB) or may be further split into smaller luma TBs. The same applies to the chroma TBs. Integer basis functions similar to those of a discrete cosine transform (DCT) are defined for the square TB sizes 4×4, 8×8, 16×16, and 32×32. For the 4×4 transform of luma intra picture prediction residuals, an integer transform derived from a form of discrete sine transform (DST) is alternatively specified.

1.2.1. Examples of tree-structured partitioning into TBs and TUs

[0061] For residual coding, a CB can be recursively partitioned into transform blocks (TBs). The partitioning is signaled by a residual quadtree. Only square CB and TB partitioning is specified, where a block can be recursively split into quadrants, as illustrated in FIG. 4. For a given luma CB of size M×M, a flag signals whether it is split into four blocks of size M/2×M/2. If further splitting is possible, as signaled by a maximum depth of the residual quadtree indicated in the sequence parameter set (SPS), each quadrant is assigned a flag that indicates whether it is split into four quadrants. The leaf node blocks resulting from the residual quadtree are the transform blocks that are further processed by transform coding. The encoder indicates the maximum and minimum luma TB sizes that it will use. Splitting is implicit when the CB size is larger than the maximum TB size. Not splitting is implicit when splitting would result in a luma TB size smaller than the indicated minimum. The chroma TB size is half the luma TB size in each dimension, except when the luma TB size is 4×4, in which case a single 4×4 chroma TB is used for the region covered by four 4×4 luma TBs. In the case of intra-picture-predicted CUs, the decoded samples of the nearest-neighboring TBs (within or outside the CB) are used as reference data for intra picture prediction.

[0062] In contrast to previous standards, the HEVC design allows a TB to span across multiple PBs for inter-picture predicted CUs to maximize the potential coding efficiency benefits of the quadtree-structured TB partitioning.

1.2.2. Parent and child nodes

[0063] A CTB is divided according to a quad-tree structure, the nodes of which are coding units. The plurality of nodes in a quad-tree structure includes leaf nodes and non-leaf nodes.

The leaf nodes have no child nodes in the tree structure (i.e., the leaf nodes are not further split). The, non-leaf nodes include a root node of the tree structure. The root node corresponds to an initial video block of the video data (e.g., a CTB). For each respective non-root node of the plurality of nodes, the respective non-root node corresponds to a video block that is a sub-block of a video block corresponding to a parent node in the tree structure of the respective non-root node. Each respective non-leaf node of the plurality of non-leaf nodes has one or more child nodes in the tree structure.

1.3. Examples of quadtree plus binary tree block structures with larger CTUs in JEM

[0064] In some embodiments, future video coding technologies are explored using a reference software known as the Joint Exploration Model (JEM). In addition to binary tree structures, JEM describes quadtree plus binary tree (QTBT) and ternary tree (TT) structures.

1.3.1. Examples of the QTBT block partitioning structure

[0065] In contrast to HEVC, the QTBT structure removes the concepts of multiple partition types, i.e. it removes the separation of the CU, PU and TU concepts, and supports more flexibility for CU partition shapes. In the QTBT block structure, a CU can have either a square or rectangular shape. As shown in FIG. 5A, a coding tree unit (CTU) is first partitioned by a quadtree structure. The quadtree leaf nodes are further partitioned by a binary tree structure. There are two splitting types, symmetric horizontal splitting and symmetric vertical splitting, in the binary tree splitting. The binary tree leaf nodes are called coding units (CUs), and that segmentation is used for prediction and transform processing without any further partitioning. This means that the CU, PU and TU have the same block size in the QTBT coding block structure. In the JEM, a CU sometimes consists of coding blocks (CBs) of different colour components, e.g. one CU contains one luma CB and two chroma CBs in the case of P and B slices of the 4:2:0 chroma format and sometimes consists of a CB of a single component, e.g., one CU contains only one luma CB or just two chroma CBs in the case of I slices.

[0066] The following parameters are defined for the QTBT partitioning scheme:

[0067] -- CTU size: the root node size of a quadtree, the same concept as in HEVC

[0068] -- *MinQTSIZE*: the minimally allowed quadtree leaf node size

[0069] -- *MaxBTSIZE*: the maximally allowed binary tree root node size

[0070] -- *MaxBTDepth*: the maximally allowed binary tree depth

[0071] -- *MinBTSIZE*: the minimally allowed binary tree leaf node size

[0072] In one example of the QTBT partitioning structure, the CTU size is set as 128×128 luma samples with two corresponding 64×64 blocks of chroma samples, the *MinQTSIZE* is set as 16×16 , the *MaxBTSIZE* is set as 64×64 , the *MinBTSIZE* (for both width and height) is set as 4×4 , and the *MaxBTDepth* is set as 4. The quadtree partitioning is applied to the CTU first to generate quadtree leaf nodes. The quadtree leaf nodes may have a size from 16×16 (i.e., the *MinQTSIZE*) to 128×128 (i.e., the CTU size). If the quadtree leaf node is 128×128 , it will not be further split by the binary tree since the size exceeds the *MaxBTSIZE* (i.e., 64×64).

Otherwise, the quadtree leaf node could be further partitioned by the binary tree. Therefore, the quadtree leaf node is also the root node for the binary tree and it has the binary tree depth as 0. When the binary tree depth reaches *MaxBTDepth* (i.e., 4), no further splitting is considered. When the binary tree node has width equal to *MinBTSIZE* (i.e., 4), no further horizontal splitting is considered. Similarly, when the binary tree node has height equal to *MinBTSIZE*, no further vertical splitting is considered. The leaf nodes of the binary tree are further processed by prediction and transform processing without any further partitioning. In the JEM, the maximum CTU size is 256×256 luma samples.

[0073] FIG. 5A shows an example of block partitioning by using QTBT, and FIG. 5B shows the corresponding tree representation. The solid lines indicate quadtree splitting and dotted lines indicate binary tree splitting. In each splitting (i.e., non-leaf) node of the binary tree, one flag is signalled to indicate which splitting type (i.e., horizontal or vertical) is used, where 0 indicates horizontal splitting and 1 indicates vertical splitting. For the quadtree splitting, there is no need to indicate the splitting type since quadtree splitting always splits a block both horizontally and vertically to produce 4 sub-blocks with an equal size.

[0074] In addition, the QTBT scheme supports the ability for the luma and chroma to have a separate QTBT structure. Currently, for P and B slices, the luma and chroma CTBs in one CTU share the same QTBT structure. However, for I slices, the luma CTB is partitioned into CUs by a QTBT structure, and the chroma CTBs are partitioned into chroma CUs by another QTBT structure. This means that a CU in an I slice consists of a coding block of the luma component or coding blocks of two chroma components, and a CU in a P or B slice consists

of coding blocks of all three colour components.

[0075] In HEVC, inter prediction for small blocks is restricted to reduce the memory access of motion compensation, such that bi-prediction is not supported for 4×8 and 8×4 blocks, and inter prediction is not supported for 4×4 blocks. In the QTBT of the JEM, these restrictions are removed.

1.4. Ternary-tree (TT) for Versatile Video Coding (VVC)

[0076] FIG. 6A shows an example of quad-tree (QT) partitioning, and FIGS. 6B and 6C show examples of the vertical and horizontal binary-tree (BT) partitioning, respectively. In some embodiments, and in addition to quad-trees and binary-trees, ternary tree (TT) partitions, e.g., horizontal and vertical center-side ternary-trees (as shown in FIGS. 6D and 6E) are supported.

[0077] In some implementations, two levels of trees are supported: region tree (quad-tree) and prediction tree (binary-tree or ternary-tree). A CTU is firstly partitioned by region tree (RT). A RT leaf may be further split with prediction tree (PT). A PT leaf may also be further split with PT until max PT depth is reached. A PT leaf is the basic coding unit. It is still called CU for convenience. A CU cannot be further split. Prediction and transform are both applied on CU in the same way as JEM. The whole partition structure is named ‘multiple-type-tree’.

1.5. Examples of partitioning structures in alternate video coding technologies

[0078] In some embodiments, a tree structure called a Multi-Tree Type (MTT), which is a generalization of the QTBT, is supported. In QTBT, as shown in FIG. 7, a Coding Tree Unit (CTU) is firstly partitioned by a quad-tree structure. The quad-tree leaf nodes are further partitioned by a binary-tree structure.

[0079] The structure of the MTT constitutes of two types of tree nodes: Region Tree (RT) and Prediction Tree (PT), supporting nine types of partitions, as shown in FIG. 8. A region tree can recursively split a CTU into square blocks down to a 4×4 size region tree leaf node. At each node in a region tree, a prediction tree can be formed from one of three tree types: Binary Tree, Ternary Tree, and Asymmetric Binary Tree. In a PT split, it is prohibited to have a quadtree partition in branches of the prediction tree. As in JEM, the luma tree and the chroma tree are separated in I slices. The signaling methods for RT and PT are illustrated in FIG. 9A.

2 Examples of inter-prediction in HEVC/H.265

[0080] Video coding standards have significantly improved over the years, and now provide, in part, high coding efficiency and support for higher resolutions. Recent standards such as HEVC and H.265 are based on the hybrid video coding structure wherein temporal prediction plus transform coding are utilized.

2.1 Examples of prediction modes

[0081] Each inter-predicted PU (prediction unit) has motion parameters for one or two reference picture lists. In some embodiments, motion parameters include a motion vector and a reference picture index. In other embodiments, the usage of one of the two reference picture lists may also be signaled using *inter_pred_idc*. In yet other embodiments, motion vectors may be explicitly coded as deltas relative to predictors.

[0082] When a CU is coded with skip mode, one PU is associated with the CU, and there are no significant residual coefficients, no coded motion vector delta or reference picture index. A merge mode is specified whereby the motion parameters for the current PU are obtained from neighboring PUs, including spatial and temporal candidates. The merge mode can be applied to any inter-predicted PU, not only for skip mode. The alternative to merge mode is the explicit transmission of motion parameters, where motion vector, corresponding reference picture index for each reference picture list and reference picture list usage are signaled explicitly per each PU.

[0083] When signaling indicates that one of the two reference picture lists is to be used, the PU is produced from one block of samples. This is referred to as ‘uni-prediction’. Uni-prediction is available both for P-slices and B-slices.

[0084] When signaling indicates that both of the reference picture lists are to be used, the PU is produced from two blocks of samples. This is referred to as ‘bi-prediction’. Bi-prediction is available for B-slices only.

2.1.1 Embodiments of constructing candidates for merge mode

[0085] When a PU is predicted using merge mode, an index pointing to an entry in the *merge candidates list* is parsed from the bitstream and used to retrieve the motion information. The construction of this list can be summarized according to the following sequence of steps:

[0086] Step 1: Initial candidates derivation

[0087] Step 1.1: Spatial candidates derivation

[0088] Step 1.2: Redundancy check for spatial candidates

[0089] Step 1.3: Temporal candidates derivation

[0090] Step 2: Additional candidates insertion

[0091] Step 2.1: Creation of bi-predictive candidates

[0092] Step 2.2: Insertion of zero motion candidates

[0093] FIG. 9B shows an example of constructing a merge candidate list based on the sequence of steps summarized above. For spatial merge candidate derivation, a maximum of four merge candidates are selected among candidates that are located in five different positions. For temporal merge candidate derivation, a maximum of one merge candidate is selected among two candidates. Since constant number of candidates for each PU is assumed at decoder, additional candidates are generated when the number of candidates does not reach to maximum number of merge candidate (MaxNumMergeCand) which is signalled in slice header. Since the number of candidates is constant, index of best merge candidate is encoded using truncated unary binarization (TU). If the size of CU is equal to 8, all the PUs of the current CU share a single merge candidate list, which is identical to the merge candidate list of the $2N \times 2N$ prediction unit.

2.1.2 Constructing spatial merge candidates

[0094] In the derivation of spatial merge candidates, a maximum of four merge candidates are selected among candidates located in the positions depicted in FIG. 10. The order of derivation is A_1 , B_1 , B_0 , A_0 and B_2 . Position B_2 is considered only when any PU of position A_1 , B_1 , B_0 , A_0 is not available (e.g. because it belongs to another slice or tile) or is intra coded. After candidate at position A_1 is added, the addition of the remaining candidates is subject to a redundancy check which ensures that candidates with same motion information are excluded from the list so that coding efficiency is improved.

[0095] To reduce computational complexity, not all possible candidate pairs are considered in the mentioned redundancy check. Instead only the pairs linked with an arrow in FIG. 11 are considered and a candidate is only added to the list if the corresponding candidate used for redundancy check has not the same motion information. Another source of duplicate motion information is the “*second PU*” associated with partitions different from $2N \times 2N$. As an example, FIG. 12A and 12B depict the second PU for the case of $N \times 2N$ and $2N \times N$, respectively. When the current PU is partitioned as $N \times 2N$, candidate at position A_1 is not

considered for list construction. In some embodiments, adding this candidate may lead to two prediction units having the same motion information, which is redundant to just have one PU in a coding unit. Similarly, position B₁ is not considered when the current PU is partitioned as 2N×N.

2.1.3 Constructing temporal merge candidates

[0096] In this step, only one candidate is added to the list. Particularly, in the derivation of this temporal merge candidate, a scaled motion vector is derived based on co-located PU belonging to the picture which has the smallest POC difference with current picture within the given reference picture list. The reference picture list to be used for derivation of the co-located PU is explicitly signaled in the slice header.

[0097] FIG. 13 shows an example of the derivation of the scaled motion vector for a temporal merge candidate (as the dotted line), which is scaled from the motion vector of the co-located PU using the POC distances, t_b and t_d , where t_b is defined to be the POC difference between the reference picture of the current picture and the current picture and t_d is defined to be the POC difference between the reference picture of the co-located picture and the co-located picture. The reference picture index of temporal merge candidate is set equal to zero. For a B-slice, two motion vectors, one is for reference picture list 0 and the other is for reference picture list 1, are obtained and combined to make the bi-predictive merge candidate.

[0098] In the co-located PU (Y) belonging to the reference frame, the position for the temporal candidate is selected between candidates C₀ and C₁, as depicted in FIG. 14. If PU at position C₀ is not available, is intra coded, or is outside of the current CTU, position C₁ is used. Otherwise, position C₀ is used in the derivation of the temporal merge candidate.

2.1.4 Constructing additional types of merge candidates

[0099] Besides spatio-temporal merge candidates, there are two additional types of merge candidates: combined bi-predictive merge candidate and zero merge candidate. Combined bi-predictive merge candidates are generated by utilizing spatio-temporal merge candidates. Combined bi-predictive merge candidate is used for B-Slice only. The combined bi-predictive candidates are generated by combining the first reference picture list motion parameters of an initial candidate with the second reference picture list motion parameters of another. If these two tuples provide different motion hypotheses, they will form a new bi-

predictive candidate.

[00100] FIG. 15 shows an example of this process, wherein two candidates in the original list (1510, on the left), which have mvL0 and refIdxL0 or mvL1 and refIdxL1, are used to create a combined bi-predictive merge candidate added to the final list (1520, on the right).

[00101] Zero motion candidates are inserted to fill the remaining entries in the merge candidates list and therefore hit the MaxNumMergeCand capacity. These candidates have zero spatial displacement and a reference picture index which starts from zero and increases every time a new zero motion candidate is added to the list. The number of reference frames used by these candidates is one and two for uni- and bi-directional prediction, respectively. In some embodiments, no redundancy check is performed on these candidates.

2.1.5 Examples of motion estimation regions for parallel processing

[00102] To speed up the encoding process, motion estimation can be performed in parallel whereby the motion vectors for all prediction units inside a given region are derived simultaneously. The derivation of merge candidates from spatial neighborhood may interfere with parallel processing as one prediction unit cannot derive the motion parameters from an adjacent PU until its associated motion estimation is completed. To mitigate the trade-off between coding efficiency and processing latency, a motion estimation region (MER) may be defined. The size of the MER may be signaled in the picture parameter set (PPS) using the “log2_parallel_merge_level_minus2” syntax element. When a MER is defined, merge candidates falling in the same region are marked as unavailable and therefore not considered in the list construction.

[00103] The picture parameter set (PPS) raw byte sequence payload (RBSP) syntax is shown in Table 1, where **log2_parallel_merge_level_minus2** plus 2 specifies the value of the variable Log2ParMrgLevel, which is used in the derivation process for luma motion vectors for merge mode and the derivation process for spatial merging candidates as specified in an existing video coding standard. The value of log2_parallel_merge_level_minus2 shall be in the range of 0 to CtbLog2SizeY – 2, inclusive.

[00104] The variable Log2ParMrgLevel is derived as follows:

[00105]
$$\text{Log2ParMrgLevel} = \text{log2_parallel_merge_level_minus2} + 2$$

[00106] Note that the value of Log2ParMrgLevel indicates the built-in capability of parallel derivation of the merging candidate lists. For example, when Log2ParMrgLevel is equal to 6, the merging candidate lists for all the prediction units (PUs) and coding units (CUs) contained in a 64x64 block can be derived in parallel.

Table 1: General picture parameter set RBSP syntax

pic_parameter_set_rbsp() {	Descriptor
pps_pic_parameter_set_id	ue(v)
pps_seq_parameter_set_id	ue(v)
dependent_slice_segments_enabled_flag	u(1)
...	
pps_scaling_list_data_present_flag	u(1)
if(pps_scaling_list_data_present_flag)	
scaling_list_data()	
lists_modification_present_flag	u(1)
log2_parallel_merge_level_minus2	ue(v)
slice_segment_header_extension_present_flag	u(1)
pps_extension_present_flag	u(1)
...	
rbp_trailing_bits()	
}	

2.2 Embodiments of motion vector prediction in AMVP mode

[00107] Motion vector prediction exploits spatio-temporal correlation of motion vector with neighboring PUs, which is used for explicit transmission of motion parameters. It constructs a motion vector candidate list by firstly checking availability of left, above temporally neighboring PU positions, removing redundant candidates and adding zero vector to make the candidate list to be constant length. Then, the encoder can select the best predictor from the candidate list and transmit the corresponding index indicating the chosen candidate. Similarly with merge index signaling, the index of the best motion vector candidate is encoded using truncated unary.

2.2.1 Examples of constructing motion vector prediction candidates

[00108] FIG. 16 summarizes derivation process for motion vector prediction candidate, and may be implemented for each reference picture list with refidx as an input.

[00109] In motion vector prediction, two types of motion vector candidates are

considered: spatial motion vector candidate and temporal motion vector candidate. For spatial motion vector candidate derivation, two motion vector candidates are eventually derived based on motion vectors of each PU located in five different positions as previously shown in FIG. 10.

[00110] For temporal motion vector candidate derivation, one motion vector candidate is selected from two candidates, which are derived based on two different co-located positions. After the first list of spatio-temporal candidates is made, duplicated motion vector candidates in the list are removed. If the number of potential candidates is larger than two, motion vector candidates whose reference picture index within the associated reference picture list is larger than 1 are removed from the list. If the number of spatio-temporal motion vector candidates is smaller than two, additional zero motion vector candidates is added to the list.

2.2.2 Constructing spatial motion vector candidates

[00111] In the derivation of spatial motion vector candidates, a maximum of two candidates are considered among five potential candidates, which are derived from PUs located in positions as previously shown in FIG. 10, those positions being the same as those of motion merge. The order of derivation for the left side of the current PU is defined as A_0 , A_1 , and scaled A_0 , scaled A_1 . The order of derivation for the above side of the current PU is defined as B_0 , B_1 , B_2 , scaled B_0 , scaled B_1 , scaled B_2 . For each side there are therefore four cases that can be used as motion vector candidate, with two cases not required to use spatial scaling, and two cases where spatial scaling is used. The four different cases are summarized as follows:

[00112] -- No spatial scaling

[00113] (1) Same reference picture list, and same reference picture index
(same POC)

[00114] (2) Different reference picture list, but same reference picture
(same POC)

[00115] -- Spatial scaling

[00116] (3) Same reference picture list, but different reference picture
(different POC)

[00117] (4) Different reference picture list, and different reference picture

(different POC)

[00118] The no-spatial-scaling cases are checked first followed by the cases that allow spatial scaling. Spatial scaling is considered when the POC is different between the reference picture of the neighbouring PU and that of the current PU regardless of reference picture list. If all PUs of left candidates are not available or are intra coded, scaling for the above motion vector is allowed to help parallel derivation of left and above MV candidates. Otherwise, spatial scaling is not allowed for the above motion vector.

[00119] As shown in the example in FIG. 17, for the spatial scaling case, the motion vector of the neighbouring PU is scaled in a similar manner as for temporal scaling. One difference is that the reference picture list and index of current PU is given as input; the actual scaling process is the same as that of temporal scaling.

2.2.3 Constructing temporal motion vector candidates

[00120] Apart from the reference picture index derivation, all processes for the derivation of temporal merge candidates are the same as for the derivation of spatial motion vector candidates (as shown in the example in FIG. 14). In some embodiments, the reference picture index is signaled to the decoder.

2.2.4 Signaling of Merge/AMVP information

[00121] For the AMVP mode, four parts may be signalled in the bitstream, e.g., prediction direction, reference index, MVD and mv predictor candidate index, which are described in the context of the syntax shown in Table 2-4. While for the merge mode, only a merge index may need to be signalled.

Table 2: General slice segment header syntax

slice_segment_header() {	Descriptor
...	
if(slice_type == P slice_type == B) {	
if((weighted_pred_flag && slice_type == P) (weighted_bipred_flag && slice_type == B))	
pred_weight_table()	
five_minus_max_num_merge_cand	ue(v)
if(motion_vector_resolution_control_idc == 2)	
use_integer_mv_flag	u(1)
}	
...	

