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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
16 July 2020 (16.07.2020)

WIPO | PCT

(10) International Publication Number
WO 2020/143824 A1

(51) International Patent Classification:
H04N 19/82 (2014.01) *H04N 19/176* (2014.01)
H04N 19/117 (2014.01) *H04N 19/157* (2014.01)

(21) International Application Number:
PCT/CN2020/071656

(22) International Filing Date:
13 January 2020 (13.01.2020)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
PCT/CN2019/071510
13 January 2019 (13.01.2019) 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, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

(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,

(54) **Title:** INTERACTION BETWEEN LUT AND SHARED MERGE LIST

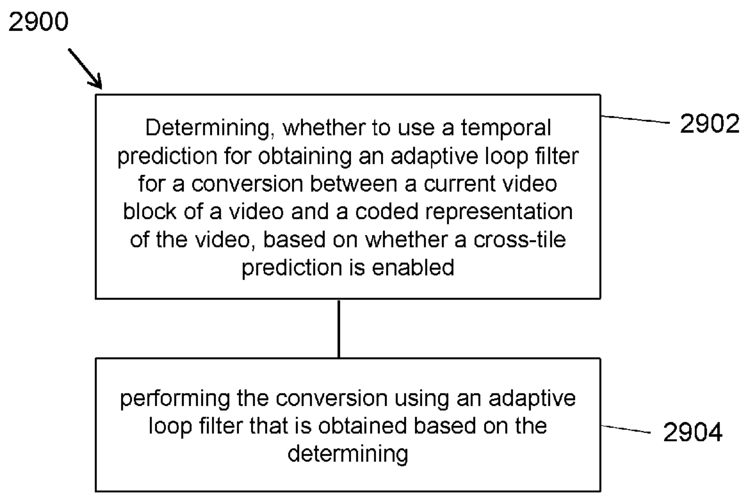


FIG. 29

(57) **Abstract:** A method for video processing is provided to comprise: determining, whether to use a temporal prediction for obtaining an adaptive loop filter for a conversion between a current video block of a video and a coded representation of the video, based on whether a cross-tile prediction is enabled; and performing the conversion using an adaptive loop filter that is obtained based on the determining.

WO 2020/143824 A1

[Continued on next page]

TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:
— *of inventorship (Rule 4.17(iv))*

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

INTERACTION BETWEEN LUT AND SHARED MERGE LIST

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/CN2019/071510, filed on January 13, 2019. For all purposes under the U.S. law, the entire disclosure of the aforementioned application is incorporated by reference as part of the disclosure of this application.

TECHNICAL FIELD

[0002] This patent document relates to video coding 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] This document discloses methods, systems, and devices for encoding and decoding digital video using a merge list of motion vectors.

[0005] In one example aspect, a method for video processing is disclosed. The method includes determining, whether to use a temporal prediction for obtaining an adaptive loop filter for a conversion between a current video block of a video and a coded representation of the video, based on whether a cross-tile prediction is enabled; and performing the conversion using an adaptive loop filter that is obtained based on the determining.

[0006] In another example aspect, a method for video processing is disclosed. The method includes determining whether a sharing of merge list information is enabled for a merge sharing node that corresponds to an ancestor node in a coding unit split tree to allow leaf coding units of the ancestor node in the coding unit split tree to use the merge list information; and performing a conversion between a current video block of a video and a coded representation of the video based on the determining.

[0007] In yet another example aspect, a video encoder device that implements a video encoding method described herein is disclosed.

[0008] In yet another representative aspect, the various techniques described herein may be embodied as a computer program product stored on a non-transitory computer readable media. The computer program product includes program code for carrying out the methods described herein.

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

[0010] The details of one or more implementations are set forth in the accompanying attachments, the drawings, and the description below. Other features will be apparent from the description and drawings, and from the claims.

BRIEF DESCRIPTION OF THE DRAWINGS

[0011] FIG. 1 is a block diagram showing an example of a video encoder implementation

[0012] FIG. 2 illustrates macroblock partitioning in the H.264 video coding standard.

[0013] FIG. 3 illustrates an example of splitting coding blocks (CB) into prediction blocks (PU).

[0014] FIG. 4 illustrates an example implementation for subdivision of a CTB into CBs and transform block (TBs). Solid lines indicate CB boundaries and dotted lines indicate TB boundaries, including an example CTB with its partitioning, and a corresponding quadtree.

[0015] FIG. 5 shows an example of a Quad Tree Binary Tree (QTBT) structure for partitioning video data.

[0016] FIG. 6 shows an example of video block partitioning.

[0017] FIG. 7 shows an example of quad-tree partitioning.

[0018] FIG. 8 shows an example of tree-type signaling.

[0019] FIG. 9 shows an example of a derivation process for merge candidate list construction.

[0020] FIG. 10 shows example positions of spatial merge candidates.

[0021] FIG. 11 shows examples of candidate pairs considered for redundancy check of spatial merge candidates.

[0022] FIG. 12 shows examples of positions for the second PU of Nx2N and 2NxN partitions.