Table 3: Prediction unit syntax

prediction_unit(x0, y0, nPbW, nPbH) {	Descriptor
if(cu_skip_flag[x0][y0]) {	
if(MaxNumMergeCand > 1)	
merge_idx [x0][y0]	ae(v)
} else { /* MODE_INTER */	
merge_flag [x0][y0]	ae(v)
if(merge_flag[x0][y0]) {	
if(MaxNumMergeCand > 1)	
merge_idx [x0][y0]	ae(v)
} else {	
if(slice_type == B)	
inter_pred_idc [x0][y0]	ae(v)
if(inter_pred_idc[x0][y0] != PRED_L1) {	
if(num_ref_idx_l0_active_minus1 > 0)	
ref_idx_l0 [x0][y0]	ae(v)
mvd_coding(x0, y0, 0)	
mvp_l0_flag [x0][y0]	ae(v)
}	
if(inter_pred_idc[x0][y0] != PRED_L0) {	
if(num_ref_idx_l1_active_minus1 > 0)	
ref_idx_l1 [x0][y0]	ae(v)
if(mvd_l1_zero_flag && inter_pred_idc[x0][y0] == PRED_BI) {	
MvdL1[x0][y0][0] = 0	
MvdL1[x0][y0][1] = 0	
} else	
mvd_coding(x0, y0, 1)	
mvp_l1_flag [x0][y0]	ae(v)
}	
}	
}	
}	

Table 4: Motion vector difference syntax

mvd_coding(x0, y0, refList) {	Descriptor
abs_mvd_greater0_flag[0]	ae(v)
abs_mvd_greater0_flag[1]	ae(v)
if(abs_mvd_greater0_flag[0])	
abs_mvd_greater1_flag[0]	ae(v)
if(abs_mvd_greater0_flag[1])	
abs_mvd_greater1_flag[1]	ae(v)
if(abs_mvd_greater0_flag[0]) {	
if(abs_mvd_greater1_flag[0])	
abs_mvd_minus2[0]	ae(v)
mvd_sign_flag[0]	ae(v)
}	
if(abs_mvd_greater0_flag[1]) {	
if(abs_mvd_greater1_flag[1])	
abs_mvd_minus2[1]	ae(v)
mvd_sign_flag[1]	ae(v)
}	
}	

[00122] The corresponding semantics include:

[00123] **five_minus_max_num_merge_cand** specifies the maximum number of merging MVP candidates supported in the slice subtracted from 5. The maximum number of merging MVP candidates, MaxNumMergeCand is derived as follows:

[00124] $\text{MaxNumMergeCand} = 5 - \text{five_minus_max_num_merge_cand}$

[00125] The value of MaxNumMergeCand shall be in the range of 1 to 5, inclusive.

[00126] **merge_flag[x0][y0]** specifies whether the inter prediction parameters for the current prediction unit are inferred from a neighboring inter-predicted partition. The array indices x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered prediction block relative to the top-left luma sample of the picture.

[00127] When merge_flag[x0][y0] is not present, it is inferred as follows:

[00128] -- If CuPredMode[x0][y0] is equal to MODE_SKIP, merge_flag[x0][y0] is inferred to be equal to 1.

[00129] -- Otherwise, merge_flag[x0][y0] is inferred to be equal to 0.

[00130] **merge_idx[x0][y0]** specifies the merging candidate index of the merging candidate list where x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered prediction block relative to the top-left luma sample of the picture.

3. Example of inter prediction methods in Joint Exploration Model (JEM)

[00131] In some embodiments, future video coding technologies are explored using a reference software known as the Joint Exploration Model (JEM). In JEM, sub-block based prediction is adopted in several coding tools, such as affine prediction, alternative temporal motion vector prediction (ATMVP), spatial-temporal motion vector prediction (STMVP), bi-directional optical flow (BIO), Frame-Rate Up Conversion (FRUC), Locally Adaptive Motion Vector Resolution (LAMVR), Overlapped Block Motion Compensation (OBMC), Local Illumination Compensation (LIC), and Decoder-side Motion Vector Refinement (DMVR).

3.1 Examples of sub-CU based motion vector prediction

[00132] In the JEM with quadtrees plus binary trees (QTBT), each CU can have at most one set of motion parameters for each prediction direction. In some embodiments, two sub-CU level motion vector prediction methods are considered in the encoder by splitting a large CU into sub-CUs and deriving motion information for all the sub-CUs of the large CU. Alternative temporal motion vector prediction (ATMVP) method allows each CU to fetch multiple sets of motion information from multiple blocks smaller than the current CU in the collocated reference picture. In spatial-temporal motion vector prediction (STMVP) method motion vectors of the sub-CUs are derived recursively by using the temporal motion vector predictor and spatial neighbouring motion vector. In some embodiments, and to preserve more accurate motion field for sub-CU motion prediction, the motion compression for the reference frames may be disabled.

3.1.1 Examples of alternative temporal motion vector prediction (ATMVP)

[00133] In the ATMVP method, the temporal motion vector prediction (TMVP) method is modified by fetching multiple sets of motion information (including motion vectors and reference indices) from blocks smaller than the current CU.

[00134] FIG. 18 shows an example of ATMVP motion prediction process for a CU 1800. The ATMVP method predicts the motion vectors of the sub-CUs 1801 within a CU 1800 in two steps. The first step is to identify the corresponding block 1851 in a reference picture 1850 with a temporal vector. The reference picture 1850 is also referred to as the motion source picture. The second step is to split the current CU 1800 into sub-CUs 1801 and obtain the motion vectors as well as the reference indices of each sub-CU from the block

corresponding to each sub-CU.

[00135] In the first step, a reference picture 1850 and the corresponding block is determined by the motion information of the spatial neighboring blocks of the current CU 1800. To avoid the repetitive scanning process of neighboring blocks, the first merge candidate in the merge candidate list of the current CU 1800 is used. The first available motion vector as well as its associated reference index are set to be the temporal vector and the index to the motion source picture. This way, the corresponding block may be more accurately identified, compared with TMVP, wherein the corresponding block (sometimes called collocated block) is always in a bottom-right or center position relative to the current CU.

[00136] In one example, if the first merge candidate is from the left neighboring block (i.e., A_1 in FIG. 19), the associated MV and reference picture are utilized to identify the source block and source picture.

[00137] In the second step, a corresponding block of the sub-CU 1851 is identified by the temporal vector in the motion source picture 1850, by adding to the coordinate of the current CU the temporal vector. For each sub-CU, the motion information of its corresponding block (e.g., the smallest motion grid that covers the center sample) is used to derive the motion information for the sub-CU. After the motion information of a corresponding $N \times N$ block is identified, it is converted to the motion vectors and reference indices of the current sub-CU, in the same way as TMVP of HEVC, wherein motion scaling and other procedures apply. For example, the decoder checks whether the low-delay condition (e.g. the POCs of all reference pictures of the current picture are smaller than the POC of the current picture) is fulfilled and possibly uses motion vector MV_x (e.g., the motion vector corresponding to reference picture list X) to predict motion vector MV_y (e.g., with X being equal to 0 or 1 and Y being equal to $1-X$) for each sub-CU.

3.1.2 Examples of spatial-temporal motion vector prediction (STMVP)

[00138] In the STMVP method, the motion vectors of the sub-CUs are derived recursively, following raster scan order. FIG. 20 shows an example of one CU with four sub-blocks and neighboring blocks. Consider an 8×8 CU 2000 that includes four 4×4 sub-CUs A (2001), B (2002), C (2003), and D (2004). The neighboring 4×4 blocks in the current frame are labelled as a (2011), b (2012), c (2013), and d (2014).

[00139] The motion derivation for sub-CU A starts by identifying its two spatial neighbors. The first neighbor is the $N \times N$ block above sub-CU A 2001 (block c 2013). If this block c (2013) is not available or is intra coded the other $N \times N$ blocks above sub-CU A (2001) are checked (from left to right, starting at block c 2013). The second neighbor is a block to the left of the sub-CU A 2001 (block b 2012). If block b (2012) is not available or is intra coded other blocks to the left of sub-CU A 2001 are checked (from top to bottom, starting at block b 2012). The motion information obtained from the neighboring blocks for each list is scaled to the first reference frame for a given list. Next, temporal motion vector predictor (TMVP) of sub-block A 2001 is derived by following the same procedure of TMVP derivation as specified in HEVC. The motion information of the collocated block at block D 2004 is fetched and scaled accordingly. Finally, after retrieving and scaling the motion information, all available motion vectors are averaged separately for each reference list. The averaged motion vector is assigned as the motion vector of the current sub-CU.

3.1.3 Examples of sub-CU motion prediction mode signaling

[00140] In some embodiments, the sub-CU modes are enabled as additional merge candidates and there is no additional syntax element required to signal the modes. Two additional merge candidates are added to merge candidates list of each CU to represent the ATMVP mode and STMVP mode. In other embodiments, up to seven merge candidates may be used, if the sequence parameter set indicates that ATMVP and STMVP are enabled. The encoding logic of the additional merge candidates is the same as for the merge candidates in the HM, which means, for each CU in P or B slice, two more RD checks may be needed for the two additional merge candidates. In some embodiments, e.g., JEM, all bins of the merge index are context coded by CABAC (Context-based Adaptive Binary Arithmetic Coding). In other embodiments, e.g., HEVC, only the first bin is context coded and the remaining bins are context by-pass coded.

3.2 Examples of adaptive motion vector difference resolution

[00141] In some embodiments, motion vector differences (MVDs) (between the motion vector and predicted motion vector of a PU) are signalled in units of quarter luma samples when `use_integer_mv_flag` is equal to 0 in the slice header. In the JEM, a locally adaptive motion vector resolution (LAMVR) is introduced. In the JEM, MVD can be coded in units of quarter luma samples, integer luma samples or four luma samples. The MVD

resolution is controlled at the coding unit (CU) level, and MVD resolution flags are conditionally signalled for each CU that has at least one non-zero MVD components.

[00142] For a CU that has at least one non-zero MVD components, a first flag is signalled to indicate whether quarter luma sample MV precision is used in the CU. When the first flag (equal to 1) indicates that quarter luma sample MV precision is not used, another flag is signalled to indicate whether integer luma sample MV precision or four luma sample MV precision is used.

[00143] When the first MVD resolution flag of a CU is zero, or not coded for a CU (meaning all MVDs in the CU are zero), the quarter luma sample MV resolution is used for the CU. When a CU uses integer-luma sample MV precision or four-luma-sample MV precision, the MVPs in the AMVP candidate list for the CU are rounded to the corresponding precision.

[00144] In the encoder, CU-level RD checks are used to determine which MVD resolution is to be used for a CU. That is, the CU-level RD check is performed three times for each MVD resolution. To accelerate encoder speed, the following encoding schemes are applied in the JEM:

[00145] -- During RD check of a CU with normal quarter luma sample MVD resolution, the motion information of the current CU (integer luma sample accuracy) is stored. The stored motion information (after rounding) is used as the starting point for further small range motion vector refinement during the RD check for the same CU with integer luma sample and 4 luma sample MVD resolution so that the time-consuming motion estimation process is not duplicated three times.

[00146] -- RD check of a CU with 4 luma sample MVD resolution is conditionally invoked. For a CU, when RD cost integer luma sample MVD resolution is much larger than that of quarter luma sample MVD resolution, the RD check of 4 luma sample MVD resolution for the CU is skipped.

3.2.1 Examples of AMVP candidate list construction

[00147] In JEM, the procedure is similar to the HEVC design. However, when the current block chooses a lower precision of MVs (e.g., integer-precision), rounding operations may be applied. In the current implementation, after selecting the 2 candidates from spatial positions, if both are available, these two are rounded, followed by pruning.

3.3 Examples of pattern matched motion vector derivation (PMMVD)

[00148] The PMMVD mode is a special merge mode based on the Frame-Rate Up Conversion (FRUC) method. With this mode, motion information of a block is not signaled but derived at decoder side.

[00149] A FRUC flag can be signaled for a CU when its merge flag is true. When the FRUC flag is false, a merge index can be signaled and the regular merge mode is used. When the FRUC flag is true, an additional FRUC mode flag can be signaled to indicate which method (e.g., bilateral matching or template matching) is to be used to derive motion information for the block.

[00150] At the encoder side, the decision on whether using FRUC merge mode for a CU is based on RD cost selection as done for normal merge candidate. For example, multiple matching modes (e.g., bilateral matching and template matching) are checked for a CU by using RD cost selection. The one leading to the minimal cost is further compared to other CU modes. If a FRUC matching mode is the most efficient one, FRUC flag is set to true for the CU and the related matching mode is used.

[00151] Typically, motion derivation process in FRUC merge mode has two steps: a CU-level motion search is first performed, then followed by a Sub-CU level motion refinement. At CU level, an initial motion vector is derived for the whole CU based on bilateral matching or template matching. First, a list of MV candidates is generated and the candidate that leads to the minimum matching cost is selected as the starting point for further CU level refinement. Then a local search based on bilateral matching or template matching around the starting point is performed. The MV results in the minimum matching cost is taken as the MV for the whole CU. Subsequently, the motion information is further refined at sub-CU level with the derived CU motion vectors as the starting points.

[00152] For example, the following derivation process is performed for a $W \times H$ CU motion information derivation. At the first stage, MV for the whole $W \times H$ CU is derived. At the second stage, the CU is further split into $M \times M$ sub-CUs. The value of M is calculated as in Eq. (3), D is a predefined splitting depth which is set to 3 by default in the JEM. Then the MV for each sub-CU is derived.

[00153]
$$M = \max\{4, \min\{\frac{M}{2^D}, \frac{N}{2^D}\}\}$$

Eq. (3)

[00154] FIG. 21 shows an example of bilateral matching used in the Frame-Rate Up Conversion (FRUC) method. The bilateral matching is used to derive motion information of the current CU by finding the closest match between two blocks along the motion trajectory of the current CU (2100) in two different reference pictures (2110, 2111). Under the assumption of continuous motion trajectory, the motion vectors MV0 (2101) and MV1 (2102) pointing to the two reference blocks are proportional to the temporal distances, e.g., TD0 (2103) and TD1 (2104), between the current picture and the two reference pictures. In some embodiments, when the current picture 2100 is temporally between the two reference pictures (2110, 2111) and the temporal distance from the current picture to the two reference pictures is the same, the bilateral matching becomes mirror based bi-directional MV.

[00155] FIG. 22 shows an example of template matching used in the Frame-Rate Up Conversion (FRUC) method. Template matching can be used to derive motion information of the current CU 2200 by finding the closest match between a template (e.g., top and/or left neighboring blocks of the current CU) in the current picture and a block (e.g., same size to the template) in a reference picture 2210. Except the aforementioned FRUC merge mode, the template matching can also be applied to AMVP mode. In both JEM and HEVC, AMVP has two candidates. With the template matching method, a new candidate can be derived. If the newly derived candidate by template matching is different to the first existing AMVP candidate, it is inserted at the very beginning of the AMVP candidate list and then the list size is set to two (e.g., by removing the second existing AMVP candidate). When applied to AMVP mode, only CU level search is applied.

[00156] The MV candidate set at CU level can include the following: (1) original AMVP candidates if the current CU is in AMVP mode, (2) all merge candidates, (3) several MVs in the interpolated MV field (described later), and top and left neighboring motion vectors.

[00157] When using bilateral matching, each valid MV of a merge candidate can be used as an input to generate a MV pair with the assumption of bilateral matching. For example, one valid MV of a merge candidate is (MV_a, ref_a) at reference list A. Then the reference picture ref_b of its paired bilateral MV is found in the other reference list B so that ref_a and ref_b are temporally at different sides of the current picture. If such a ref_b is not

available in reference list B, ref_b is determined as a reference which is different from ref_a and its temporal distance to the current picture is the minimal one in list B. After ref_b is determined, MV_b is derived by scaling MV_a based on the temporal distance between the current picture and ref_a , ref_b .

[00158] In some implementations, four MVs from the interpolated MV field can also be added to the CU level candidate list. More specifically, the interpolated MVs at the position (0, 0), (W/2, 0), (0, H/2) and (W/2, H/2) of the current CU are added. When FRUC is applied in AMVP mode, the original AMVP candidates are also added to CU level MV candidate set. In some implementations, at the CU level, 15 MVs for AMVP CUs and 13 MVs for merge CUs can be added to the candidate list.

[00159] The MV candidate set at sub-CU level includes an MV determined from a CU-level search, (2) top, left, top-left and top-right neighboring MVs, (3) scaled versions of collocated MVs from reference pictures, (4) one or more ATMVP candidates (e.g., up to four), and (5) one or more STMVP candidates (e.g., up to four). The scaled MVs from reference pictures are derived as follows. The reference pictures in both lists are traversed. The MVs at a collocated position of the sub-CU in a reference picture are scaled to the reference of the starting CU-level MV. ATMVP and STMVP candidates can be the four first ones. At the sub-CU level, one or more MVs (e.g., up to 17) are added to the candidate list.

[00160] Generation of an interpolated MV field. Before coding a frame, interpolated motion field is generated for the whole picture based on unilateral ME. Then the motion field may be used later as CU level or sub-CU level MV candidates.

[00161] In some embodiments, the motion field of each reference pictures in both reference lists is traversed at 4×4 block level. FIG. 23 shows an example of unilateral Motion Estimation (ME) 2300 in the FRUC method. For each 4×4 block, if the motion associated to the block passing through a 4×4 block in the current picture and the block has not been assigned any interpolated motion, the motion of the reference block is scaled to the current picture according to the temporal distance TD0 and TD1 (the same way as that of MV scaling of TMVP in HEVC) and the scaled motion is assigned to the block in the current frame. If no scaled MV is assigned to a 4×4 block, the block's motion is marked as unavailable in the interpolated motion field.

[00162] Interpolation and matching cost. When a motion vector points to a

fractional sample position, motion compensated interpolation is needed. To reduce complexity, bi-linear interpolation instead of regular 8-tap HEVC interpolation can be used for both bilateral matching and template matching.

[00163] The calculation of matching cost is a bit different at different steps. When selecting the candidate from the candidate set at the CU level, the matching cost can be the absolute sum difference (SAD) of bilateral matching or template matching. After the starting MV is determined, the matching cost C of bilateral matching at sub-CU level search is calculated as follows:

$$\text{[00164]} \quad C = SAD + w \cdot (|MV_x - MV_x^s| + |MV_y - MV_y^s|)$$

Eq. (4)

[00165] Here, w is a weighting factor. In some embodiments, w can be empirically set to 4. MV and MV^s indicate the current MV and the starting MV, respectively. SAD may still be used as the matching cost of template matching at sub-CU level search.

[00166] In FRUC mode, MV is derived by using luma samples only. The derived motion will be used for both luma and chroma for MC inter prediction. After MV is decided, final MC is performed using 8-taps interpolation filter for luma and 4-taps interpolation filter for chroma.

[00167] MV refinement is a pattern based MV search with the criterion of bilateral matching cost or template matching cost. In the JEM, two search patterns are supported – an unrestricted center-biased diamond search (UCBDS) and an adaptive cross search for MV refinement at the CU level and sub-CU level, respectively. For both CU and sub-CU level MV refinement, the MV is directly searched at quarter luma sample MV accuracy, and this is followed by one-eighth luma sample MV refinement. The search range of MV refinement for the CU and sub-CU step are set equal to 8 luma samples.

[00168] In the bilateral matching merge mode, bi-prediction is applied because the motion information of a CU is derived based on the closest match between two blocks along the motion trajectory of the current CU in two different reference pictures. In the template matching merge mode, the encoder can choose among uni-prediction from list0, uni-prediction from list1, or bi-prediction for a CU. The selection can be based on a template matching cost as follows:

$$\text{[00169]} \quad \text{If } \text{costBi} \leq \text{factor} * \min(\text{cost0}, \text{cost1})$$

[00170] bi-prediction is used;
 [00171] Otherwise, if $\text{cost0} \leq \text{cost1}$
 [00172] uni-prediction from list0 is used;
 [00173] Otherwise,
 [00174] uni-prediction from list1 is used;
 [00175] Here, cost0 is the SAD of list0 template matching, cost1 is the SAD of list1 template matching and costBi is the SAD of bi-prediction template matching. For example, when the value of factor is equal to 1.25, it means that the selection process is biased toward bi-prediction. The inter prediction direction selection can be applied to the CU-level template matching process.

3.4 Examples of decoder-side motion vector refinement (DMVR)

[00176] In a bi-prediction operation, for the prediction of one block region, two prediction blocks, formed using a motion vector (MV) of list0 and a MV of list1, respectively, are combined to form a single prediction signal. In the decoder-side motion vector refinement (DMVR) method, the two motion vectors of the bi-prediction are further refined by a bilateral template matching process. The bilateral template matching applied in the decoder to perform a distortion-based search between a bilateral template and the reconstruction samples in the reference pictures in order to obtain a refined MV without transmission of additional motion information.

[00177] In DMVR, a bilateral template is generated as the weighted combination (i.e. average) of the two prediction blocks, from the initial MV0 of list0 and MV1 of list1, respectively, as shown in FIG. 24. The template matching operation consists of calculating cost measures between the generated template and the sample region (around the initial prediction block) in the reference picture. For each of the two reference pictures, the MV that yields the minimum template cost is considered as the updated MV of that list to replace the original one. In the JEM, nine MV candidates are searched for each list. The nine MV candidates include the original MV and 8 surrounding MVs with one luma sample offset to the original MV in either the horizontal or vertical direction, or both. Finally, the two new MVs, i.e., MV0' and MV1' as shown in FIG. 24, are used for generating the final bi-prediction results. A sum of absolute differences (SAD) is used as the cost measure.

[00178] DMVR is applied for the merge mode of bi-prediction with one MV from

a reference picture in the past and another from a reference picture in the future, without the transmission of additional syntax elements. In the JEM, when LIC, affine motion, FRUC, or sub-CU merge candidate is enabled for a CU, DMVR is not applied.

3.5 Local illumination compensation

[00179] Local Illumination Compensation (IC) is based on a linear model for illumination changes, using a scaling factor a and an offset b . And it is enabled or disabled adaptively for each inter-mode coded coding unit (CU).

[00180] When IC applies for a CU, a least square error method is employed to derive the parameters a and b by using the neighbouring samples of the current CU and their corresponding reference samples. More specifically, as illustrated in FIG. 25, the subsampled (2:1 subsampling) neighbouring samples of the CU and the corresponding samples (identified by motion information of the current CU or sub-CU) in the reference picture are used. The IC parameters are derived and applied for each prediction direction separately.

[00181] When a CU is coded with merge mode, the IC flag is copied from neighbouring blocks, in a way similar to motion information copy in merge mode; otherwise, an IC flag is signalled for the CU to indicate whether LIC applies or not.

[00182] When IC is enabled for a picture, additional CU level RD check is needed to determine whether LIC is applied or not for a CU. When IC is enabled for a CU, mean-removed sum of absolute difference (MR-SAD) and mean-removed sum of absolute Hadamard-transformed difference (MR-SATD) are used, instead of SAD and SATD, for integer pel motion search and fractional pel motion search, respectively.

[00183] To reduce the encoding complexity, the following encoding scheme is applied in the JEM. IC is disabled for the entire picture when there is no obvious illumination change between a current picture and its reference pictures. To identify this situation, histograms of a current picture and every reference picture of the current picture are calculated at the encoder. If the histogram difference between the current picture and every reference picture of the current picture is smaller than a given threshold, IC is disabled for the current picture; otherwise, IC is enabled for the current picture.

3.6 Examples of merge/skip mode with bilateral matching refinement

[00184] A merge candidate list is first constructed by inserting the motion vectors and reference indices of the spatial neighboring and temporal neighboring blocks into the

candidate list with redundancy checking until the number of the available candidates reaches the maximum candidate size of 19. The merge candidate list for the merge/skip mode is constructed by inserting spatial candidates, temporal candidates, affine candidates, advanced temporal MVP (ATMVP) candidate, spatial temporal MVP (STMVP) candidate and the additional candidates as used in HEVC (Combined candidates and Zero candidates) according to a pre-defined insertion order, and in the context of the numbered blocks shown in FIG. 25:

- [00185] (1) Spatial candidates for blocks 1-4
- [00186] (2) Extrapolated affine candidates for blocks 1-4
- [00187] (3) ATMVP
- [00188] (4) STMVP
- [00189] (5) Virtual affine candidate
- [00190] (6) Spatial candidate (block 5) (used only when the number of the available candidates is smaller than 6)
- [00191] (7) Extrapolated affine candidate (block 5)
- [00192] (8) Temporal candidate (derived as in HEVC)
- [00193] (9) **Non-adjacent spatial candidate followed by extrapolated affine candidate (blocks 6 to 49)**
- [00194] (10) Combined candidates
- [00195] (11) Zero candidates

[00196] It may be noted that IC flags are also inherited from merge candidates except for STMVP and affine. Moreover, for the first four spatial candidates, the bi-prediction ones are inserted before the ones with uni-prediction.

4. Examples of binarization methods and merge index coding

[00197] In some embodiments, several binarization methods could be selected. For one syntax element, the related value should be firstly binarized to a bin string based on the distribution. For each bin, it may be coded with context or bypass coding methods.

4.1 Exemplary Unary and Truncated Unary (TU) binarization process

[00198] For each unsigned integer valued symbol $x \geq 0$ the unary code word in CABAC consists of x “1” bits plus a terminating “0” bit. The truncated unary (TU) code is only defined for x with $0 \leq x \leq S$, where for $x < S$ the code is given by the unary code,

whereas for $x = S$ the terminating “0” bit is neglected such that the TU code of $x = S$ is given by codeword consisting of x “1” bits only.

Table 5: Bin string of the unary binarization

prefixVal	Bin string					
0	0					
1	1	0				
2	1	1	0			
3	1	1	1	0		
4	1	1	1	1	0	
5	1	1	1	1	1	0
...						
binIdx	0	1	2	3	4	5

Table 6: Bin string of the truncated unary binarization

prefixVal	Bin string					
0	0					
1	1	0				
2	1	1	0			
3	1	1	1	0		
4	1	1	1	1	0	
5	1	1	1	1	1	0
binIdx	0	1	2	3	4	5

4.2 Exemplary K-th order Exp-Golomb (EGk) binarization process

[00199] For the EGk binarization the number of symbols having the same code length of $k + 2 \cdot l(x) + 1$ is geometrically growing. By inverting Shannon’s relationship between ideal code length and symbol probability, we can e.g. easily deduce that EG0 is the optimal code for a pdf $p(x) = \frac{1}{2} \cdot (x + 1)^{-2}$ with $x \geq 0$. This implies that for an appropriately chosen parameter k the EGk code represents a fairly good first-order approximation of the ideal prefix-free code for tails of typically observed pdfs.

Table 7: Bin string of the EG0 binarization

	K=0						
prefixVal	Bin string						
0	0						
1	1	0	0				
2	1	0	1				
3	1	1	0	0	0		
4	1	1	0	0	1		
5	1	1	0	1	0		
6	1	1	0	1	1		
7	1	1	1	0	0	0	0
...							
binIdx	0	1	2	3	4	5	6

4.3 Exemplary Truncated Rice (TR) binarization process

[00200] Input to this process is a request for a TR binarization, cMax, and cRiceParam.

[00201] Output of this process is the TR binarization associating each value symbolVal with a corresponding bin string.

[00202] A TR bin string is a concatenation of a prefix bin string and, when present, a suffix bin string.

[00203] For the derivation of the prefix bin string, the following applies:

[00204] -- The prefix value of symbolVal, prefixVal, is derived as follows:

[00205] $\text{prefixVal} = \text{symbolVal} \gg \text{cRiceParam}$

[00206] -- The prefix of the TR bin string is specified as follows:

[00207] If prefixVal is less than $\text{cMax} \gg \text{cRiceParam}$, the prefix bin string is a bit string of length prefixVal + 1 indexed by binIdx. The bins for binIdx less than prefixVal are equal to 1. The bin with binIdx equal to prefixVal is equal to 0.

[00208] When cMax is greater than symbolVal and cRiceParam is greater than 0, the suffix of the TR bin string is present and it is derived as follows:

[00209] -- The suffix value suffixVal is derived as follows:

[00210] $\text{suffixVal} = \text{symbolVal} - ((\text{prefixVal}) \ll \text{cRiceParam})$

[00211] -- The suffix of the TR bin string is specified by invoking the fixed-length (FL) binarization process for suffixVal with a cMax value equal to $(1 \ll \text{cRiceParam}) - 1$.

[00212] Note that for the input parameter $\text{cRiceParam} = 0$ the TR binarization is exactly a truncated unary binarization and it is always invoked with a cMax value equal to the largest possible value of the syntax element being decoded.

4.4 Exemplary Fixed-Length (FL) binarization process

[00213] Inputs to this process are a request for a FL binarization and cMax.

[00214] Output of this process is the FL binarization associating each value symbolVal with a corresponding bin string.

[00215] FL binarization is constructed by using the fixedLength-bit unsigned integer bin string of the symbol value symbolVal, where $\text{fixedLength} = \text{Ceil}(\text{Log2}(\text{cMax} + 1))$. The indexing of bins for the FL binarization is such that the $\text{binIdx} = 0$ relates to the most significant bit with increasing values of binIdx towards the least significant bit.

Table 8: Bin string of the FL binarization (cMax = 7)

prefixVal	Bin String		
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
...			
binIdx	0	1	2

4.5 Exemplary coding of merge_idx

[00216] As specified in the HEVC specification, the merge index is firstly binarized to a bin string if the total number of allowed merge candidate is larger than 1.

Table 9: binarization and ways of coding bins for merge_idx

merge_idx[][]		TR /T U	cMax = MaxNumMergeCand - 1, cRiceParam = 0			
Syntax element	binIdx					
	0	1	2	3	4	>= 5
merge_idx[][]	0	bypass	bypass	bypass	n/a	N/a

[00217] TR with cRiceParam equal to 0, i.e., TU is used. The first bin of merge_idx is coded with one context and the remaining bins, if exist, are coded with bypass.

5 Example embodiments of intra prediction in JEM

5.1 Examples of intra mode coding with 67 intra prediction modes

[00218] To capture the arbitrary edge directions presented in natural video, the number of directional intra modes is extended from 33, as used in HEVC, to 65. The additional directional modes are depicted as light grey dotted arrows in FIG. 27, and the planar and DC modes remain the same. These denser directional intra prediction modes apply for all block sizes and for both luma and chroma intra predictions.

5.2 Examples of luma intra mode coding

[00219] In JEM, the total number of intra prediction modes has been increased from 35 in HEVC to 67. FIG. 27 depicts the examples of the 67 intra prediction modes.

[00220] To accommodate the increased number of directional intra modes, an intra mode coding method with 6 Most Probable Modes (MPMs) is used. Two major technical aspects are involved: 1) the derivation of 6 MPMs, and 2) entropy coding of 6 MPMs and non-MPM modes.

[00221] In the JEM, the modes included into the MPM lists are classified into three groups:

[00222] -- Neighbor intra modes

[00223] -- Derived intra modes

[00224] -- Default intra modes

[00225] Five neighbouring intra prediction modes are used to form the MPM list. Those locations of the 5 neighbouring blocks are the same as those used in the merge mode, i.e., left (L), above (A), below-left (BL), above-right (AR), and above-left (AL) as shown in FIG. 28. An initial MPM list is formed by inserting 5 neighbour intra modes and the planar

and DC modes into the MPM list. A pruning process is used to remove duplicated modes so that only unique modes can be included into the MPM list. The order in which the initial modes are included is: left, above, planar, DC, below-left, above-right, and then above-left.

[00226] If the MPM list is not full (i.e., there are less than 6 MPM candidates in the list), derived modes are added; these intra modes are obtained by adding -1 or $+1$ to the angular modes that are already included in the MPM list. Such additional derived modes are not generated from the non-angular modes (DC or planar).

[00227] Finally, if the MPM list is still not complete, the default modes are added in the following order: **vertical, horizontal, mode 2, and diagonal mode**. As a result of this process, a unique list of 6 MPM modes is generated.

[00228] For entropy coding of the selected mode using the 6 MPMs, a truncated unary binarization is used. The first three bins are coded with contexts that depend on the MPM mode related to the bin currently being signalled. The MPM mode is classified into one of three categories: (a) modes that are predominantly horizontal (i.e., the MPM mode number is less than or equal to the mode number for the diagonal direction), (b) modes that are predominantly vertical (i.e., the MPM mode is greater than the mode number for the diagonal direction), and (c) the non-angular (DC and planar) class. Accordingly, three contexts are used to signal the MPM index based on this classification.

[00229] The coding for selection of the remaining 61 non-MPMs is done as follows. The 61 non-MPMs are first divided into two sets: a selected mode set and a non-selected mode set. The selected modes set contains 16 modes and the rest (45 modes) are assigned to the non-selected modes set. The mode set that the current mode belongs to is indicated in the bitstream with a flag. If the mode to be indicated is within the selected modes set, the selected mode is signalled with a 4-bit fixed-length code, and if the mode to be indicated is from the non-selected set, the selected mode is signalled with a truncated binary code. The selected modes set is generated by sub-sampling the 61 non-MPM modes as follows:

[00230] -- Selected modes set = {0, 4, 8, 12, 16, 20 ... 60}

[00231] -- Non-selected modes set = {1, 2, 3, 5, 6, 7, 9, 10 ... 59}

[00232] At the encoder side, the similar two-stage intra mode decision process of HM is used. In the first stage, i.e., the intra mode pre-selection stage, a lower complexity

Sum of Absolute Transform Difference (SATD) cost is used to pre-select N intra prediction modes from all the available intra modes. In the second stage, a higher complexity R-D cost selection is further applied to select one intra prediction mode from the N candidates.

However, when 67 intra prediction modes is applied, since the total number of available modes is roughly doubled, the complexity of the intra mode pre-selection stage will also be increased if the same encoder mode decision process of HM is directly used. To minimize the encoder complexity increase, a two-step intra mode pre-selection process is performed. At the first step, N (N depends on intra prediction block size) modes are selected from the original 35 intra prediction modes (indicated by black solid arrows in FIG. 27) based on the Sum of Absolute Transform Difference (SATD) measure; At the second step, the direct neighbours (additional intra prediction directions as indicated by light grey dotted arrows in FIG. 27) of the selected N modes are further examined by SATD, and the list of selected N modes are updated. Finally, the first M MPMs are added to the N modes if not already included, and the final list of candidate intra prediction modes is generated for the second stage R-D cost examination, which is done in the same way as HM. The value of M is increased by one based on the original setting in the HM, and N is decreased somewhat as shown below in Table 10.

Table 10: Number of mode candidates at the intra mode pre-selection step

Intra prediction block size	4×4	8×8	16×16	32×32	64×64	>64×64
HM	8	8	3	3	3	3
JEM with 67 intra prediction modes	7	7	2	2	2	2

5.3 Examples of chroma intra mode coding

[00233] In the JEM, a total of 11 intra modes are allowed for chroma CB coding. Those modes include 5 traditional intra modes and 6 cross-component linear model modes. The list of chroma mode candidates includes the following three parts:

[00234] • CCLM modes

[00235] • DM modes, intra prediction modes derived from luma CBs covering the collocated five positions of the current chroma block

[00236] ○ The five positions to be checked in order are: center (CR), top-left (TL), top-right (TR), bottom-left (BL) and bottom-right (BR) 4×4 block within the corresponding luma block of current chroma block for I slices. For P and B slices, only one of these five sub-blocks is checked since they have the same mode index. An example of five collocated luma positions is shown in FIG. 29.

- Chroma prediction modes from spatial neighbouring blocks:

[00238] ○ 5 chroma prediction modes: from left, above, below-left,
above-right, and above-left spatially neighbouring blocks

[00239] ○ Planar and DC modes

[00240] ○ Derived modes are added, these intra modes are obtained by adding -1 or $+1$ to the angular modes which are already included into the list

[00241] ○ Vertical, horizontal, mode 2

[00242] A pruning process is applied whenever a new chroma intra mode is added to the candidate list. The non-CCLM chroma intra mode candidates list size is then trimmed to 5. For the mode signalling, a flag is first signalled to indicate whether one of the CCLM modes or one of the traditional chroma intra prediction mode is used. Then a few more flags may follow to specify the exact chroma prediction mode used for the current chroma CBs.

6. Examples of existing implementations

[00243] The current HEVC design could take the correlation of current block its neighbouring blocks (next to the current block) to better code the motion information. However, it is possible that that the neighbouring blocks correspond to different objects with different motion trajectories. In this case, prediction from its neighbouring blocks is not efficient.

[00244] Prediction from motion information of non-adjacent blocks could bring additional coding gain with the cost of storing all the motion information (typically on 4x4 level) into cache which significantly increase the complexity for hardware implementation.

[00245] Unary binarization method works fine for smaller number of allowed merge candidates. However, when the total number of allowed candidate becomes larger, the unary binarization may be sub-optimal.

[00246] The HEVC design of AMVP candidate list construction process only invokes pruning among two spatial AMVP candidates. Full pruning (each of the available

candidate compared to all others) is not utilized since there is negligible coding loss due to limited pruning. However, if there are more AMVP candidates available, pruning becomes important. Also, when LAMVR is enabled, how to construct the AVMP candidate list should be studied.

7. Example methods for LUT-based motion vector prediction

[00247] Embodiments of the presently disclosed technology overcome the drawbacks of existing implementations, thereby providing video coding with higher coding efficiencies. LUT-based motion vector prediction techniques using one or more tables (e.g., look up tables) with at least one motion candidate stored to predict motion information of a block can be implemented in various embodiments to provide video coding with higher coding efficiencies. A look up table is an example of a table which can be used to include motion candidates to predict motion information of a block and other implementations are also possible. Each LUT can include one or more motion candidates, each associated with corresponding motion information. Motion information of a motion candidate can include partial or all of the prediction direction, reference indices/pictures, motion vectors, LIC flags, affine flags, Motion Vector Derivation (MVD) precisions, and/or MVD values. Motion information may further include the block position information to indicate from which the motion information is coming.

[00248] The LUT-based motion vector prediction based on the disclosed technology, which may enhance both existing and future video coding standards, is elucidated in the following examples described for various implementations. Because the LUTs allow the encoding/decoding process to be performed based on historical data (e.g., the blocks that have been processed), the LUT-based motion vector prediction can also be referred to as History-based Motion Vector Prediction (HMVP) method. In the LUT-based motion vector prediction method, one or multiple tables with motion information from previously coded blocks are maintained during the encoding/decoding process. These motion candidates stored in the LUTs are named HMVP candidates. During the encoding/decoding of one block, the associated motion information in LUTs may be added to the motion candidate lists (e.g., merge/AMVP candidate lists), and after encoding/decoding one block, LUTs may be updated. The updated LUTs are then used to code the subsequent blocks. That is, the updating of motion candidates in the LUTs are based on the encoding/decoding order

of blocks. The examples below should be considered as examples to explain general concepts. These examples should not be interpreted in a narrow way. Furthermore, these examples can be combined in any manner.

[00249] The LUT-based motion vector prediction based on the disclosed technology, which may enhance both existing and future video coding standards, is elucidated in the following examples described for various implementations. The examples of the disclosed technology provided below explain general concepts, and are not meant to be interpreted as limiting. In an example, unless explicitly indicated to the contrary, the various features described in these examples may be combined.

[00250] With regard to terminology, the following examples an entry of a LUT is a motion candidate. The term *motion candidate* is used to indicate a set of motion information stored in a look up table. For conventional AMVP or merge modes, *AMVP or merge candidates* are used for storing the motion information. As will be described below, and in a non-limiting example, the notion of LUTs with motion candidates for motion vector prediction is extended to LUTs with intra prediction modes for intra mode coding, or extended to LUTs with illumination compensation parameters for IC parameter coding or extended to LUTs with filter parameters. The LUT based methods for motion candidates may be extended to other types of coding information, as described in this patent document, existing and future video coding standards.

[00251] Examples 1 and 2 advantageously define LUTs as used in the disclosed technology.

[00252] Example 1. Each look up table may contain one or more motion candidates wherein each candidate is associated with its motion information.

a. The table size (number of allowed motion candidates) and/or number of tables may be the fixed or adaptive. The table size may be same for all tables, or different for different tables.

i. Alternatively, different sizes may be used for different look-up tables (e.g., 1 or 2).

ii. In one example, the table sizes and/or number of tables may be pre-defined.

iii. In one example, the table sizes and/or number of tables may be signaled in Video Parameter Set (VPS), Sequence Parameter Set (SPS), Picture Parameter Set (PPS), Slice header, tile header, Coding Tree Unit (CTU), Coding Tree Block (CTB), Coding Unit (CU) or Prediction Unit (PU), region covering multiple CTU/CTB/CU/PUs.

iv. The table size and/or number of tables may further depend on the slice type, temporal layer index of a picture, picture order count (POC) distance between one slice and the closest intra slice.

b. Suppose there are N tables used for a coding thread, N*P tables may be required for coding a slice, wherein P indicates the number of LCU rows or the number of tiles.

i. Alternatively, only P tables may be required for coding a slice, wherein P indicates the number of LCU rows wherein each LCU row only use one look up table even N could be larger than 1 when tile is disabled.

[00253] Example 2. In one example, the table size (e.g., the number of maximally allowed entries of motion candidates) and/or number of tables may depend on sequence resolution, largest coding unit size, size of merge candidate lists.

[00254] (a) In one example, there is no additional signaling of the table size.

[00255] (b) The table size is inferred to be the same as the signaled maximum merge candidates.

[00256] (c) Alternatively, the table size is inferred to be the same as the signaled maximum merge candidates minus a certain value. The certain value may be defined as how many spatial/temporal neighboring blocks may be accessed in the merge candidate list construction process, e.g., 5 in HEVC.

[00257] Example 3. A LUT may be updated with motion information associated with the block when the block is coded with merge or AMVP mode.

[00258] (a) In one example, all of the LUTs are not updated after coding/decoding a block with intra mode.

[00259] (b) In one example, all of the LUTs are not updated after coding/decoding a block with intra block copy mode.

[00260] (c) In one example, all of the LUTs are not updated after

coding/decoding a block with affine merge or affine AMVP mode.

[00261] Example 4. Before updating a LUT by adding a motion candidate obtained from a coded block, pruning may be applied.

[00262] Example 5. A LUT may be updated periodically.

[00263] Example 6. A LUT may be used when a block is coded with merge or AMVP mode.

[00264] Example 7. When a block with certain information (such as block mode) is not allowed to use the information from LUTs, the coded motion information is not allowed to update LUTs. That is, after coding or decoding such a block, the process of updating LUTs is skipped.

[00265] In some implementations, similar to the usage of LUTs with motion candidates for motion vector prediction, it is proposed that one or multiple LUTs may be constructed, and/or updated to store intra prediction modes from previously coded blocks and LUTs may be used for coding/decoding an intra-coded block.

8. Additional embodiments for LUT-based motion vector prediction

[00266] An example of the coding flow for the LUT-based prediction methods is depicted in FIG. 30. In some implementations, the updating process is done after decoding a region. An example is depicted in FIGS. 31 to avoid frequently updating LUTs.

[00267] The examples described above may be incorporated in the context of the methods described below, e.g., method 3200, which may be implemented at a video decoder and/or video encoder.