[0023] FIG. 13 illustrates motion vector scaling for temporal merge candidates.

[0024] FIG. 14 shows candidate positions for temporal merge candidates, and their co-located picture.

[0025] FIG. 15 shows an example of a combined bi-predictive merge candidate.

[0026] FIG. 16 shows an example of a derivation process for motion vector prediction candidates.

[0027] FIG. 17 shows an example of motion vector scaling for spatial motion vector candidates.

[0028] FIG. 18 shows an example Alternative Temporal Motion Vector Prediction (ATMVP) for motion prediction of a CU.

[0029] FIG. 19 pictorially depicts an example of identification of a source block and a source picture.

[0030] FIG. 20 shows an example of one CU with four sub-blocks and neighboring blocks.

[0031] FIG. 21 illustrates an example of bilateral matching.

[0032] FIG. 22 illustrates an example of template matching.

[0033] FIG. 23 depicts an example of unilateral Motion Estimation (ME) in Frame Rate Up Conversion (FRUC).

[0034] FIG. 24 shows an example of DMVR based on bilateral template matching.

[0035] FIG. 25 shows an example of spatially neighboring blocks used to derive spatial merge candidates.

[0036] FIG. 26 depicts an example how selection of a representative position for look-up table updates.

[0037] FIG. 27A and 27B illustrate examples of updating look up table with new set of motion information.

[0038] FIGS. 28A and 28B show examples of hardware platforms for implementing a visual media decoding or a visual media encoding technique described in the present document.

[0039] FIG. 29 is an example of a flowchart for video processing method based on some implementations of the disclosed technology.

[0040] FIG. 30 is an example of a flowchart for video processing method based on some implementations of the disclosed technology.

[0041] FIG. 31 shows an example of a decoding flow chart with the proposed HMVP method.

[0042] FIG. 32 shows examples of updating tables using the proposed HMVP method.

[0043] FIGS. 33A and 33B show examples of a redundancy-removal based LUT updating method (with one redundancy motion candidate removed).

[0044] FIGS. 34A and 34B show examples of a redundancy-removal based LUT updating method (with multiple redundancy motion candidates removed).

[0045] FIG. 35 shows an example of differences between Type 1 and Type 2 blocks.

DETAILED DESCRIPTION

[0046] To improve compression ratio of video, researchers are continually looking for new techniques by which to encode video.

[0047] **1. Introduction**

[0048] The present document is related to video coding technologies. Specifically, it is related to motion information coding (such as merge mode, AMVP mode) in video coding. It may be applied to the existing video coding standard like HEVC, or the standard (Versatile Video Coding) to be finalized. It may be also applicable to future video coding standards or video codec.

[0049] **Brief discussion**

[0050] Video coding standards have evolved primarily through the development of the well-known ITU-T and ISO/IEC standards. The ITU-T produced H.261 and H.263, ISO/IEC produced MPEG-1 and MPEG-4 Visual, and the two organizations jointly produced the H.262/MPEG-2 Video and H.264/MPEG-4 Advanced Video Coding (AVC) and H.265/HEVC standards. Since H.262, the video coding standards are based on the hybrid video coding structure wherein temporal prediction plus transform coding are utilized. An example of a typical HEVC encoder framework is depicted in FIG. 1.

2.1 Partition Structure

2.1.1 Partition tree structure in H.264/AVC

[0051] 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.

[0052] An intra-coded block uses spatial prediction to exploit spatial correlation among pixels. Two partitions are defined: 16x16 and 4x4.

[0053] An inter-coded block uses temporal prediction, instead of spatial prediction, by estimating motion among pictures. Motion can be estimated independently for either 16x16 macroblock or any of its sub-macroblock partitions: 16x8, 8x16, 8x8, 8x4, 4x8, 4x4 (see FIG. 2). Only one motion vector (MV) per sub-macroblock partition is allowed.

[0054] **2.1.2 Partition tree structure in HEVC**

[0055] In HEVC, a CTU is split into 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 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] In the following, the various features involved in hybrid video coding using HEVC are highlighted as follows.

[0057] 1) Coding tree units 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×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 a MxM CU.

[0060] 4) 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.

[0061] FIG. 4 shows an example of a subdivision of a CTB into CBs and transform block (TBs)]. Solid lines indicate CB borders and dotted lines indicate TB borders. (a) CTB with its partitioning. (b) corresponding quadtree.

[0062] **2.1.2.1 Tree-Structured Partitioning into Transform Blocks and Units**

[0063] 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 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.

[0064] 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.

[0065] 2.1.2.2 Parent and child nodes

[0066] 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.

[0067] 2.1.3 Quadtree plus binary tree block structure with larger CTUs in JEM

[0068] To explore the future video coding technologies beyond HEVC, Joint Video Exploration Team (JVET) was founded by VCEG and MPEG jointly in 2015. Since then, many new methods have been adopted by JVET and put into the reference software named Joint Exploration Model (JEM).

[0069] 2.1.3.1 QTBT block partitioning structure

[0070] Different from 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. 5, 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.