[00268] FIG. 32A shows a flowchart of an exemplary method for video processing. The method 3200 includes, at step 3202, maintaining N tables, indexed by an index $i=1$ to N, wherein an i -th table has a size equal to $M(i)$ corresponding to a maximum number of motion candidates in the i -th table, wherein i , N, and $M(i)$ are integers, and wherein each motion candidate is associated with corresponding motion information. The method 3200 includes, at step 3204, performing a conversion between a current video block and a bitstream representation of the video data including the current video block using motion candidates from the tables.

[00269] FIG. 32B shows another flowchart of an exemplary method for video processing. The method 3600 includes, at step 3602, receiving a bitstream representation of a

video data including a current video block. The method 366 further includes, at step 3604, processing the bitstream representation based on N tables, indexed by an index $i=1$ to N , wherein an i -th table has a size equal to $M(i)$ corresponding to a maximum number of motion candidates in the i -th table, wherein i , N , and $M(i)$ are integers, and wherein each motion candidate is associated with corresponding motion information.

9. Example implementations of the disclosed technology

[00270] FIG. 33 is a block diagram of a video processing apparatus 3300. The apparatus 3300 may be used to implement one or more of the methods described herein. The apparatus 3300 may be embodied in a smartphone, tablet, computer, Internet of Things (IoT) receiver, and so on. The apparatus 3300 may include one or more processors 3302, one or more memories 3304 and video processing hardware 3306. The processor(s) 3302 may be configured to implement one or more methods (including, but not limited to, methods 3200 and 3600) described in the present document. The memory (memories) 3304 may be used for storing data and code used for implementing the methods and techniques described herein. The video processing hardware 3306 may be used to implement, in hardware circuitry, some techniques described in the present document.

[00271] In some embodiments, the video coding methods may be implemented using an apparatus that is implemented on a hardware platform as described with respect to FIG. 36.

[00272] Features and embodiments of the above-described methods/techniques are described below.

[00273] 1. A video processing method, comprising: maintaining N tables, indexed by an index $i=1$ to N , wherein an i -th table has a size equal to $M(i)$ corresponding to a maximum number of motion candidates in the i -th table, wherein i , N , and $M(i)$ are integers, and wherein each motion candidate is associated with corresponding motion information; and performing a conversion between a current video block and a bitstream representation of a video data including the current video block using motion candidates from the tables.

[00274] 2. The method of clause 1, wherein the N tables include motion information from previously processed video blocks in the video data.

[00275] 3. The method of clause 1, wherein $M(i)$ is same for all tables.

[00276] 4. The method of clause 1, wherein at least some tables have a

size different from that of any one of other tables.

[00277] 5. The method of clause 1, wherein N and $M(i)$ are predefined.

[00278] 6. The method of clause 1, wherein sizes of the tables are signaled in one or more of a Video Parameter Set, a Sequence Parameter Set, a Picture Parameter Set, a Slice Header, or a tile header.

[00279] 7. The method of clause 1, wherein sizes of the tables are signaled in a bit field corresponding to one or more of a Coding Tree Unit or a Coding Tree Block Unit or a Coding Unit or a Prediction Unit.

[00280] 8. The method of clause 1, wherein at least one of N and $M(i)$ is dependent on at least one of i) a slice type, ii) temporal layer index of a picture, or iii) picture order count (POC) distance between one slice and an intra slice closest to the slice.

[00281] 9. The method of clause 1, wherein the current video block corresponds to a slice of video data, and wherein N is equal to $K \cdot P$, wherein K is an integer representing tables per decoding thread, and P is an integer representing a number of Largest Coding Unit (LCU) rows or a number of tiles in the slice of video data.

[00282] 10. The method of clause 1, wherein the portion of video data corresponds to a slice of video data, and wherein N is equal to P when each LCU row uses a single table and P is an integer representing a number of Largest Coding Unit rows or a number of tiles in the slice of video data.

[00283] 11. The method of clause 1, wherein at least one of N or $M(i)$ is based on a sequence resolution, a largest coding unit (LCU) size, or a size of a merge or Advanced Motion Vector Prediction (AMVP) candidate list.

[00284] 12. The method of clause 1, wherein a size of a table is not signaled.

[00285] 13. The method of clause 1, wherein a size of a table is same as the maximum number of merge candidates in a merge candidate list.

[00286] 14. The method of clause 1, wherein a size of a table is same as one obtained by subtracting a certain value from the maximum number of merge candidates in a merge candidate list.

[00287] 15. The method of clause 14, wherein the certain value represents how many spatial or temporal neighboring blocks are accessed during a construction

of a merge candidate list.

[00288] 16. A method for video processing, comprising: receiving a bitstream representation of a video data including a current video block; and processing the bitstream representation based on N tables, indexed by an index $i=1$ to N, wherein an i -th table has a size equal to $M(i)$ corresponding to a maximum number of motion candidates in the i -th table, wherein i , N, and $M(i)$ are integers, and wherein each motion candidate is associated with corresponding motion information.

[00289] 17. The method of clause 16, wherein the N tables include motion information from previously processed video blocks in the video data.

[00290] 18. The method of clause 16, further comprising: updating one or more tables based on motion information from video blocks in the video data.

[00291] 19. The method of clause 16, wherein each motion candidate is associated with corresponding motion information.

[00292] 20. The method of clause 16, wherein at least one of N or $M(i)$ is based on a sequence resolution, a largest coding unit (LCU) size, or a size of a merge or Advanced Motion Vector Prediction (AMVP) candidate list.

[00293] 21. The method of clause 16, wherein a size of a table is not signaled.

[00294] 22. The method of clause 16, wherein a size of a table is equal to the maximum number of merge candidates in a merge candidate list.

[00295] 23. The method of clause 16, wherein a size of a table is obtained by subtracting a fixed value from the maximum number of merge candidates in a merge candidate list.

[00296] 24. The method of clause 23, wherein the fixed value is based on the number of temporal or spatial neighboring blocks that are accessed during a construction of a merge candidate list or AMVP candidate list.

[00297] 25. The method of clause 16, wherein $M(i)$ is same for all tables.

[00298] 26. The method of clause 16, wherein at least some tables have a size different from that of any one of other tables.

[00299] 27. The method of clause 16, wherein N and $M(i)$ are predefined.

[00300] 28. The method of clause 16, wherein sizes of the tables are

signaled in one or more of a Video Parameter Set, a Sequence Parameter Set, a Picture Parameter Set, a Slice Header, or a tile header.

[00301] 29. The method of clause 16, wherein sizes of the tables are signaled in a bit field corresponding to one or more of a Coding Tree Unit, a Coding Tree Block Unit, a Coding Unit, or a Prediction Unit.

[00302] 30. The method of clause 16, wherein at least one of N or M(i) is dependent on at least one of i) a slice type, ii) temporal layer index of a picture, or iii) picture order count (POC) distance between one slice and an intra slice closest to the slice.

[00303] 31. The method of clause 16, wherein the current video block corresponds to a slice of video data, and wherein N is equal to $K \cdot P$, wherein K is an integer representing tables per decoding thread, and P is an integer representing a number of Largest Coding Unit rows or a number of tiles in the slice of video data.

[00304] 32. The method of clause 16, wherein the current video block corresponds to a slice of video data, and wherein N is equal to P when each LCU row uses a single table and P is an integer representing a number of Largest Coding Unit rows or a number of tiles in the slice of video data.

[00305] 33. The method of any one of clauses 1-32, wherein the performing of the conversion includes generating the bitstream representation from the current video block.

[00306] 34. The method of any one of clauses 1-32, wherein the performing of the conversion includes generating the current video block from the bitstream representation.

[00307] 35. The method of any one of clauses 1 to 34, wherein a motion candidate is associated with motion information including at least one of: a prediction direction, a reference picture index, motion vector values, an intensity compensation flag, an affine flag, a motion vector difference precision, or motion vector difference value.

[00308] 36. The method of any one of clauses 16-32, further comprising updating, based on the conversion, one or multiple tables.

[00309] 37. The method of any one of clauses 1-36, wherein the updating of one or more tables includes updating one or more tables based on the motion information of the current video block after performing the conversion.

[00310] 38. The method of clause 37, further comprising: performing a conversion between a subsequent video block of the video data and the bitstream representation of the video data based on the updated tables.

[00311] 39. An apparatus in a video system comprising a processor and a non-transitory memory with instructions thereon, wherein the instructions upon execution by the processor, cause the processor to implement the method in any one of clauses 1-38.

[00312] 40. A computer program product stored on a non-transitory computer readable media, the computer program product including program code for carrying out the method in any one of clauses 1-38.

[00313] From the foregoing, it will be appreciated that specific embodiments of the presently disclosed technology have been described herein for purposes of illustration, but that various modifications may be made without deviating from the scope of the invention. Accordingly, the presently disclosed technology is not limited except as by the appended claims.

[00314] Implementations of the subject matter and the functional operations described in this patent document can be implemented in various systems, digital electronic circuitry, or in computer software, firmware, or hardware, including the structures disclosed in this specification and their structural equivalents, or in combinations of one or more of them. Implementations of the subject matter described in this specification can be implemented as one or more computer program products, i.e., one or more modules of computer program instructions encoded on a tangible and non-transitory computer readable medium for execution by, or to control the operation of, data processing apparatus. The computer readable medium can be a machine-readable storage device, a machine-readable storage substrate, a memory device, a composition of matter effecting a machine-readable propagated signal, or a combination of one or more of them. The term “data processing unit” or “data processing apparatus” encompasses all apparatus, devices, and machines for processing data, including by way of example a programmable processor, a computer, or multiple processors or computers. The apparatus can include, in addition to hardware, code that creates an execution environment for the computer program in question, e.g., code that constitutes processor firmware, a protocol stack, a database management system, an operating system, or a combination of one or more of them.

[00315] A computer program (also known as a program, software, software application, script, or code) can be written in any form of programming language, including compiled or interpreted languages, and it can be deployed in any form, including as a stand-alone program or as a module, component, subroutine, or other unit suitable for use in a computing environment. A computer program does not necessarily correspond to a file in a file system. A program can be stored in a portion of a file that holds other programs or data (e.g., one or more scripts stored in a markup language document), in a single file dedicated to the program in question, or in multiple coordinated files (e.g., files that store one or more modules, sub programs, or portions of code). A computer program can be deployed to be executed on one computer or on multiple computers that are located at one site or distributed across multiple sites and interconnected by a communication network.

[00316] The processes and logic flows described in this specification can be performed by one or more programmable processors executing one or more computer programs to perform functions by operating on input data and generating output. The processes and logic flows can also be performed by, and apparatus can also be implemented as, special purpose logic circuitry, e.g., an FPGA (field programmable gate array) or an ASIC (application specific integrated circuit).

[00317] Processors suitable for the execution of a computer program include, by way of example, both general and special purpose microprocessors, and any one or more processors of any kind of digital computer. Generally, a processor will receive instructions and data from a read only memory or a random access memory or both. The essential elements of a computer are a processor for performing instructions and one or more memory devices for storing instructions and data. Generally, a computer will also include, or be operatively coupled to receive data from or transfer data to, or both, one or more mass storage devices for storing data, e.g., magnetic, magneto optical disks, or optical disks. However, a computer need not have such devices. Computer readable media suitable for storing computer program instructions and data include all forms of nonvolatile memory, media and memory devices, including by way of example semiconductor memory devices, e.g., EPROM, EEPROM, and flash memory devices. The processor and the memory can be supplemented by, or incorporated in, special purpose logic circuitry.

[00318] It is intended that the specification, together with the drawings, be considered exemplary only, where exemplary means an example. As used herein, the singular forms “a”, “an” and “the” are intended to include the plural forms as well, unless the context clearly indicates otherwise. Additionally, the use of “or” is intended to include “and/or”, unless the context clearly indicates otherwise.

[00319] While this patent document contains many specifics, these should not be construed as limitations on the scope of any invention or of what may be claimed, but rather as descriptions of features that may be specific to particular embodiments of particular inventions. Certain features that are described in this patent document in the context of separate embodiments can also be implemented in combination in a single embodiment. Conversely, various features that are described in the context of a single embodiment can also be implemented in multiple embodiments separately or in any suitable subcombination. Moreover, although features may be described above as acting in certain combinations and even initially claimed as such, one or more features from a claimed combination can in some cases be excised from the combination, and the claimed combination may be directed to a subcombination or variation of a subcombination.

[00320] Similarly, while operations are depicted in the drawings in a particular order, this should not be understood as requiring that such operations be performed in the particular order shown or in sequential order, or that all illustrated operations be performed, to achieve desirable results. Moreover, the separation of various system components in the embodiments described in this patent document should not be understood as requiring such separation in all embodiments.

[00321] Only a few implementations and examples are described and other implementations, enhancements and variations can be made based on what is described and illustrated in this patent document.

CLAIMS

What is claimed is:

1. A video processing method, comprising:
maintaining N tables, indexed by an index $i=1$ to N, wherein an i -th table has a size equal to $M(i)$ corresponding to a maximum number of motion candidates in the i -th table, wherein i , N, and $M(i)$ are integers, and wherein each motion candidate is associated with corresponding motion information; and
performing a conversion between a current video block and a bitstream representation of a video data including the current video block using motion candidates from the tables.
2. The method of claim 1, wherein the N tables include motion information derived from previously processed video blocks in the video data.
3. The method of claim 1, wherein $M(i)$ is same for all tables.
4. The method of claim 1, wherein at least some tables have a size different from that of any one of other tables.
5. The method of claim 1, wherein N and $M(i)$ are predefined.
6. The method of claim 1, wherein sizes of the tables are signaled in one or more of a Video Parameter Set, a Sequence Parameter Set, a Picture Parameter Set, a Slice Header, or a tile header.
7. The method of claim 1, wherein sizes of the tables are signaled in a bit field corresponding to one or more of a Coding Tree Unit or a Coding Tree Block Unit or a Coding Unit or a Prediction Unit.

8. The method of claim 1, wherein at least one of N and M(i) is dependent on at least one of i) a slice type, ii) temporal layer index of a picture, or iii) picture order count (POC) distance between one slice and an intra slice closest to the slice.
9. The method of claim 1, wherein the current video block corresponds to a slice of video data, and wherein N is equal to $K \cdot P$, wherein K is an integer representing tables per decoding thread, and P is an integer representing a number of Largest Coding Unit (LCU) rows or a number of tiles in the slice of video data.
10. The method of claim 1, wherein the current video block corresponds to a slice of video data, and wherein N is equal to P when each LCU row uses a single table and P is an integer representing a number of Largest Coding Unit rows or a number of tiles in the slice of video data.
11. The method of claim 1, wherein at least one of N or M(i) is based on a sequence resolution, a largest coding unit (LCU) size, or a size of a merge or Advanced Motion Vector Prediction (AMVP) candidate list.
12. The method of claim 1, wherein a size of a table is not signaled.
13. The method of claim 1, wherein a size of a table is same as the maximum number of merge candidates in a merge candidate list.
14. The method of claim 1, wherein a size of a table is same as one obtained by subtracting a certain value from the maximum number of merge candidates in a merge candidate list.
15. The method of claim 14, wherein the certain value represents how many spatial or temporal neighboring blocks are accessed during a construction of a merge candidate list.
16. A method for video processing, comprising:

receiving a bitstream representation of a video data including a current video block; and
processing the bitstream representation based on N tables, indexed by an index $i=1$ to N ,
wherein an i -th table has a size equal to $M(i)$ corresponding to a maximum number of motion
candidates in the i -th table, wherein i , N , and $M(i)$ are integers, and wherein each motion
candidate is associated with corresponding motion information.

17. The method of claim 16, wherein the N tables include motion information derived
from previously processed video blocks in the video data.

18. The method of claim 16, further comprising: updating one or more tables based
on motion information from video blocks in the video data.

19. The method of claim 16, wherein each motion candidate is associated with
corresponding motion information.

20. The method of claim 16, wherein at least one of N or $M(i)$ is based on a sequence
resolution, a largest coding unit (LCU) size, or a size of a merge or Advanced Motion Vector
Prediction (AMVP) candidate list.

21. The method of claim 16, wherein a size of a table is not signaled.

22. The method of claim 16, wherein a size of a table is equal to the maximum
number of merge candidates in a merge candidate list.

23. The method of claim 16, wherein a size of a table is obtained by subtracting a
fixed value from the maximum number of merge candidates in a merge candidate list.

24. The method of claim 23, wherein the fixed value is based on the number of
temporal or spatial neighboring blocks that are accessed during a construction of a merge
candidate list or AMVP candidate list.

25. The method of claim 16, wherein $M(i)$ is same for all tables.

26. The method of claim 16, wherein at least some tables have a size different from that of any one of other tables.
27. The method of claim 16, wherein N and $M(i)$ are predefined.
28. The method of claim 16, wherein sizes of the tables are signaled in one or more of a Video Parameter Set, a Sequence Parameter Set, a Picture Parameter Set, a Slice Header, or a tile header.
29. The method of claim 16, wherein sizes of the tables are signaled in a bit field corresponding to one or more of a Coding Tree Unit, a Coding Tree Block Unit, a Coding Unit, or a Prediction Unit.
30. The method of claim 16, wherein at least one of N or $M(i)$ is dependent on at least one of i) a slice type, ii) temporal layer index of a picture, or iii) picture order count (POC) distance between one slice and an intra slice closest to the slice.
31. The method of claim 16, wherein the current video block corresponds to a slice of video data, and wherein N is equal to $K \cdot P$, wherein K is an integer representing tables per decoding thread, and P is an integer representing a number of Largest Coding Unit rows or a number of tiles in the slice of video data.
32. The method of claim 16, wherein the current video block corresponds to a slice of video data, and wherein N is equal to P when each LCU row uses a single table and P is an integer representing a number of Largest Coding Unit rows or a number of tiles in the slice of video data.
33. The method of any one of claims 1-32, wherein the performing of the conversion includes generating the bitstream representation from the current video block.

34. The method of any one of claims 1-32, wherein the performing of the conversion includes generating the current video block from the bitstream representation.

35. The method of any one of claims 1 to 34, wherein a motion candidate is associated with motion information including at least one of: a prediction direction, a reference picture index, motion vector values, an intensity compensation flag, an affine flag, a motion vector difference precision, or motion vector difference value.

36. The method of any one of claims 16-32, further comprising updating, based on the conversion, one or multiple tables.

37. The method of any one of claims 1-36, wherein the updating of one or more tables includes updating one or more tables based on the motion information of the current video block after performing the conversion.

38. The method of claim 37, further comprising: performing a conversion between a subsequent video block of the video data and the bitstream representation of the video data based on the updated tables.

39. An apparatus in a video system comprising a processor and a non-transitory memory with instructions thereon, wherein the instructions upon execution by the processor, cause the processor to implement the method recited in one or more of claims 1-38.

40. A computer program product stored on a non-transitory computer readable media, the computer program product including program code for carrying out the method recited in one or more of claims 1-38.

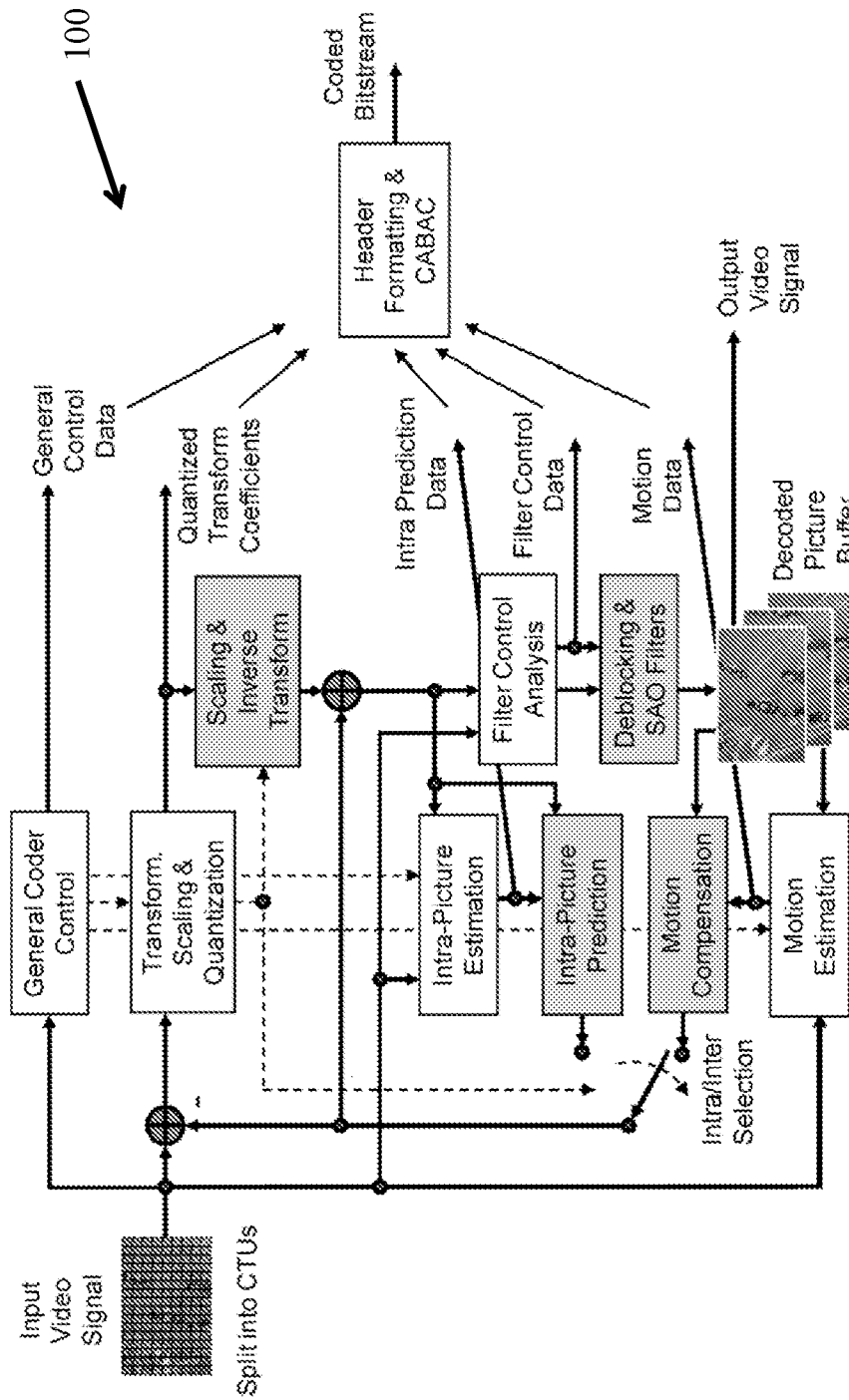


FIG. 1

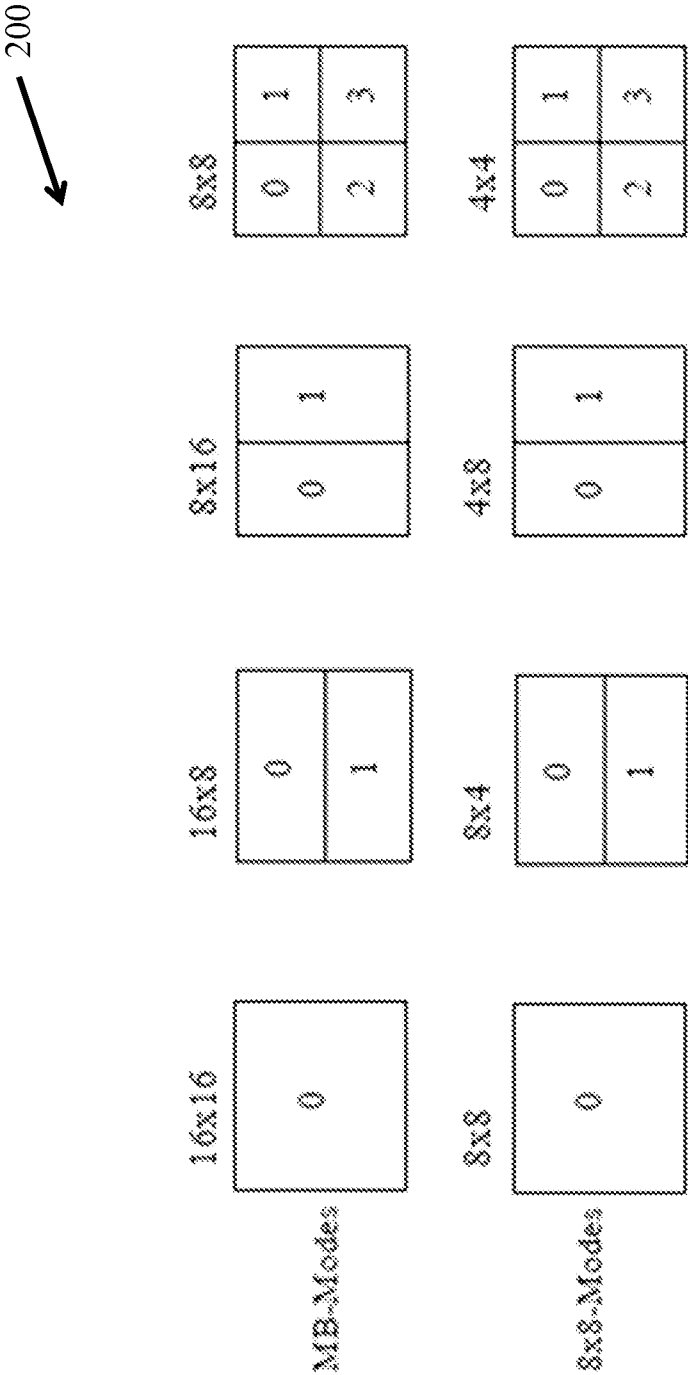


FIG. 2

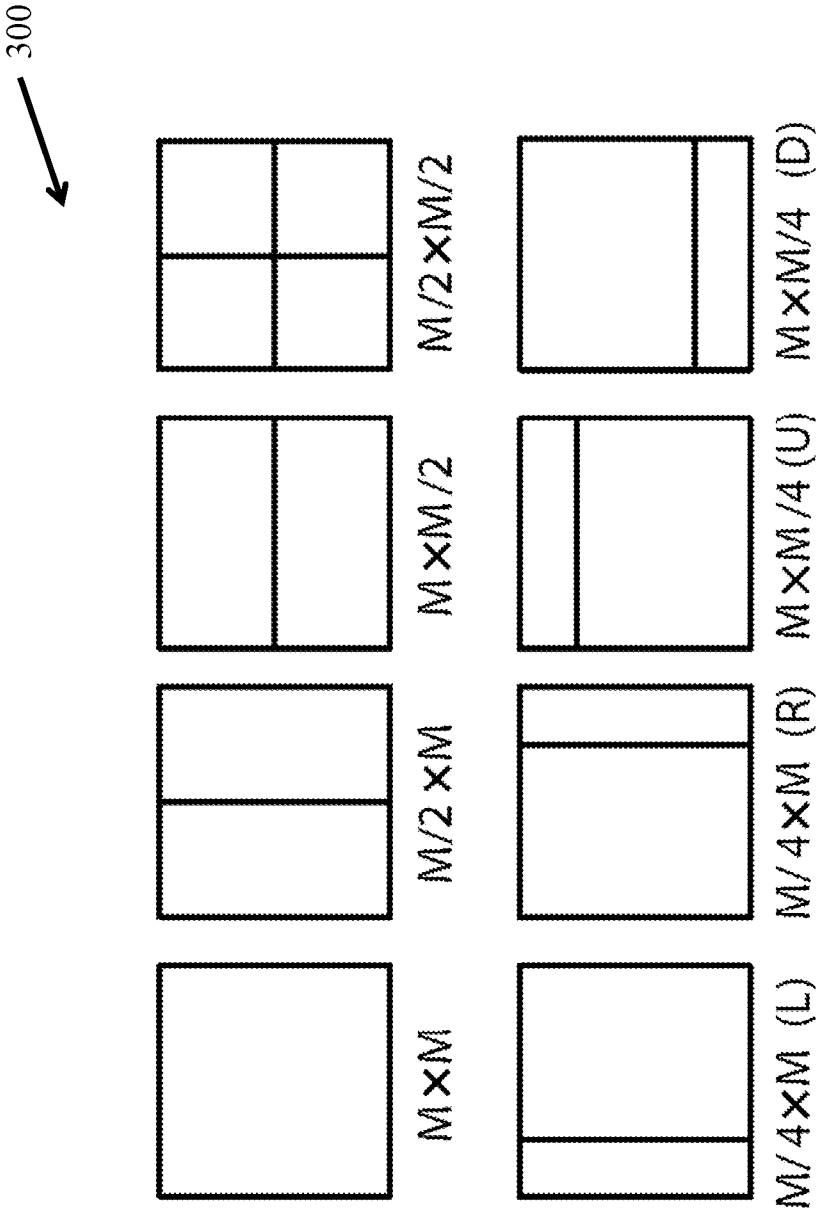


FIG. 3

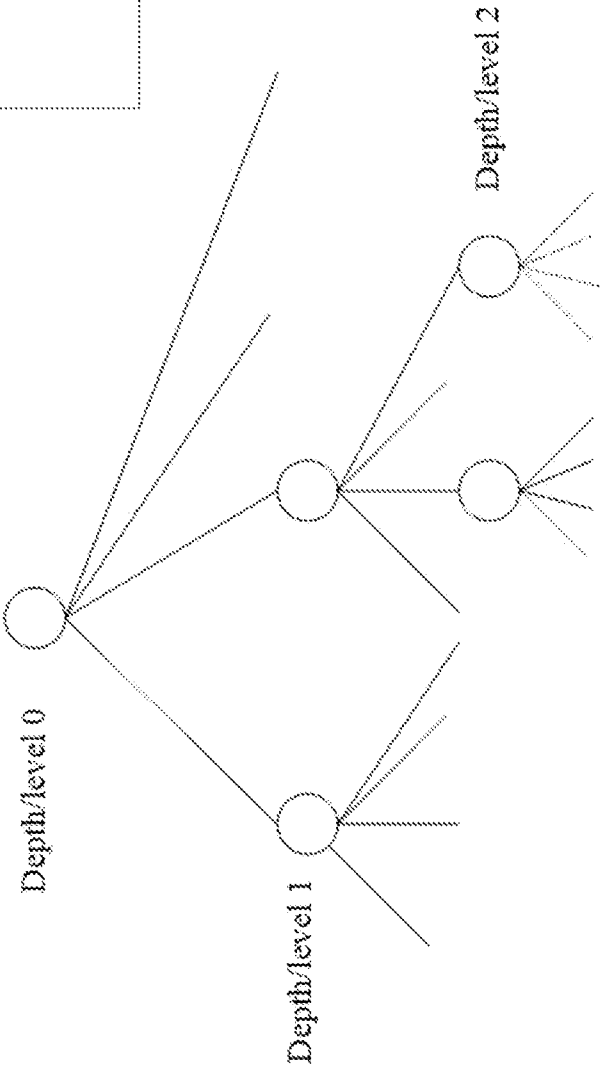
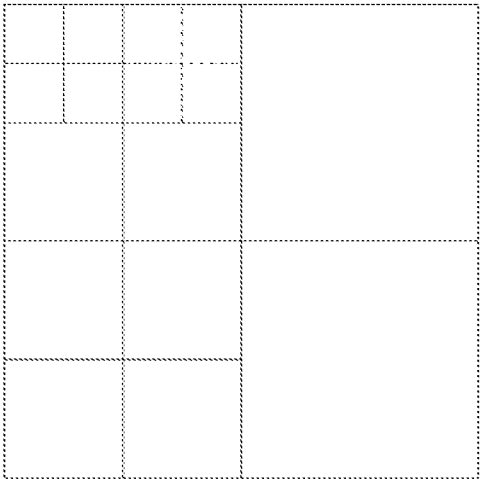


FIG. 5A

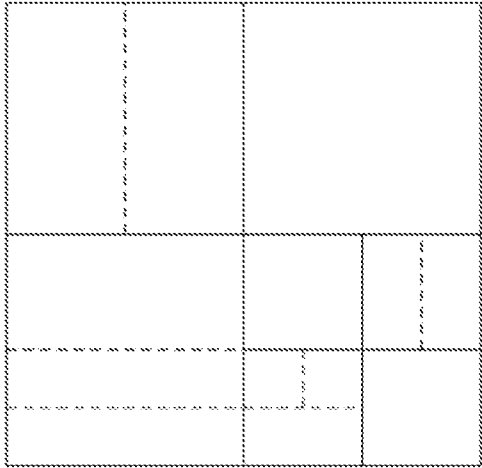
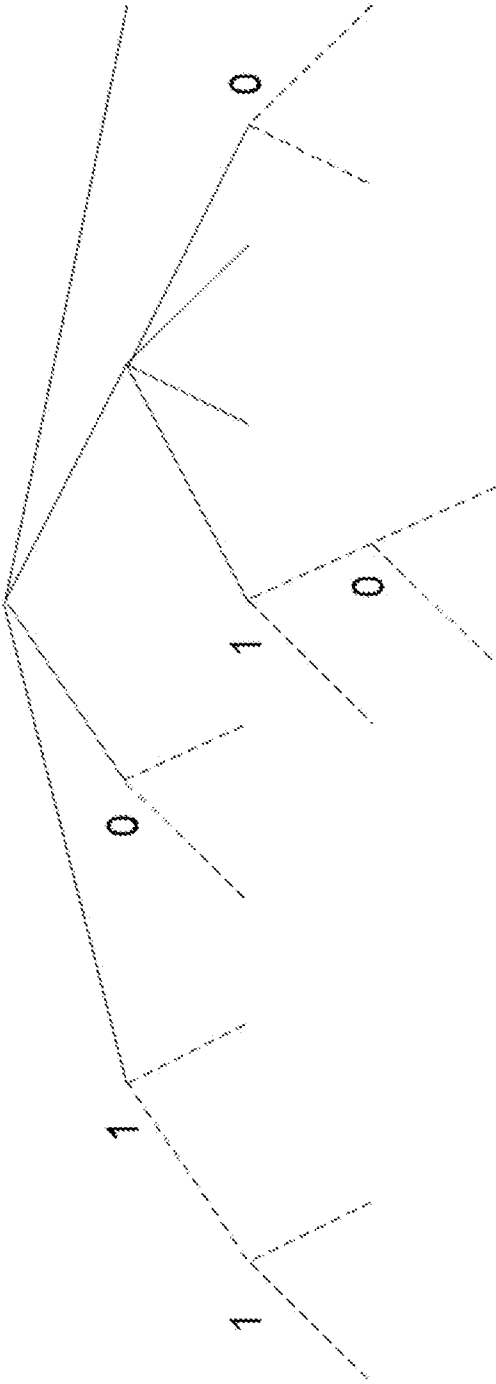