[0071] The following parameters are defined for the QTBT partitioning scheme.

– CTU size: the root node size of a quadtree, the same concept as in HEVC
– *MinQTSize*: the minimally allowed quadtree leaf node size
– *MaxBTSize*: the maximally allowed binary tree root node size
– *MaxBTDepth*: the maximally allowed binary tree depth
– *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 leaf quadtree 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 leaf quadtree 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. 5 (left) illustrates an example of block partitioning by using QTBT, and FIG. 5 (right) illustrates 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.

[0076] **2.1.4 Ternary-tree for VVC**

[0077] Tree types other than quad-tree and binary-tree are supported. In the implementation, two more ternary tree (TT) partitions, i.e., horizontal and vertical center-side ternary-trees are introduced, as shown in FIG. 6 (d) and (e).

[0078] FIG. 6 shows: (a) quad-tree partitioning (b) vertical binary-tree partitioning (c) horizontal binary-tree partitioning (d) vertical center-side ternary-tree partitioning (e) horizontal center-side ternary-tree partitioning.

[0079] There are two levels of trees, 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'.

[0080] **2.1.5 Partitioning structure**

[0081] The tree structure used in this response, called Multi-Tree Type (MTT), is a generalization of the QTBT. In QTBT, as shown in FIG. 5, 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.

[0082] The fundamental structure of MTT constitutes of two types of tree nodes: Region Tree (RT) and Prediction Tree (PT), supporting nine types of partitions, as shown in FIG. 7.

[0083] FIG. 7 shows: (a) quad-tree partitioning (b) vertical binary-tree partitioning (c) horizontal binary-tree partitioning (d) vertical ternary-tree partitioning (e) horizontal ternary-tree partitioning (f) horizontal-up asymmetric binary-tree partitioning (g) horizontal-down asymmetric binary-tree partitioning (h) vertical-left asymmetric binary-tree partitioning (i) vertical-right asymmetric binary-tree partitioning.

[0084] A region tree can recursively split a CTU into square blocks down to a 4x4 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 (BT), Ternary Tree (TT), and Asymmetric Binary Tree (ABT). 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. 8.

[0085] **2.2 Inter prediction in HEVC/H.265**

[0086] Each inter-predicted PU has motion parameters for one or two reference picture lists. Motion parameters include a motion vector and a reference picture index. Usage of one of the two reference picture lists may also be signalled using *inter_pred_idc*. Motion vectors may be explicitly coded as deltas relative to predictors, such a coding mode is called AMVP mode.

[0087] 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 neighbouring 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 signalled explicitly per each PU.

[0088] When signalling 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.

[0089] When signalling 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.

[0090] The following text provides the details on the inter prediction modes specified in HEVC. The description will start with the merge mode.

[0091] 2.2.1 Merge mode

[0092] 2.2.1.1 Derivation of candidates for merge mode

[0093] 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 is specified in the HEVC standard and can be summarized according to the following sequence of steps:

- Step 1: Initial candidates derivation
  - Step 1.1: Spatial candidates derivation
  - Step 1.2: Redundancy check for spatial candidates
  - Step 1.3: Temporal candidates derivation
- Step 2: Additional candidates insertion
  - Step 2.1: Creation of bi-predictive candidates
  - Step 2.2: Insertion of zero motion candidates

[0094] These steps are also schematically depicted in FIG. 9. 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×2N prediction unit.

[0095] In the following, the operations associated with the aforementioned steps are detailed.

[0096] 2.2.1.2 Spatial **candidates derivation**

[0097] 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. 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 2Nx2N. As an example, FIG. 12 depicts the second PU for the case of N×2N and 2N×N, respectively. When the current PU is partitioned as N×2N, candidate at position A_1 is not considered for list construction. In fact, by adding this candidate will lead to two prediction units having the same motion information, which is redundant to just have one PU in a coding unit. Similarly, position B_1 is not considered when the current PU is partitioned as 2N×N.

[0098] **2.2.1.3 Temporal candidate derivation**

[0099] 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 signalled in the slice header. The scaled motion vector for temporal merge candidate is obtained as illustrated by the dashed line in FIG. 13, which is scaled from the motion vector of the co-located PU using the POC distances, tb and td, where tb is defined to be the POC difference between the reference picture of the current picture and the current picture and td 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. A practical realization of the scaling process is described in the HEVC specification. 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. Illustration of motion vector scaling for temporal merge candidate.

[00100] In the co-located PU (Y) belonging to the reference frame, the position for the temporal candidate is selected between candidates C_0 and C_1, as depicted in FIG. 14. If PU at position C_0 is not available, is intra coded, or is outside of the current CTU, position C_1 is used. Otherwise, position C_0 is used in the derivation of the temporal merge candidate.

[00101] **2.2.1.4 Additional candidates insertion**

[00102] 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. As an example, FIG. 15 depicts the case when two candidates in the original list (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 (on the right). There are numerous rules regarding the combinations which are considered to generate these additional merge candidates.

[00103] 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. Finally, no redundancy check is performed on these candidates.

[00104] **2.2.1.5 Motion estimation regions for parallel processing**

[00105] 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 neighbourhood 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, HEVC defines the motion estimation region (MER) whose size is signalled in the picture parameter set 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.

7.3.2.3 Picture parameter set RBSP syntax
7.3.2.3.1 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)
…	
rbsp_trailing_bits( )	
}	

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 as specified in clause 8.5.3.2.2 and the derivation process for spatial merging candidates as specified in clause 8.5.3.2.3. The value of log2_parallel_merge_level_minus2 shall be in the range of 0 to CtbLog2SizeY − 2, inclusive.

The variable Log2ParMrgLevel is derived as follows:

Log2ParMrgLevel = log2_parallel_merge_level_minus2 + 2 (7-37)

NOTE 3 – 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.

[00106] **2.2.2 Motion vector prediction in AMVP mode**

[00107] Motion vector prediction exploits spatio-temporal correlation of motion vector with neighbouring 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 neighbouring 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 signalling, the index of the best motion vector candidate is encoded using truncated unary. The maximum value to be encoded in this case is 2 (e.g., FIGs. 2 to 8). In the following sections, details about derivation process of motion vector prediction candidate are provided.

[00108] 2.2.2.1 Derivation of motion vector prediction candidates

[00109] FIG. 16 summarizes derivation process for motion vector prediction candidate.

[00110] 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 depicted in FIG. 11.

[00111] 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.

[00112] 2.2.2.2 Spatial motion vector candidates

[00113] 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 depicted in FIG. 11, 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.

- No spatial scaling
  - (1) Same reference picture list, and same reference picture index (same POC)
  - (2) Different reference picture list, but same reference picture (same POC)
- Spatial scaling
  - (3) Same reference picture list, but different reference picture (different POC)
  - (4) Different reference picture list, and different reference picture (different POC)

[00114] The no-spatial-scaling cases are checked first followed by the 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.

[00115] In a spatial scaling process, the motion vector of the neighbouring PU is scaled in a similar manner as for temporal scaling, as depicted as FIG. 17. The main 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.

[00116] **2.2.2.3 Temporal motion vector candidates**

[00117] Apart for 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 (see, e.g., FIG. 6). The reference picture index is signalled to the decoder.

[00118] **2.2.2.4 Signaling of AMVP information**

[00119] For the AMVP mode, four parts may be signalled in the bitstream, i.e., prediction direction, reference index, MVD and mv predictor candidate index.

<u>Syntax tables</u>:

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)
}	
}	
}	
}	

7.3.8.9 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)
}	
}	

[00120] 2.3 New inter prediction methods in JEM (Joint Exploration Model)

[00121] 2.3.1 Sub-CU based motion vector prediction

[00122] In the JEM with QTBT, each CU can have at most one set of motion parameters for each prediction direction. 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.

[00123] To preserve more accurate motion field for sub-CU motion prediction, the motion compression for the reference frames is currently disabled.

[00124] 2.3.1.1 Alternative temporal motion vector prediction

[00125] In the alternative temporal motion vector prediction (ATMVP) method, the motion vectors temporal motion vector prediction (TMVP) is modified by fetching multiple sets of motion information (including motion vectors and reference indices) from blocks smaller than the current CU. As shown in FIG. 18, the sub-CUs are square N×N blocks (N is set to 4 by default).

[00126] ATMVP predicts the motion vectors of the sub-CUs within a CU in two steps. The first step is to identify the corresponding block in a reference picture with a so-called temporal vector. The reference picture is called the motion source picture. The second step is to split the current CU into sub-CUs and obtain the motion vectors as well as the reference indices of each sub-CU from the block corresponding to each sub-CU, as shown in FIG. 18.

[00127] In the first step, a reference picture and the corresponding block is determined by the motion information of the spatial neighbouring blocks of the current CU. To avoid the repetitive scanning process of neighbouring blocks, the first merge candidate in the merge candidate list of the current CU 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, in ATMVP, 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. 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.

[00128] FIG. 19 shows an example of the identification of source block and source picture

[00129] In the second step, a corresponding block of the sub-CU is identified by the temporal vector in the motion source picture, by adding to the coordinate of the current CU the temporal vector. For each sub-CU, the motion information of its corresponding block (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×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 (i.e. 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 (the motion vector corresponding to reference picture list X) to predict motion vector MV_y (with X being equal to 0 or 1 and Y being equal to 1−X) for each sub-CU.

[00130] **2.3.1.2 Spatial-temporal motion vector prediction**

[00131] In this method, the motion vectors of the sub-CUs are derived recursively, following raster scan order. FIG. 20 illustrates this concept. Let us consider an 8×8 CU which contains four 4×4 sub-CUs A, B, C, and D. The neighbouring 4×4 blocks in the current frame are labelled as a, b, c, and d.