FIG. 5B



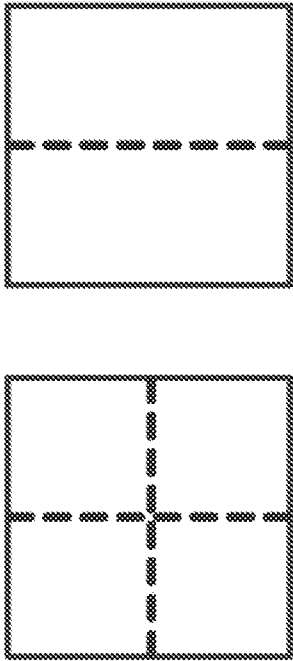


FIG. 6A

FIG. 6B

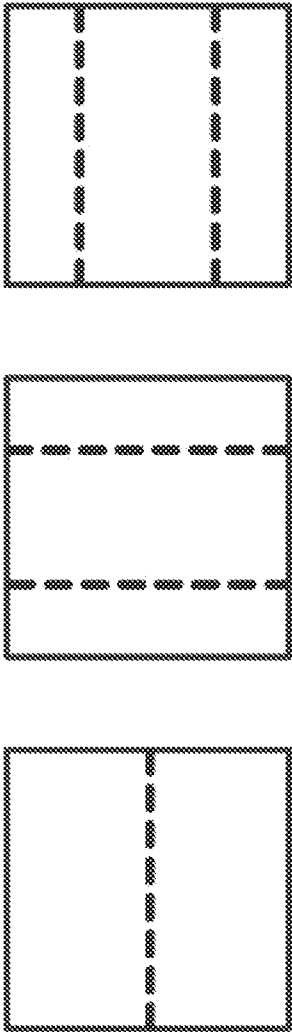


FIG. 6C

FIG. 6D

FIG. 6E

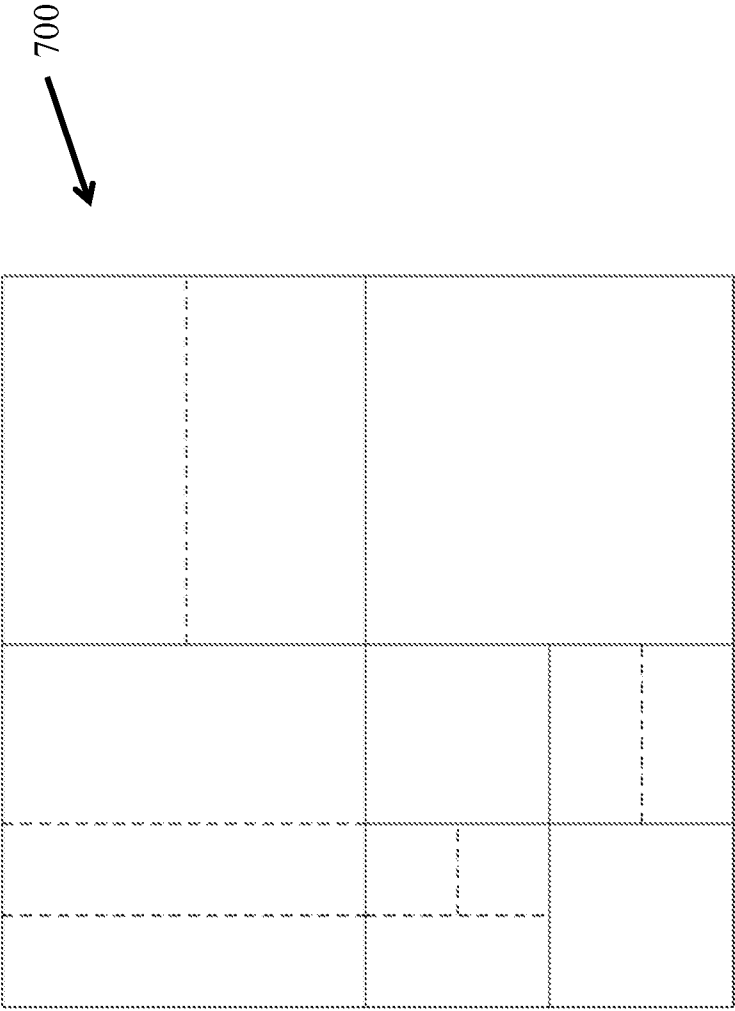


FIG. 7

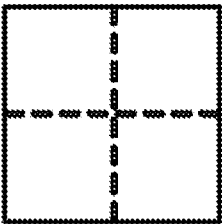


FIG. 8A

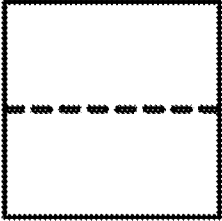


FIG. 8B

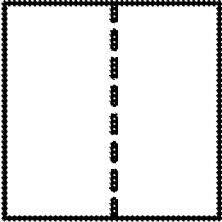


FIG. 8C

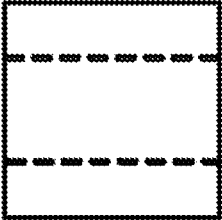


FIG. 8D

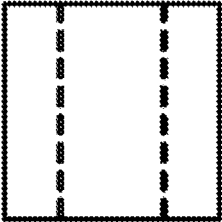


FIG. 8E



FIG. 8F

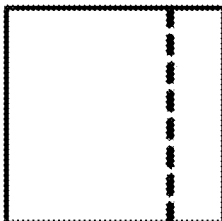


FIG. 8G

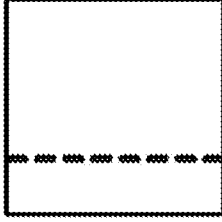


FIG. 8H

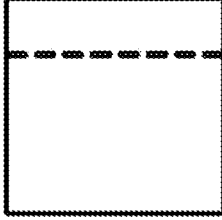


FIG. 8I

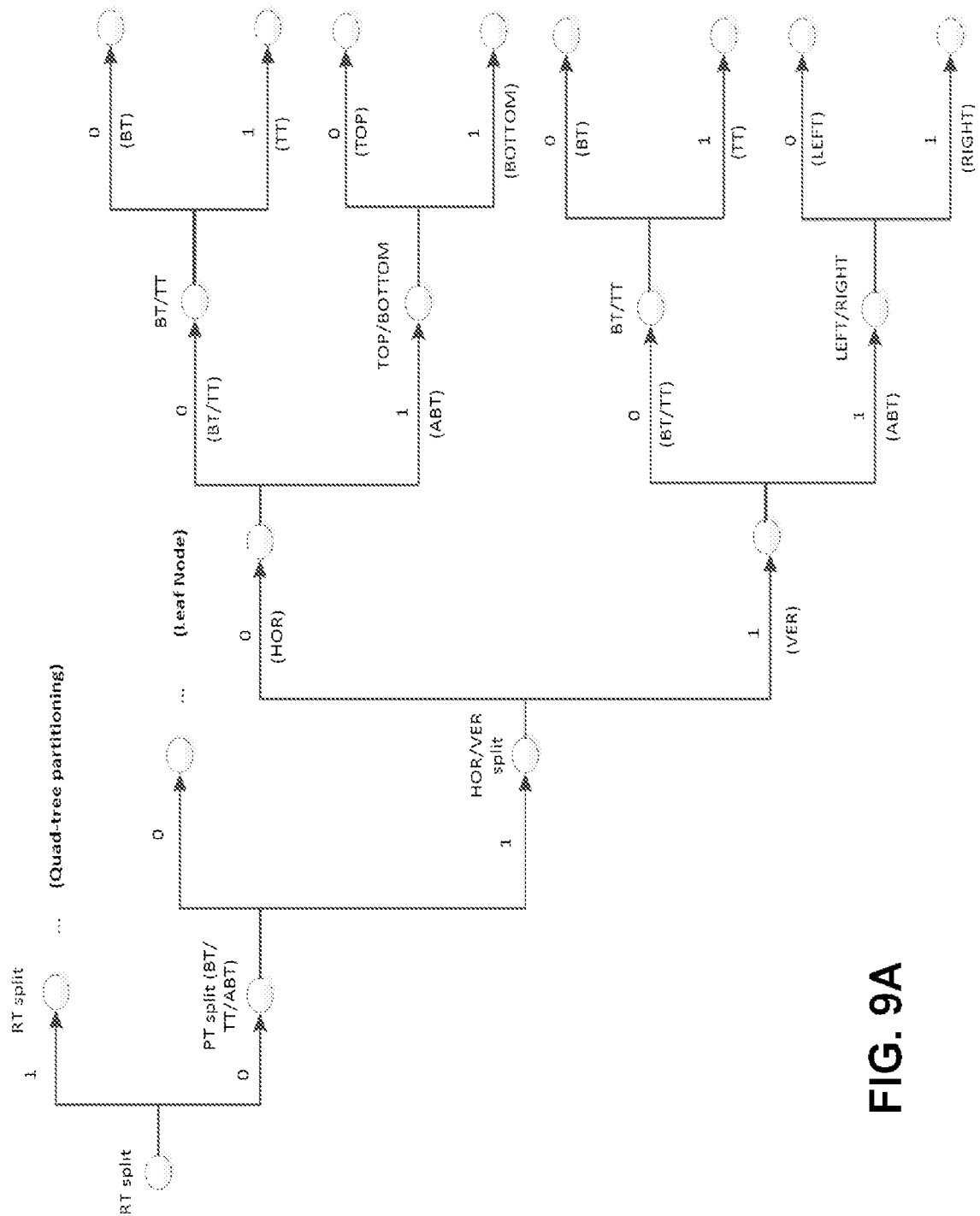


FIG. 9A

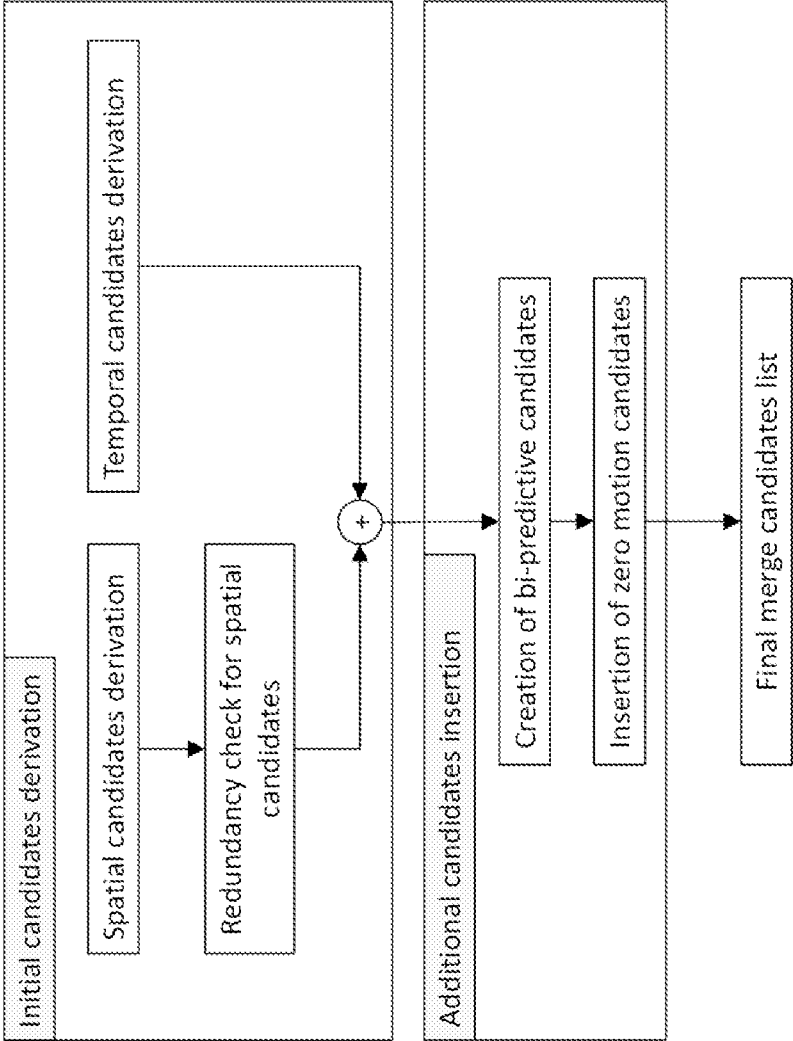


FIG. 9B

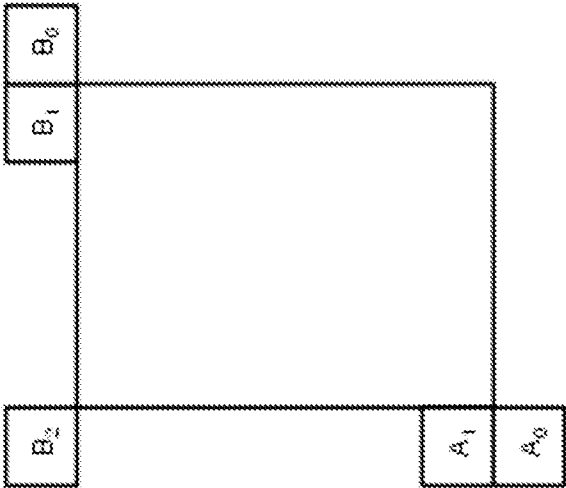


FIG. 10

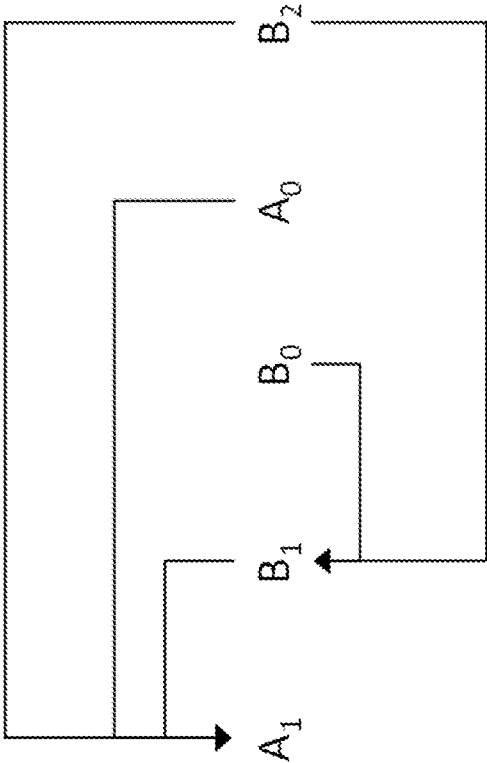


FIG. 11

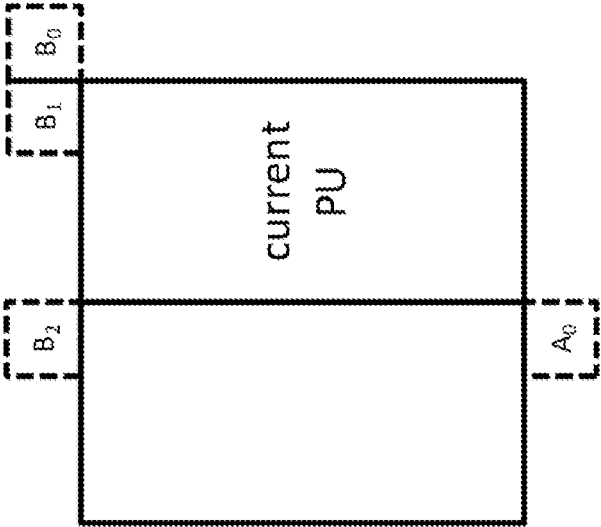


FIG. 12A

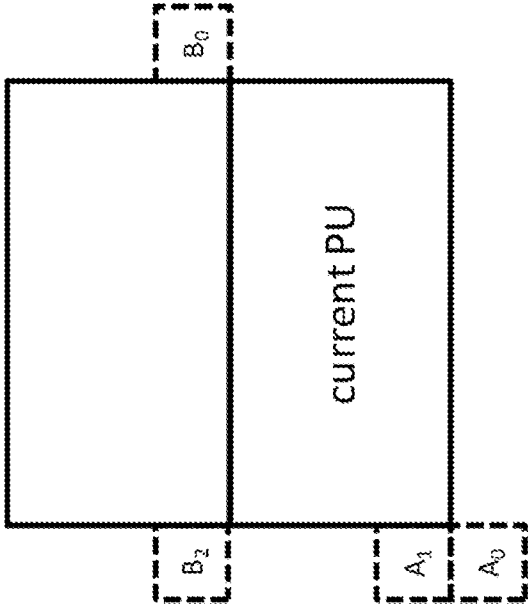


FIG. 12B

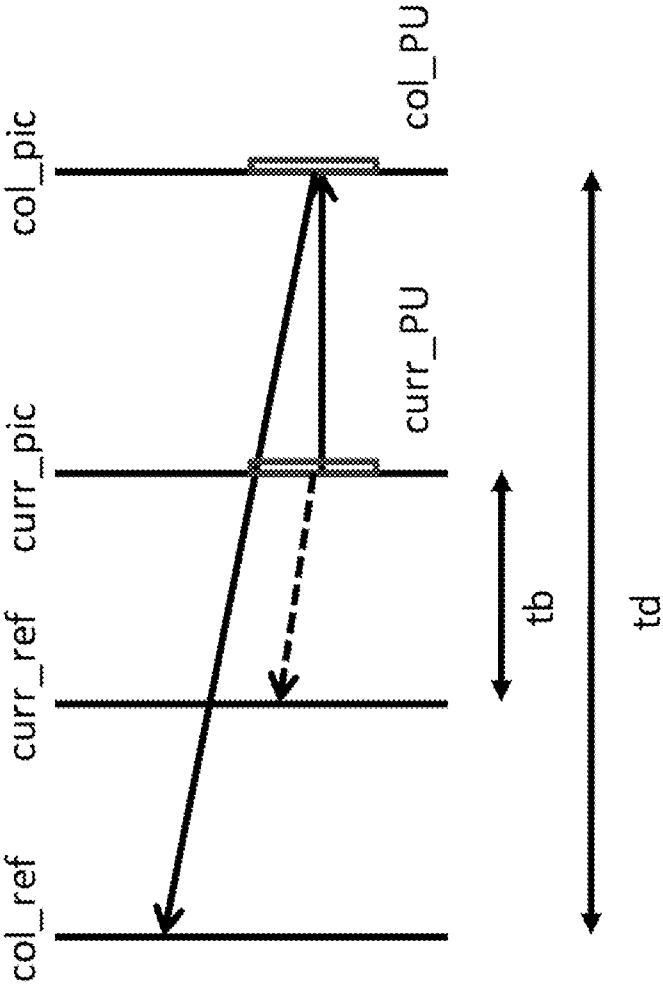


FIG. 13

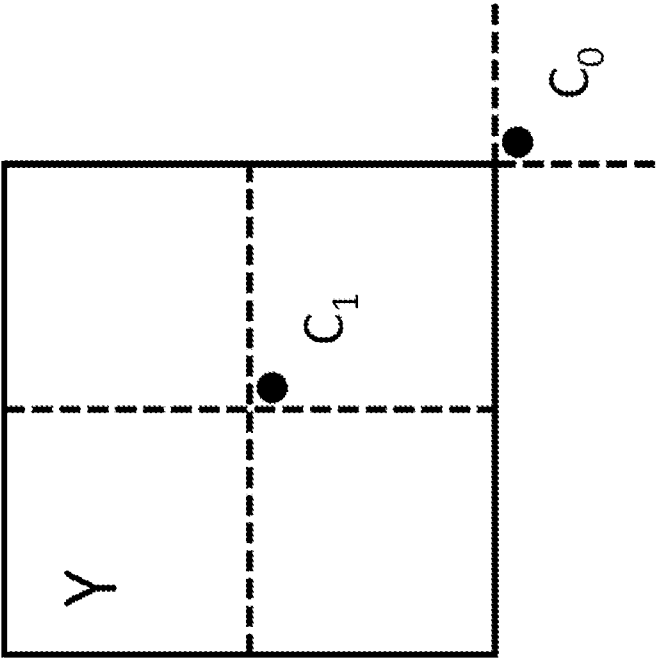


FIG. 14

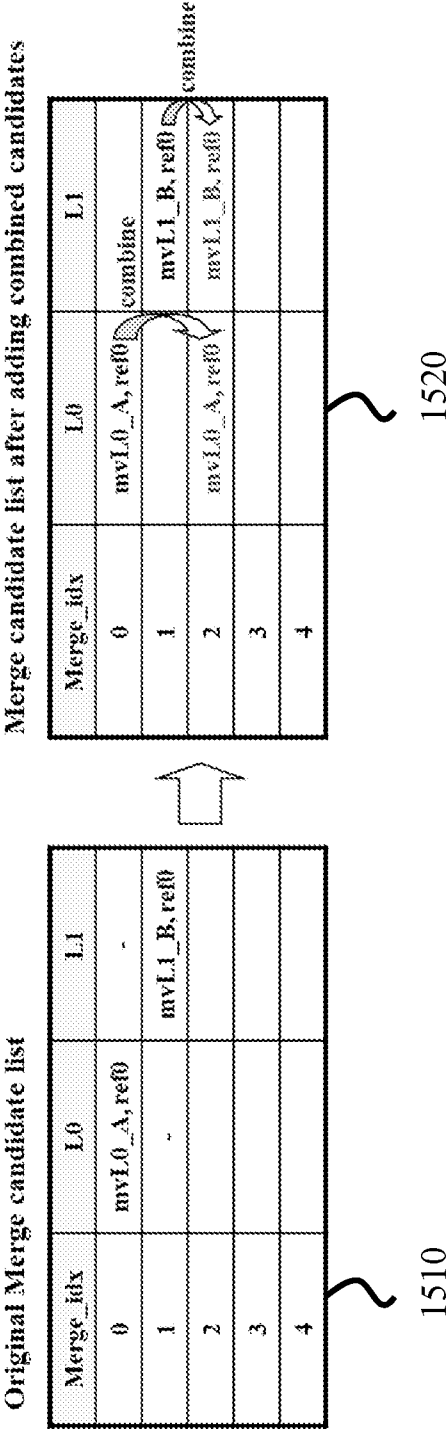


FIG. 15

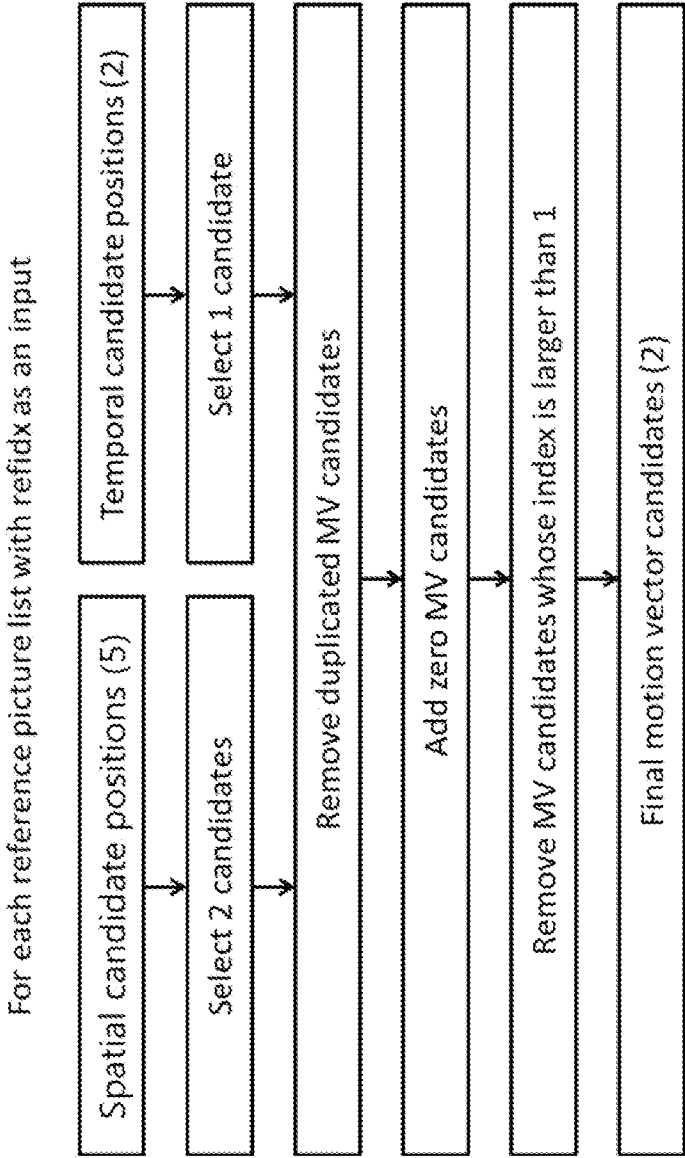


FIG. 16A

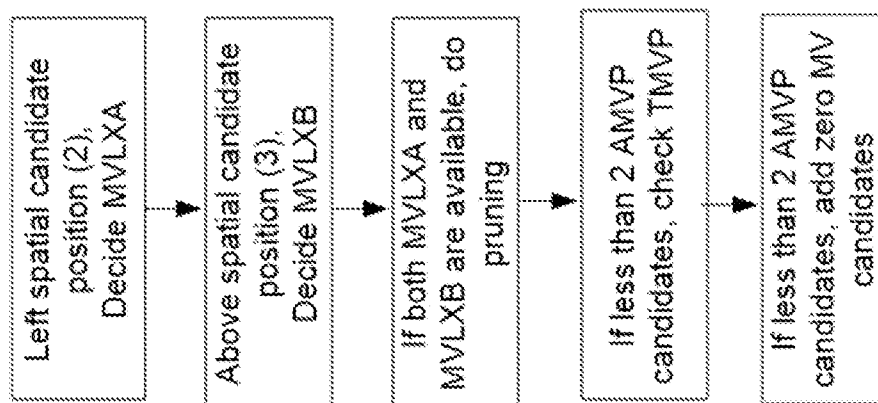


FIG. 16B

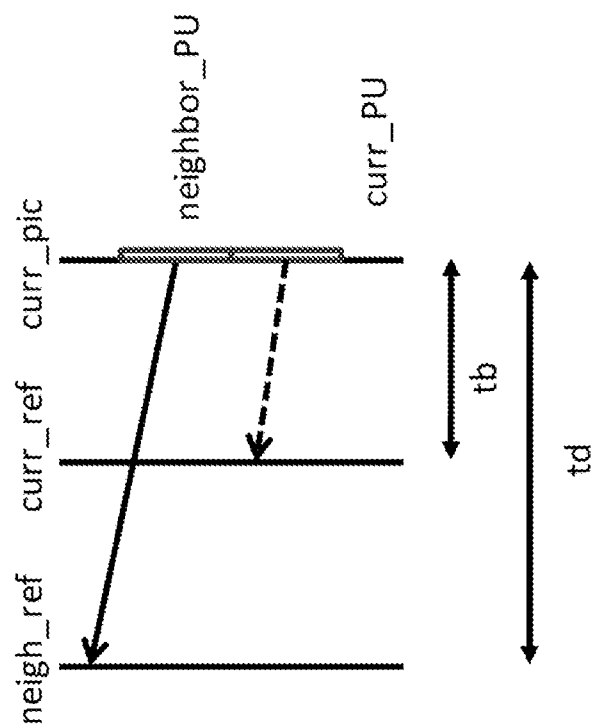


FIG. 17

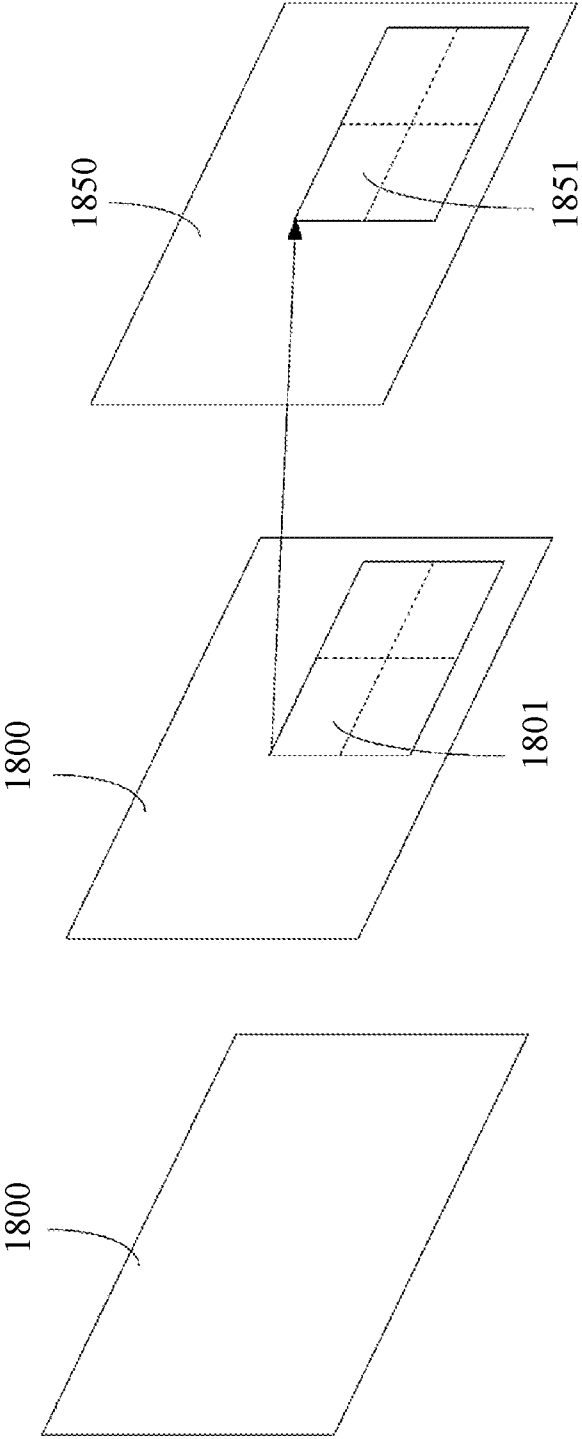


FIG. 18

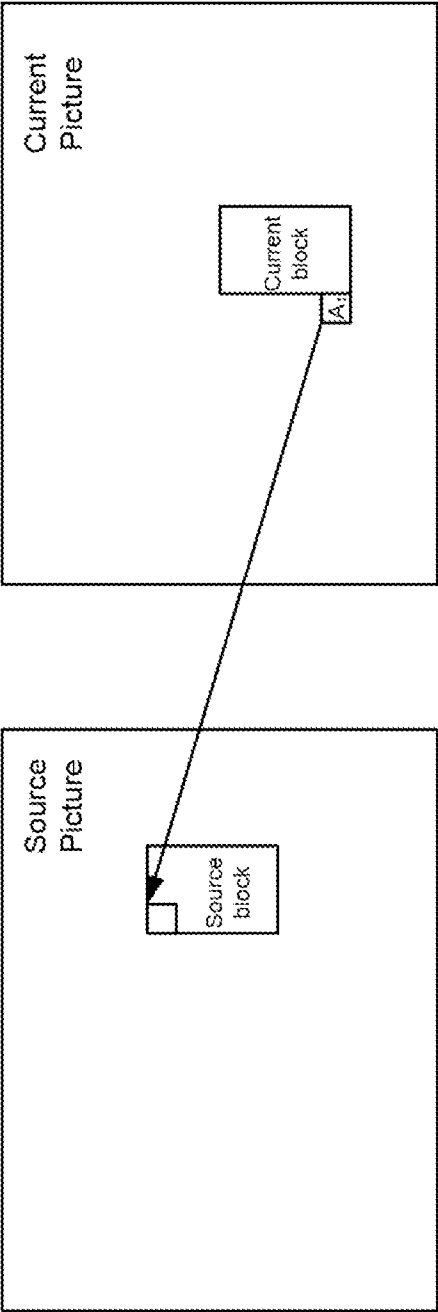


FIG. 19

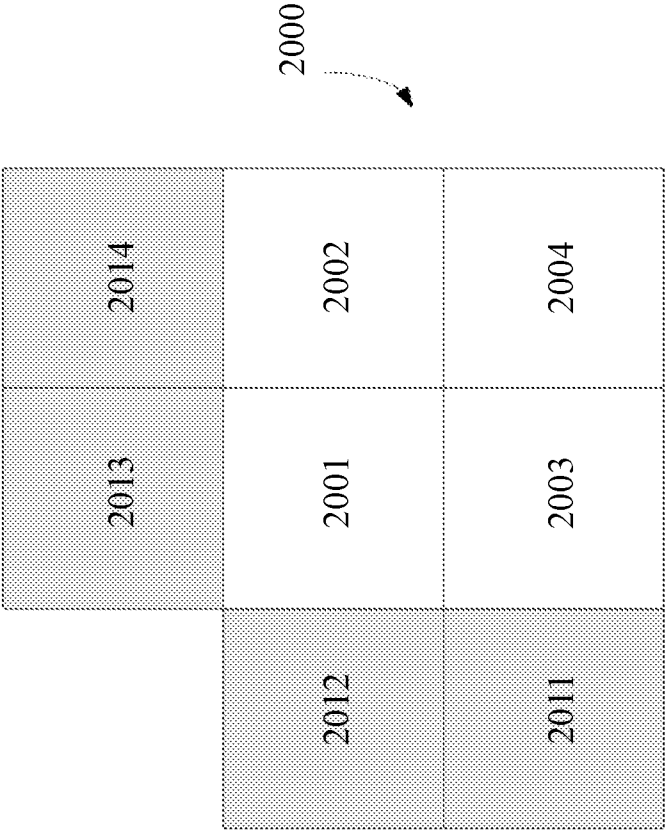


FIG. 20

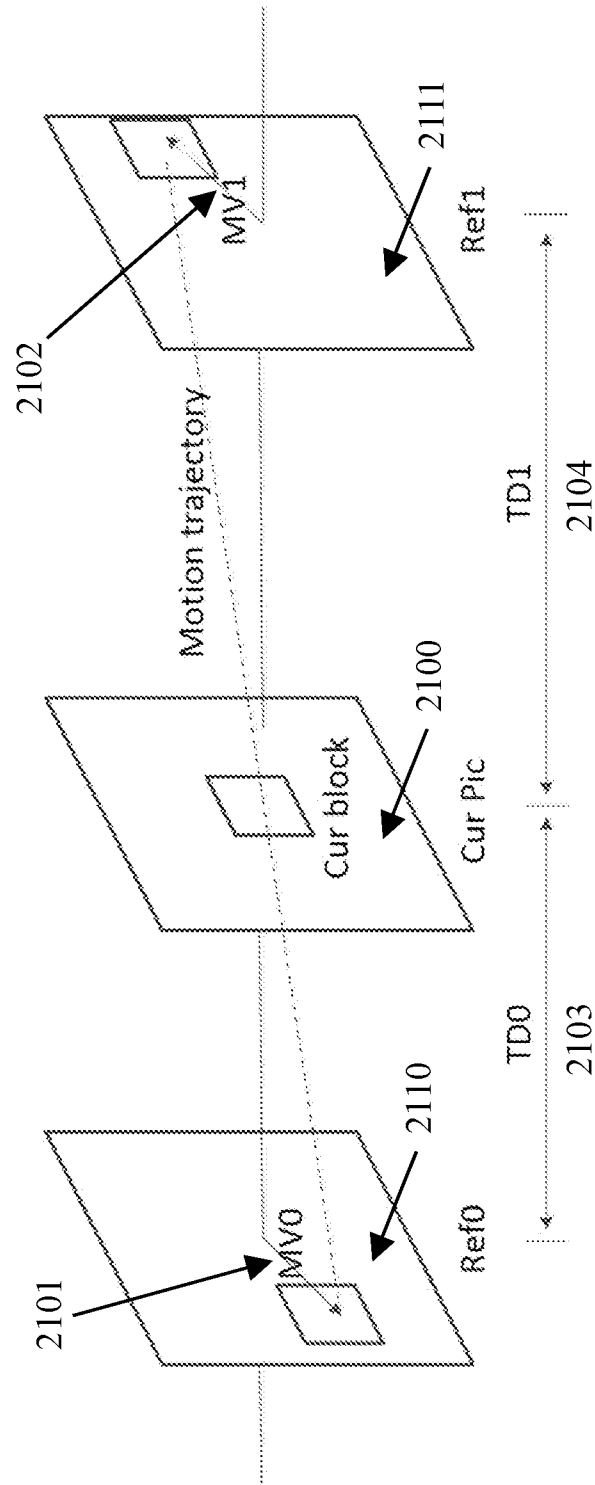


FIG. 21

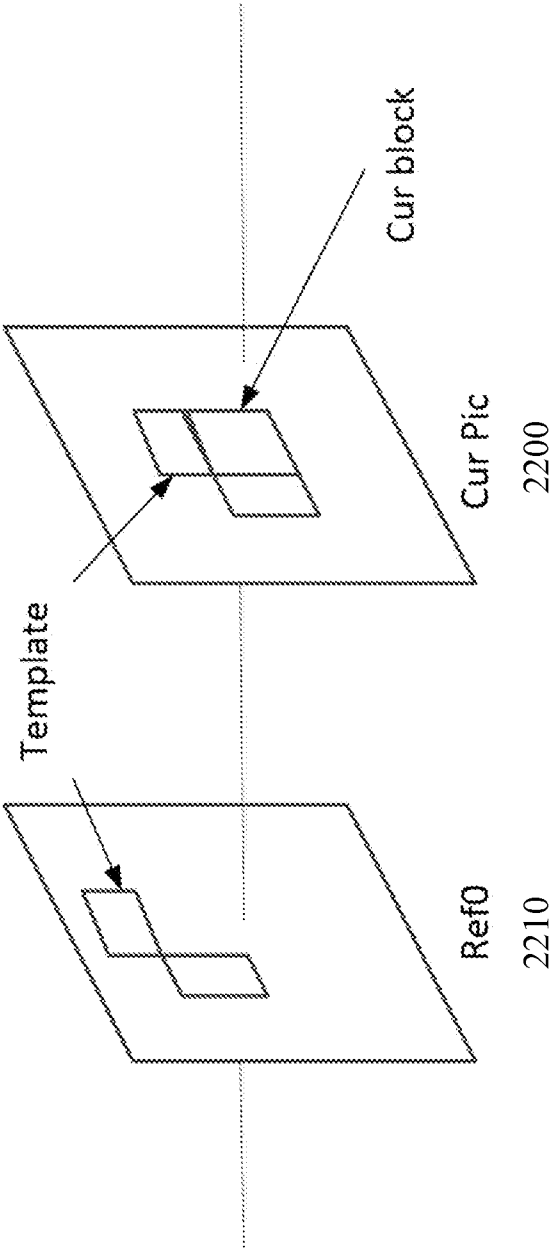


FIG. 22

2300

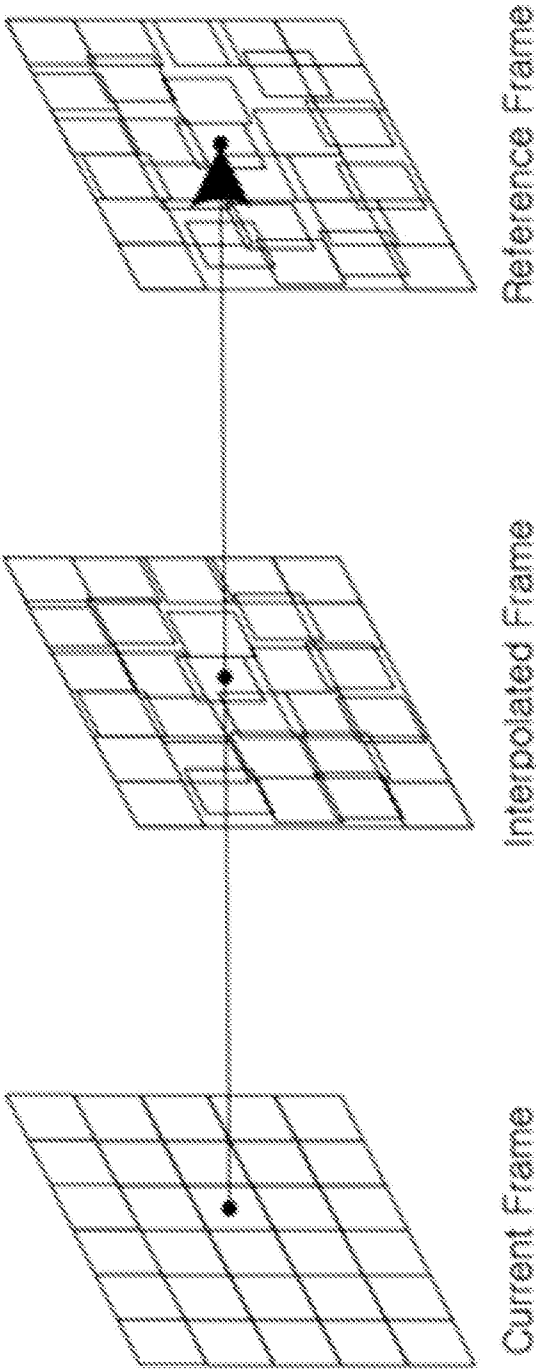


FIG. 23

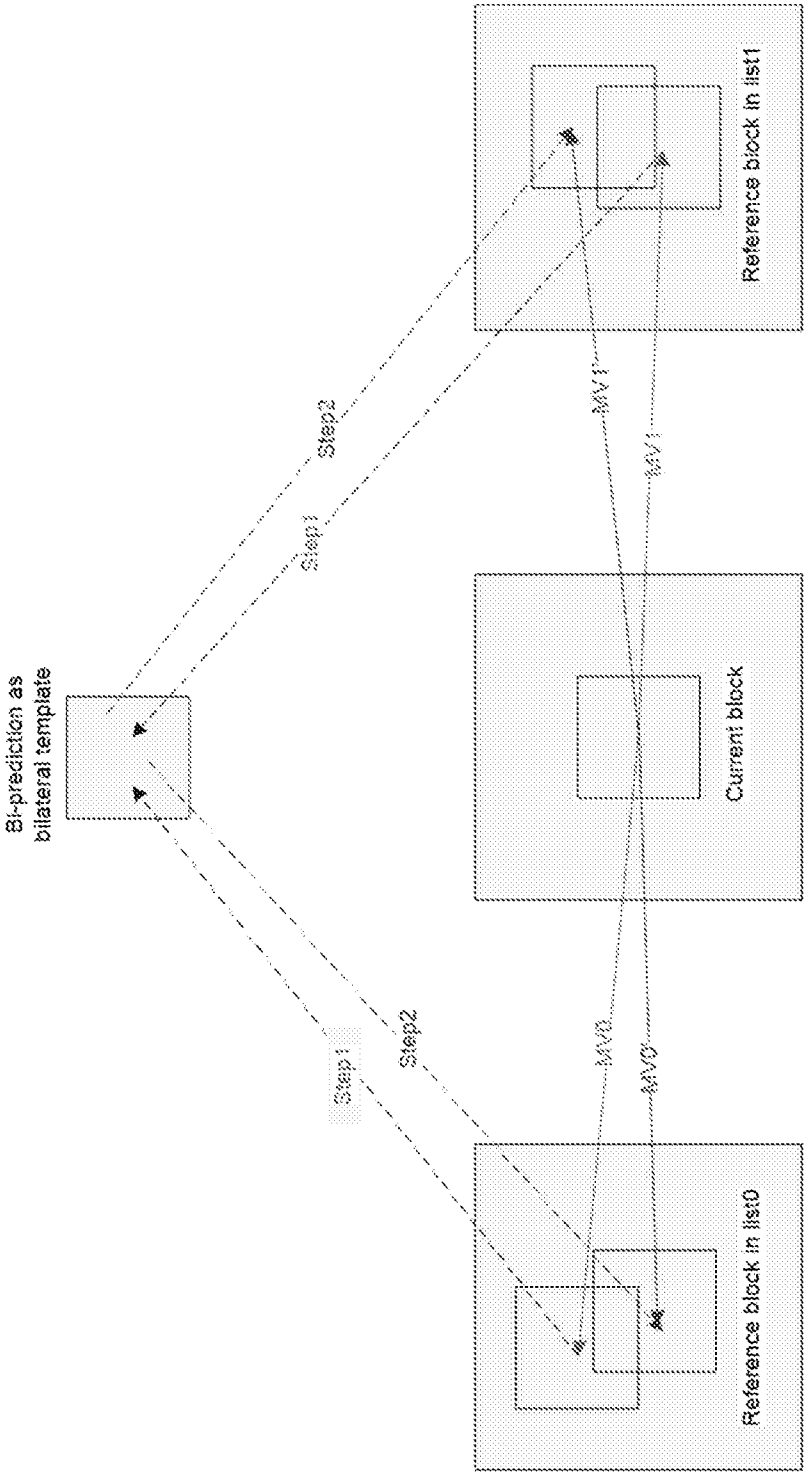