[00132] The motion derivation for sub-CU A starts by identifying its two spatial neighbours. The first neighbour is the N×N block above sub-CU A (block c). If this block c is not available or is intra coded the other N×N blocks above sub-CU A are checked (from left to right, starting at block c). The second neighbour is a block to the left of the sub-CU A (block b). If block b is not available or is intra coded other blocks to the left of sub-CU A are checked (from top to bottom, staring at block b). The motion information obtained from the neighbouring 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 is derived by following the same procedure of TMVP derivation as specified in HEVC. The motion information of the collocated block at location D is fetched and scaled accordingly. Finally, after retrieving and scaling the motion information, all available motion vectors (up to 3) are averaged separately for each reference list. The averaged motion vector is assigned as the motion vector of the current sub-CU.

[00133] FIG. 20 shows an example of one CU with four sub-blocks (A-D) and its neighbouring blocks (a–d).

[00134] **2.3.1.3 Sub-CU motion prediction mode signalling**

[00135] 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. Up to seven merge candidates are 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 is needed for the two additional merge candidates.

[00136] In the JEM, all bins of merge index is context coded by CABAC. While in HEVC, only the first bin is context coded and the remaining bins are context by-pass coded.

[00137] **2.3.2 Adaptive motion vector difference resolution**

[00138] In HEVC, 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.

[00139] 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.

[00140] 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.

[00141] 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.

[00142] 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.

[00143] 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.

[00144] **2.3.3 Pattern matched motion vector derivation**

[00145] Pattern matched motion vector derivation (PMMVD) mode is a special merge mode based on Frame-Rate Up Conversion (FRUC) techniques. With this mode, motion information of a block is not signalled but derived at decoder side.

[00146] A FRUC flag is signalled for a CU when its merge flag is true. When the FRUC flag is false, a merge index is signalled and the regular merge mode is used. When the FRUC flag is true, an additional FRUC mode flag is signalled to indicate which method (bilateral matching or template matching) is to be used to derive motion information for the block.

[00147] At 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. That is the two matching modes (bilateral matching and template matching) are both 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.

[00148] 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 which 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 and 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.

[00149] 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 (16), 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.

$$M = \max\{4, \min\{\tfrac{M}{2^D}, \tfrac{N}{2^D}\}\} \quad (1)$$

[00150] As shown in the FIG. 21, 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 in two different reference pictures. Under the assumption of continuous motion trajectory, the motion vectors MV0 and MV1 pointing to the two reference blocks shall be proportional to the temporal distances, i.e., TD0 and TD1, between the current picture and the two reference pictures. As a special case, when the current picture is temporally between the two reference pictures 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.

[00151] As shown in FIG. 22, template matching is used to derive motion information of the current CU by finding the closest match between a template (top and/or left neighbouring blocks of the current CU) in the current picture and a block (same size to the template) in a reference picture. Except the aforementioned FRUC merge mode, the template matching is also applied to AMVP mode. In the JEM, as done in HEVC, AMVP has two candidates. With template matching method, a new candidate is 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 (meaning remove the second existing AMVP candidate). When applied to AMVP mode, only CU level search is applied.

[00152] 2.3.3.1 CU level MV candidate set

[00153] The MV candidate set at CU level consists of:

(i) Original AMVP candidates if the current CU is in AMVP mode

(ii) all merge candidates,

(iii) several MVs in the interpolated MV field.

(iv) top and left neighbouring motion vectors

[00154] When using bilateral matching, each valid MV of a merge candidate is 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 (MVa, refa) at reference list A. Then the reference picture refb of its paired bilateral MV is found in the other reference list B so that refa and refb are temporally at different sides of the current picture. If such a refb is not available in reference list B, refb is determined as a reference which is different from refa and its temporal distance to the current picture is the minimal one in list B. After refb is determined, MVb is derived by scaling MVa based on the temporal distance between the current picture and refa, refb.

[00155] Four MVs from the interpolated MV field are also 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.

[00156] When FRUC is applied in AMVP mode, the original AMVP candidates are also added to CU level MV candidate set.

[00157] At the CU level, up to 15 MVs for AMVP CUs and up to 13 MVs for merge CUs are added to the candidate list.

[00158] **2.3.3.2 Sub-CU level MV candidate set**

[00159] The MV candidate set at sub-CU level consists of:

(i) an MV determined from a CU-level search,

(ii) top, left, top-left and top-right neighbouring MVs,

(iii) scaled versions of collocated MVs from reference pictures,

(iv) up to 4 ATMVP candidates,

(v) up to 4 STMVP candidates

[00160] The scaled MVs from reference pictures are derived as follows. All 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.

[00161] ATMVP and STMVP candidates are limited to the four first ones.

[00162] At the sub-CU level, up to 17 MVs are added to the candidate list.

[00163] **2.3.3.3 Generation of interpolated MV field**

[00164] 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.

[00165] First, the motion field of each reference pictures in both reference lists is traversed at 4×4 block level. For each 4×4 block, if the motion associated to the block passing through a 4×4 block in the current picture (as shown in FIG. 23) 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.

[00166] **2.3.3.4 Interpolation and matching cost**

[00167] 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 is used for both bilateral matching and template matching.

[00168] 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 is 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:

$$C = SAD + w \cdot (|MV_x - MV_x^s| + |MV_y - MV_y^s|) \quad (2)$$

[00169] where w is a weighting factor which is empirically set to 4, MV and MV^s indicate the current MV and the starting MV, respectively. SAD is still used as the matching cost of template matching at sub-CU level search.

[00170] 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.

[00171] 2.3.3.5 MV refinement

[00172] 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.

[00173] 2.3.3.6 Selection of prediction direction in template matching FRUC merge mode

[00174] In the bilateral matching merge mode, bi-prediction is always applied since 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. There is no such limitation for the template matching merge mode. 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 is based on a template matching cost as follows:

If *costBi* <= *factor* * min (*cost*0, *cost1*)

bi-prediction is used;

Otherwise, if *cost*0 <= *cost1*

uni-prediction from list0 is used;

Otherwise,

uni-prediction from list1 is used;

[00175] where 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. The value of *factor* is equal to 1.25, which means that the selection process is biased toward bi-prediction.
The inter prediction direction selection is only applied to the CU-level template matching process.

[00176] **2.3.4 Decoder-side motion vector refinement**

[00177] In 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.

[00178] 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. 23. 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.

[00179] 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.

[00180] **2.3.5 Merge/Skip mode with Bilateral Matching refinement**

[00181] 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 (FIG. 11), 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:

[00182] - Spatial candidates for blocks 1-4.

[00183] - Extrapolated affine candidates for blocks 1-4.

[00184] - ATMVP.

[00185] - STMVP.

[00186] - Virtual affine candidate.

[00187] - Spatial candidate (block 5) (used only when the number of the available candidates is smaller than 6).

[00188] - Extrapolated affine candidate (block 5).

[00189] - Temporal candidate (derived as in HEVC).

[00190] - Non-adjacent spatial candidates followed by extrapolated affine candidate (blocks 6 to 49, as depicted in FIG. 25).

[00191] - Combined candidates.

[00192] - Zero candidates

[00193] It is 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.

[00194] Blocks which are not connected with the current block may be accessed. If a non-adjacent block is coded with non-intra mode, the associated motion information may be added as an additional merge candidate.

[00195] **2.3.6 Shared merge list JVET-M0170**

[00196] It proposes to share the same merging candidate list for all leaf coding units (CUs) of one ancestor node in the CU split tree for enabling parallel processing of small skip/merge-coded CUs. The ancestor node is named merge sharing node. The shared merging candidate list is generated at the merge sharing node pretending the merge sharing node is a leaf CU.

[00197] For Type-2 definition, the merge sharing node will be decided for each CU inside a CTU during parsing stage of decoding; moreover, the merge sharing node is an ancestor node of leaf CU which must satisfy the following 2 criteria:

[00198] The merge sharing node size is equal to or larger than the size threshold

[00199] In the merge sharing node, one of the child CU size is smaller than the size threshold

[00200] Moreover, no samples of the merge sharing node are outside the picture boundary has to be guaranteed. During parsing stage, if an ancestor node satisfies the criteria (1) and (2) but has some samples outside the picture boundary, this ancestor node will not be the merge sharing node and it proceeds to find the merge sharing node for its child CUs.

[00201] Figure 35 shows an example for the difference of Type-1 and Type-2 definition. In this example, the parent node is ternary-split into 3 child CUs. The size of parent node is 128. For Type-1 definition, the 3 child-CUs will be merge sharing nodes separately. But for Type-2 definition, the parent node is the merge sharing node.

[00202] The proposed shared merging candidate list algorithm supports translational merge (including merge mode and triangle merge mode, history-based candidate is also supported) and subblock-based merge mode. For all kinds of merge mode, the behavior of shared merging candidate list algorithm looks basically the same, and it just generates candidates at the merge sharing node pretending the merge sharing node is a leaf CU. It has 2 major benefits. The first benefit is to enable parallel processing for merge mode, and the second benefit is to share all computations of all leaf CUs into the merge sharing node. Therefore, it significantly reduces the hardware cost of all merge modes for hardware codec. By the proposed shared merging candidate list algorithm, the encoder and decoder can easily support parallel encoding for merge mode and it relieves the cycle budget problem of merge mode.

[00203] **2.3.7 Tile groups**

[00204] JVET-L0686 was adopted in which slices are removed in favor of tile groups and the HEVC syntax element slice_address is substituted with tile_group_address in the tile_group_header (if there is more than one tile in the picture) as address of the first tile in the tile group.

[00205] **3. Examples of Problems Addressed by Embodiments disclosed herein**

[00206] 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.

[00207] 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.

[00208] **4. Some Examples**

[00209] 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.

[00210] Some embodiments may use one or more look up tables with at least one motion candidate stored to predict motion information of a block. Embodiments may use motion candidate to indicate a set of motion information stored in a look up table. For conventional AMVP or merge modes, embodiments may use AMVP or merge candidates for storing the motion information.

[00211] The examples below explain general concepts.