FIG. 24

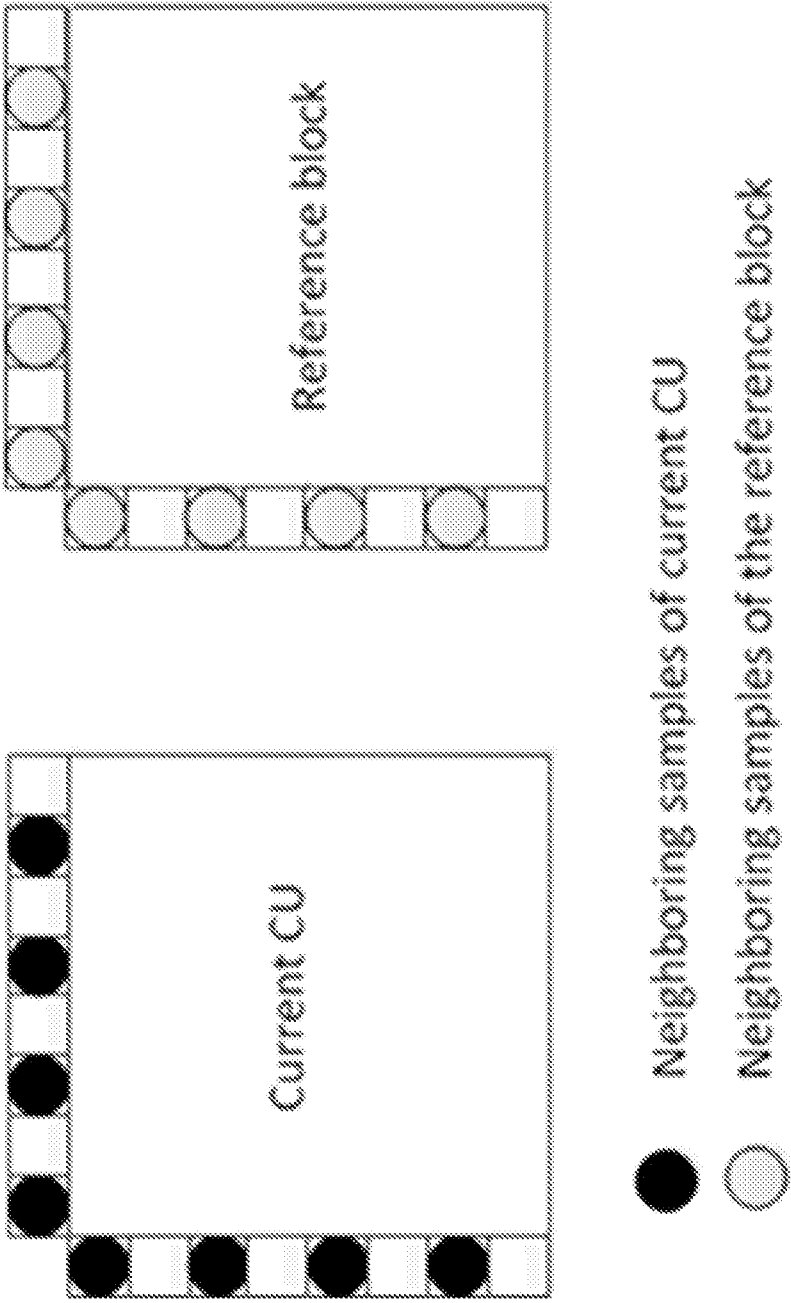


FIG. 25

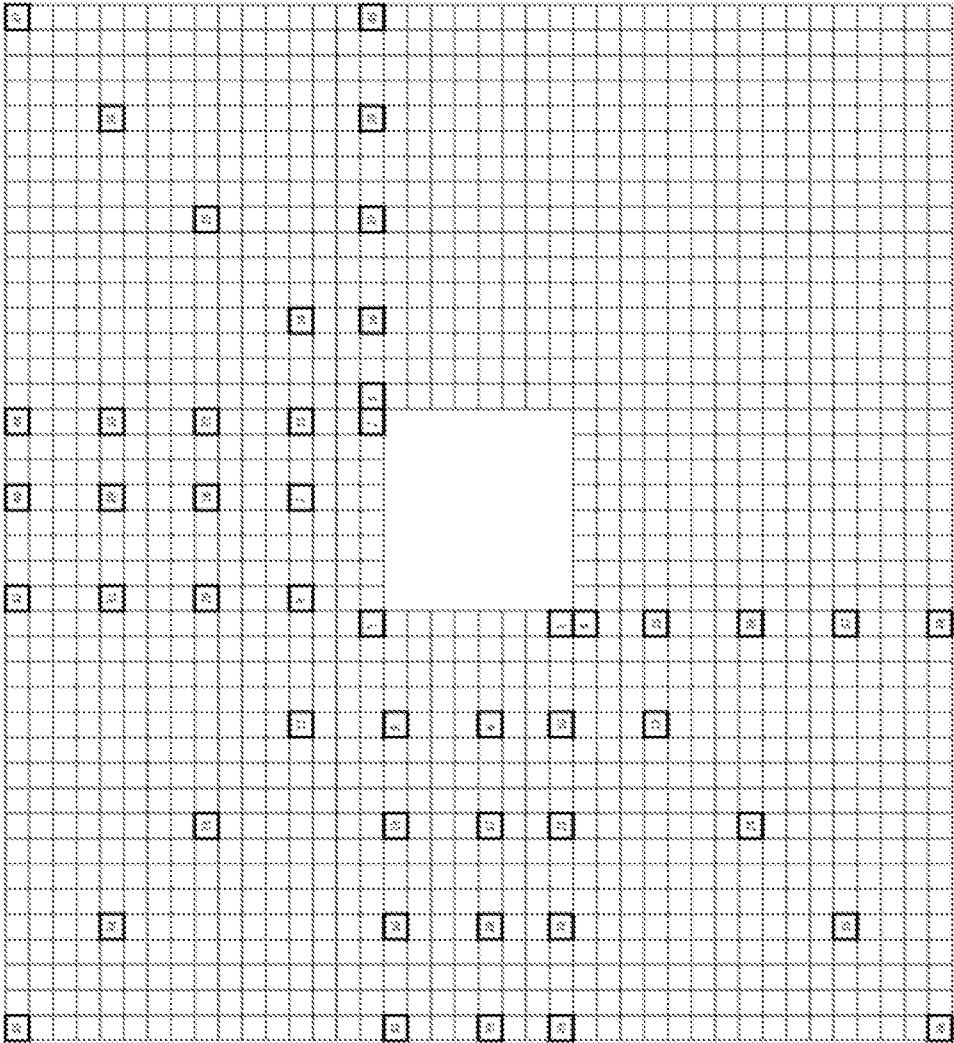


FIG. 26

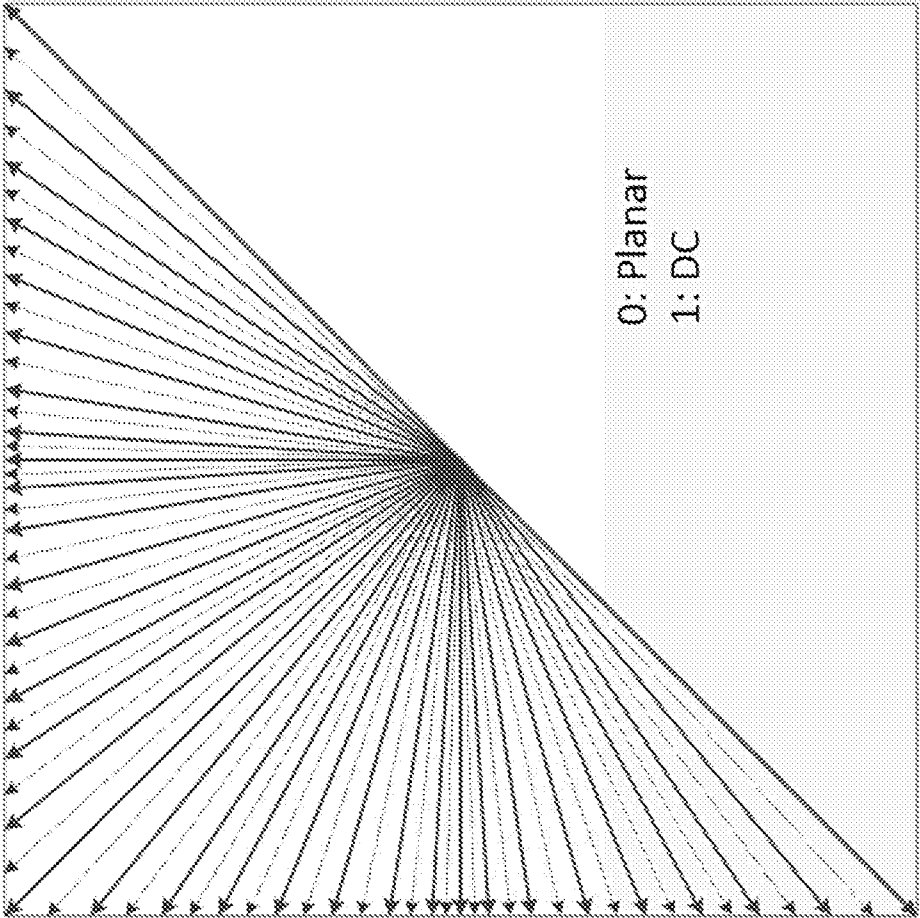


FIG. 27

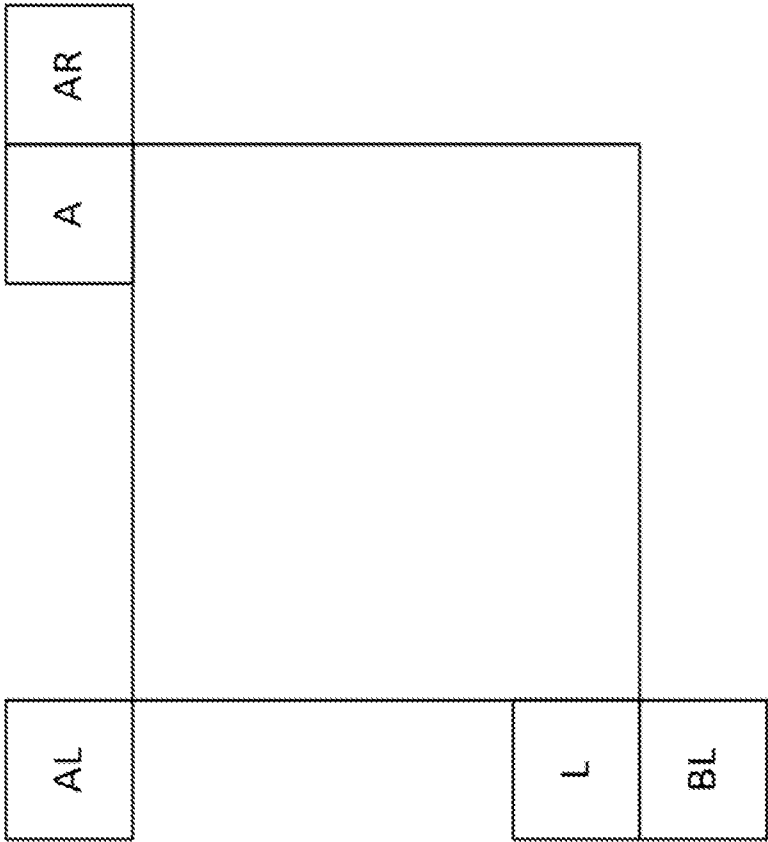


FIG. 28

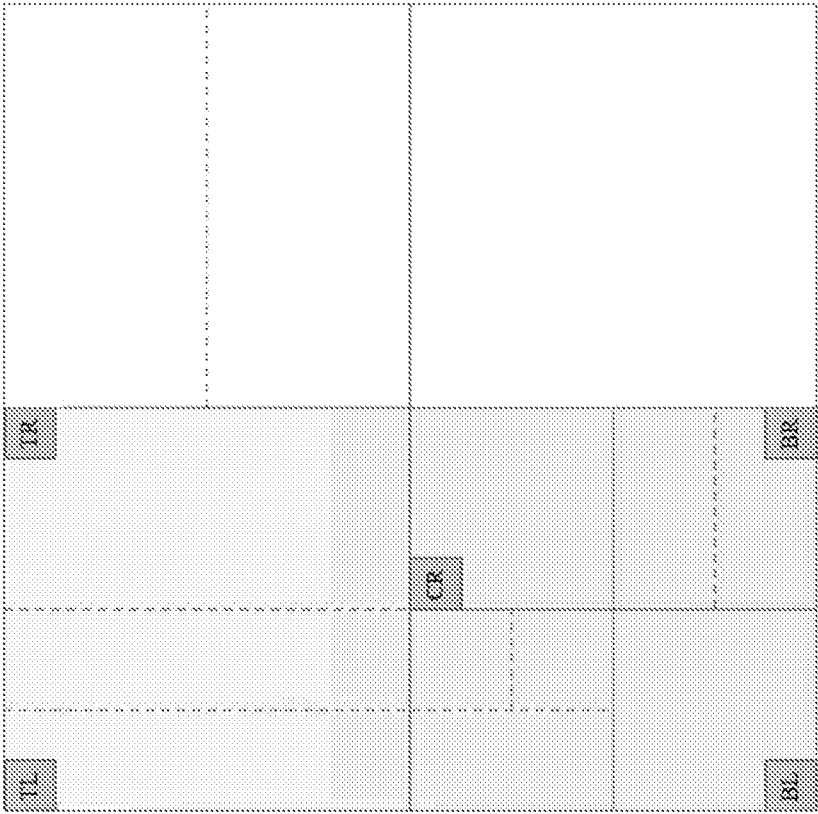


FIG. 29A

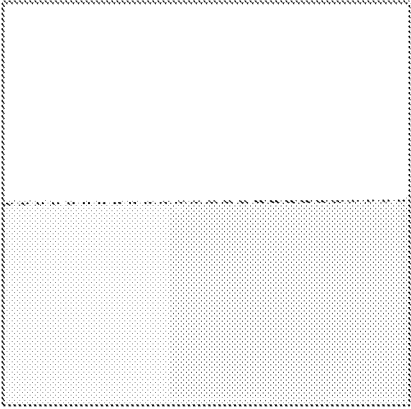


FIG. 29B

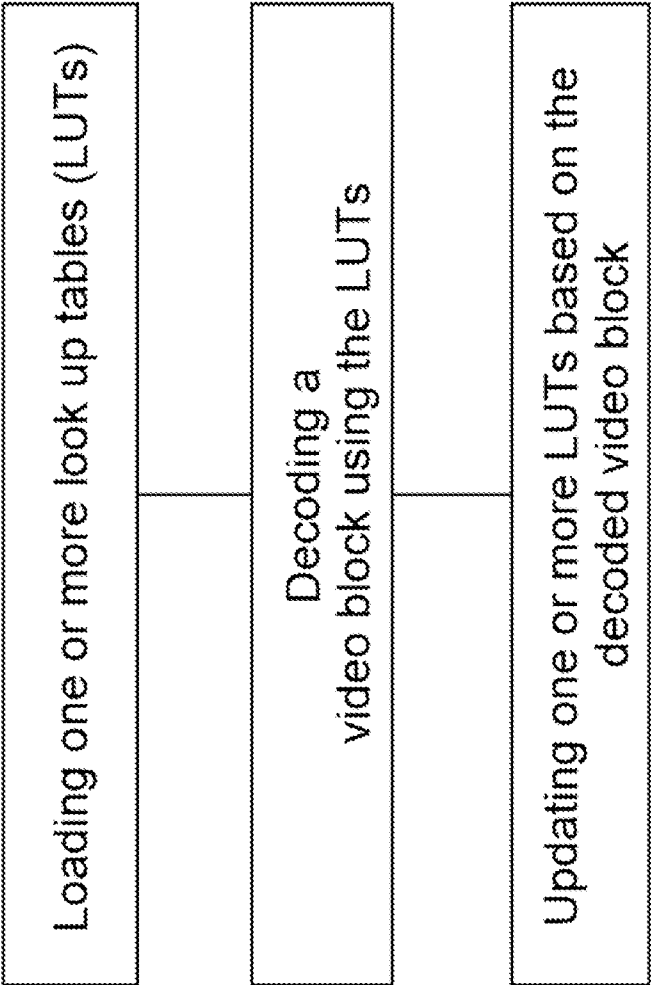


FIG. 30

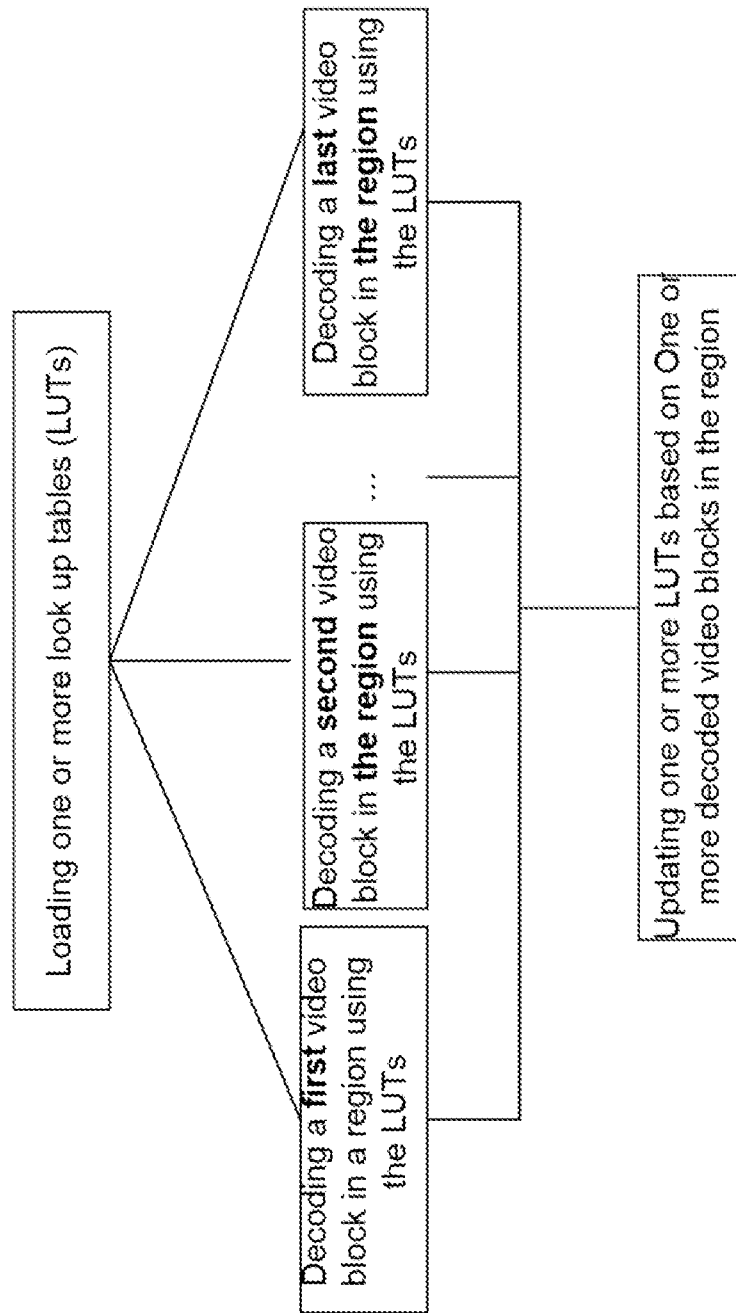


FIG. 31

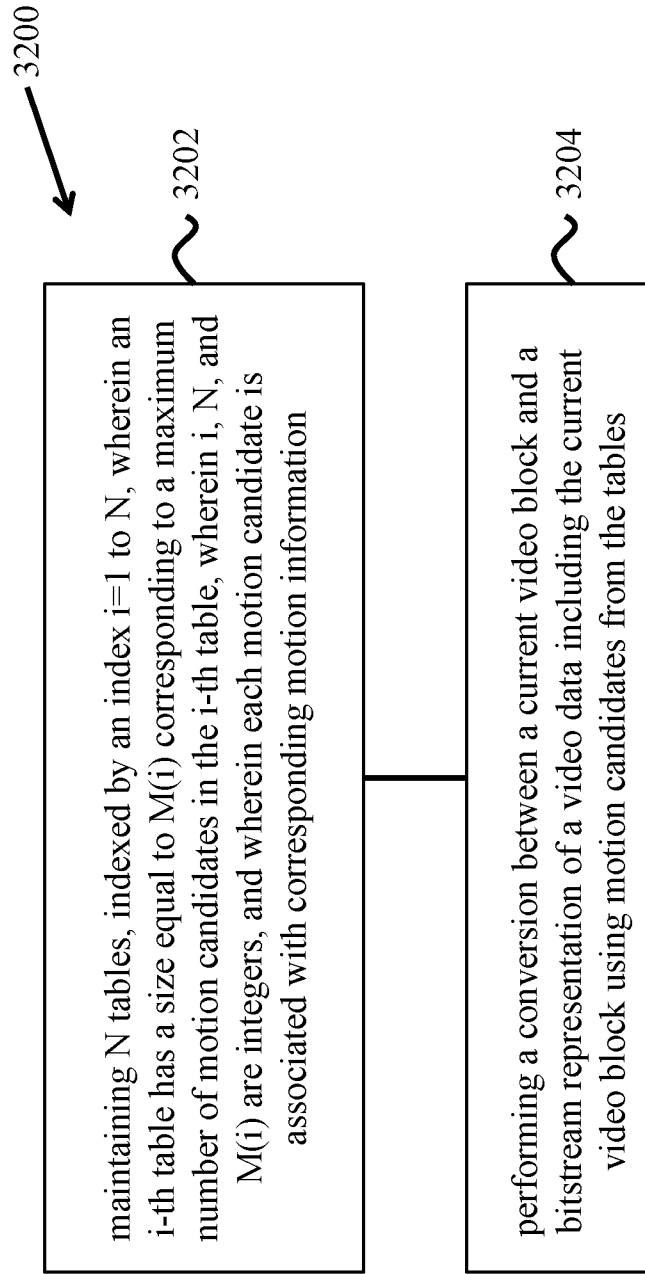


FIG. 32A

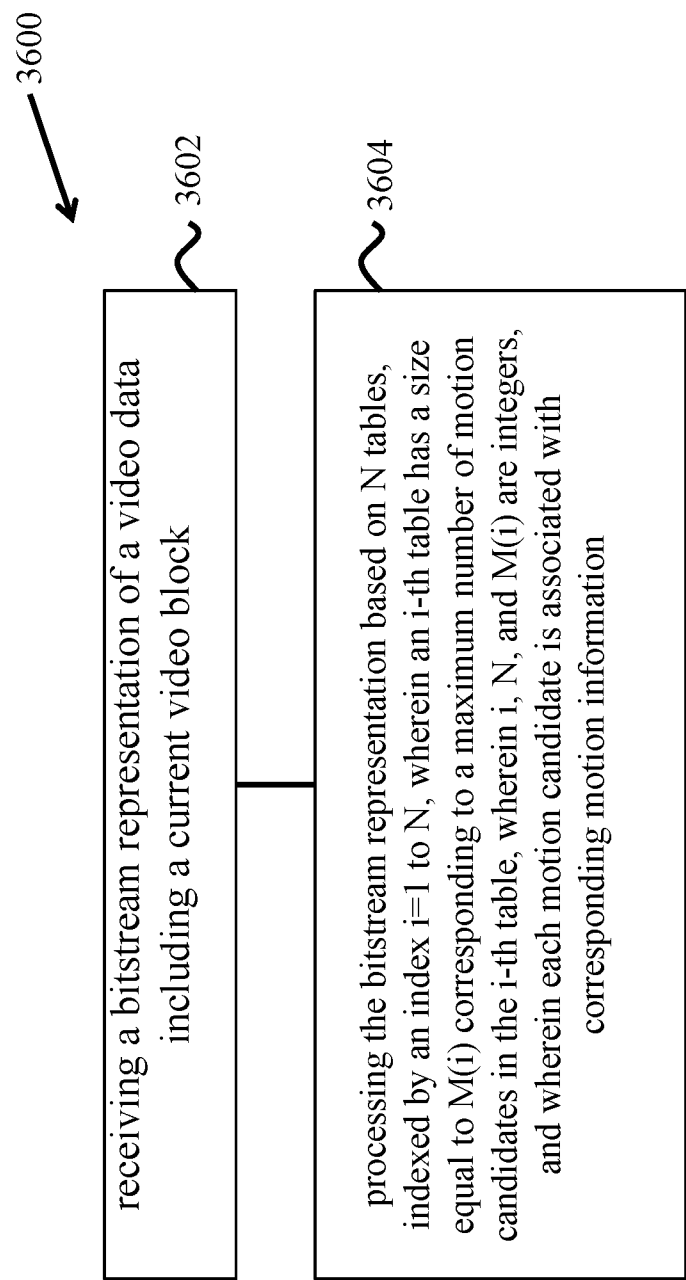


FIG. 32B

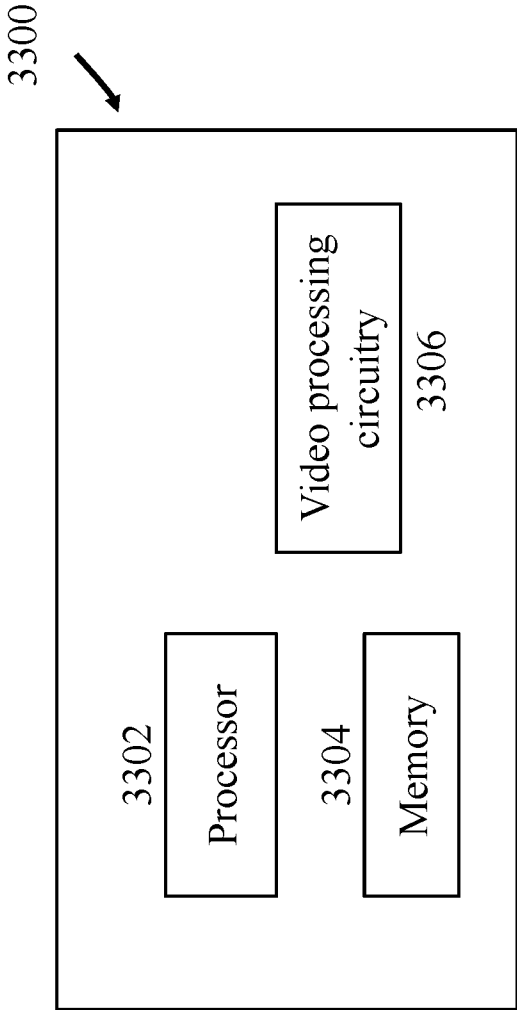


FIG. 33

INTERNATIONAL SEARCH REPORT

International application No

PCT/IB2019/055591

A. CLASSIFICATION OF SUBJECT MATTER

INV. H04N19/52 H04N19/70
ADD.

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

B. FIELDS SEARCHED

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

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

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

EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2013/188715 A1 (SEREGIN VADIM [US] ET AL) 25 July 2013 (2013-07-25) paragraphs [0127], [0129], [0130]; figures 13,15	1-40
X	EP 2 532 160 A1 (ERICSSON TELEFON AB L M [SE]) 12 December 2012 (2012-12-12) paragraphs [0014], [0021], [0022], [0067], [0071], [0074]; figure 4 ----- -/-	1,16,39, 40

☒ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents :

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

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

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

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

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

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

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

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

"&" document member of the same patent family

Date of the actual completion of the international search

17 September 2019

Date of mailing of the international search report

01/10/2019

Name and mailing address of the ISA/

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

Authorized officer

Oelbaum, Tobias

INTERNATIONAL SEARCH REPORT

International application No

PCT/IB2019/055591

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X,P	ZHANG (BYTEDANCE) L ET AL: "CE4-related: History-based Motion Vector Prediction", 11. JVET MEETING; 20180711 - 20180718; LJUBLJANA; (THE JOINT VIDEO EXPLORATION TEAM OF ISO/IEC JTC1/SC29/WG11 AND ITU-T SG.16), , no. JVET-K0104 18 July 2018 (2018-07-18), XP030200019, Retrieved from the Internet: URL:http://phenix.int-evry.fr/jvet/doc_end_user/documents/11_Ljubljana/wg11/JVET-K0104-v5.zip JVET-K0104_r4.docx [retrieved on 2018-07-18] the whole document -----	1-40

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No

PCT/IB2019/055591

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2013188715 A1	25-07-2013	US 2013188715 A1	25-07-2013
		WO 2013106336 A2	18-07-2013
EP 2532160 A1	12-12-2012	CN 102860006 A	02-01-2013
		EP 2532159 A1	12-12-2012
		EP 2532160 A1	12-12-2012
		US 2011194608 A1	11-08-2011
		US 2011194609 A1	11-08-2011
		US 2017064299 A1	02-03-2017
		WO 2011095259 A1	11-08-2011
		WO 2011095260 A1	11-08-2011