[00212] **Examples of look-up tables**

[00213] Example A1: Each look up table may contain one or more motion candidates wherein each candidate is associated with its motion information.

a. Motion information of a motion candidate here may include partial or all of the prediction direction, reference indices/pictures, motion vectors, LIC flag, affine flag, MVD precision, MVD values.
b. Motion information may further include the block position information and/or block shape to indicate wherein the motion information is coming from.
c. A counter may be further assigned for each look up table.
    i. The counter may be initialized to be zero at the beginning of encoding/decoding a picture/slice/LCU(CTU) row/tile.
    ii. In one example, the counter may be updated after encoding/decoding a CTU/CTB/CU/CB/PU/a certain region size (e.g., 8x8 or 16x16).
    iii. In one example, the counter is increased by one each time one candidate is added into the lookup table.

iv. In one example, the counter should be no larger than the table size (number of allowed motion candidates).

v. Alternatively, the counter may be used to indicate how many motion candidates have been tried to be added to the look up tables (some of them was in the look up table but later may be removed from the table). In this case, the counter could be larger than the table size.

d. 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 signalled 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.

e. 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.

[00214] Selection of LUTs

[00215] Example B1: For coding a block, partial or all of motion candidates from one look up table may be checked in order. When one motion candidate is checked during coding a block, it may be added to the motion candidate list (e.g., AMVP, merge candidate lists).

a. Alternatively, motion candidates from multiple look up tables may be checked in order.
b. The look up table indices may be signaled in CTU, CTB, CU or PU, or a region covering multiple CTU/CTB/CU/PUs.

[00216] Example B2: The selection of look up tables may depend on the position of a block.

a. It may depend on the CTU address covering the block. Here, we take two look up tables (Dual Look Up Tables, DLUT) for an example to illustrate the idea:
    i. If the block is located at one of the first *M* CTUs in a CTU row, the first look up table may be utilized for coding the block, while for blocks located in the remaining CTUs in the CTU row, the second look up table may be utilized.
    ii. If the block is located at one of the first M CTUs in a CTU row, motion candidates of the first look up table may firstly checked for coding the block, if there are not enough candidates in the first table, the second look up table may be further utilized. while for blocks located in the remaining CTUs in the CTU row, the second look up table may be utilized.
    iii. Alternatively, for blocks located in the remaining CTUs in the CTU row, motion candidates of the second look up table may firstly checked for coding the block, if there are not enough candidates in the second table, the first look up table may be further utilized.
b. It may depend on the distance between the position of the block and the position associated with one motion candidate in one or multiple look up tables.
    iv. In one example, if one motion candidate is associated with a smaller distance to the block to be coded, it may be checked earlier compared to another motion candidate.

[00217] Usage of look up tables

[00218] Example C1: The total number of motion candidates in a look up table to be checked may be pre-defined.

a. It may further depend on the coded information, block size, block shape and etc. al. For example, for the AMVP mode, only *m* motion candidates may be checked while for the merge mode, *n* motion candidates may be checked (e.g., m = 2, n = 44).
b. In one example, the total number of motion candidates to be checked may be signalled 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.

[00219] Example C2: The motion candidate(s) included in a look up table may be directly inherited by a block.

a. They may be used for the merge mode coding, i.e., motion candidates may be checked in the merge candidate list derivation process.
b. They may be used for the affine merge mode coding.
    i. A motion candidate in a look up table can be added as an affine merge candidate if its affine flag is one.
c. Checking of motion candidates in look up tables may be enabled when:
    i. the merge candidate list is not full after inserting the TMVP candidate;
    ii. the merge candidate list is not full after checking a certain spatial neighboring block for spatial merge candidate derivation;
    iii. the merge candidate list is not full after all spatial merge candidates;
    iv. the merge candidate list is not full after combined bi-predictive merge candidates;
    v. when the number of spatial or temporal (e.g., including adjacent spatial and non-adjacent spatial, TMVP, STMVP, ATMVP, etc. al) merge candidates that have been put into the merge candidate list from other coding methods (e.g., the merge derivation process of HEVC design, or JEM design) is less than the maximumly allowed merge candidates minus a given threshold.
        1. in one example, the threshold is set to 1 or 0.

2. Alternatively, the threshold may be signaled or pre-defined in SPS/PPS/sequence, picture, slice header/tile.
3. Alternatively, the threshold may be adaptively changed from block to block. For example, it may be dependent on coded block information, like block size/block shape/slice type, and/or dependent on the number of available spatial or temporal merge candidates.
4. In another example, when the number of a certain kind of merge candidates than have been put into the merge candidate list is less than the maximumly allowed merge candidates minus a given threshold. The “certain kind of merge candidates” may be spatial candidates as in HEVC or non-adjacent merge candidates.

vi. Pruning may be applied before adding a motion candidate to the merge candidate list.

1. In one example, a motion candidate may be pruned to all or partial of the available spatial or temporal (e.g., including adjacent spatial and non-adjacent spatial, TMVP, STMVP, ATMVP, etc. al) merge candidates from other coding methods in the merge candidate list.
2. a motion candidate may be NOT pruned to sub-block based motion candidates, e.g., ATMVP, STMVP.
3. In one example, a current motion candidate may be pruned to all or partial of the available motion candidates (inserted before the current motion candidate) in the merge candidate list.
4. Number of pruning operations related to motion candidates (that is, how many times that motion candidates need to be compared to other candidates in the merge list) may depend on the number of available spatial or temporal merge candidates. For example, when checking a new motion candidate, if there are M candidates available in the merge list, the new motion candidate may be only compared to the first K (K<=M) candidates. If the pruning

function returns false (e.g., not identical to any of the first K candidates), the new motion candidate is considered to be different from all of the M candidates and it could be added to the merge candidate list. In one example, K is set to min (K, 2).

5. In one example, a newly appended motion candidate is only compared with the first N candidate in the merge candidate list. For example, N = 3, 4 or 5. N may be signaled from the encoder to the decoder.
6. In one example, a new motion candidate to be checked is only compared with the last N candidate in the merge candidate list. For example, N = 3, 4 or 5. N may be signaled from the encoder to the decoder.
7. In one example, how to select candidates previously added in the list to be compared with a new motion candidate from a table may depend on where the previously added candidates derived from.
   a. In one example, a motion candidate in a look-up table may be compared to candidates derived from a given temporal and/or spatial neighboring block.
   b. In one example, different entries of motion candidates in a look-up table may be compared to different previously added candidates (i.e., derived from different locations).

[00220] Example C3: The motion candidate(s) included in a look up table may be used as a predictor for coding motion information of a block.

a. They may be used for the AMVP mode coding, i.e., motion candidates may be checked in the AMVP candidate list derivation process.
b. Checking of motion candidates in look up tables may be enabled when:
   i. the AMVP candidate list is not full after inserting the TMVP candidate;
   ii. the AMVP candidate list is not full after selecting from spatial neighbors and pruning, right before inserting the TMVP candidate;

iii. when there is no AMVP candidate from above neighboring blocks without scaling and/or when there is no AMVP candidate from left neighboring blocks without scaling
iv. Pruning may be applied before adding a motion candidate to the AMVP candidate list.
v. Similar rules as mentioned in bullet 5.(3) (4) may be applied to AMVP mode

c. Motion candidates with identical reference picture to the current reference picture is checked.
    i. Alternatively, in addition, motion candidates with different reference pictures from the current reference picture are also checked (with MV scaled).
    ii. Alternatively, all motion candidates with identical reference picture to the current reference picture are first checked, then, motion candidates with different reference pictures from the current reference picture are checked.
    iii. Alternatively, motion candidates are checked following the same in merge.

[00221] Example C4: The checking order of motion candidates in a look up table is defined as follows (suppose K (K>=1) motion candidates are allowed to be checked):

a. The last K motion candidates in the look up table, e.g., in descending order of entry indices to the LUT.
b. The first K%L candidates wherein L is the look up table size when K>=L, e.g., in descending order of entry indices to the LUT.
c. All the candidates (L candidates) in the look up table when K >=L, based on an order. In one example, the first K%L candidates in the table are checked in descending order of entry indices to the LUT, and then check the last (L – K%L) candidates in descending order of entry indices.
d. Alternatively, furthermore, based on the descending order of motion candidate indices.

e. Alternatively, furthermore, based on the ascending order of motion candidate indices
f. Alternatively, selecting K motion candidates based on the candidate information, such as the distance of positions associated with the motion candidates and current block.
   i. In one example, K nearest motion candidates are selected.
   ii. in one example, the candidate information may further consider block shape when calculating the distance
g. In one example, the checking order of K of motion candidates from the table which includes L candidates may be defined as: selecting those candidates with index equal to a_0, a_0+T_0, $a_0+T_0+T_1$, $a_0+ T_0+ T_1+T_2$, … $a_0+T_0+T_1+T_2+..+T_{K-1}$ in order wherein a_0 and T_i (*i* being 0 … K-1) are integer values.
   i. In one example, a_0 is set to 0 (i.e., the first entry of motion candidate in the table). Alternatively, a_0 is set to (K – L/K). The arithmetic operation '/' is defined as integer division with truncation of the result toward zero. Alternatively, a_0 is set to any integer between 0 and L/K.
      1. Alternatively, the value of a_0 may depend on coding information of the current block and neighbouring blocks.
   ii. In one example, all the intervals T_i (*i* being 0 … K-1) are the same, such as L/K. The arithmetic operation '/' is defined as integer division with truncation of the result toward zero.
   iii. In one example, (K, L, a_0, T_i) is set to (4, 16, 0, 4), or (4, 12, 0, 3) or (4, 8, 0, 1) or (4, 16, 3, 4) or (4, 12, 2, 3), or (4, 8, 1, 2). T_i are the same for all i.
   iv. Such method may be only applied when K is smaller than L.
   v. Alternatively, furthermore, when K is larger than or equal to a threshold, bullet 7.c. may be applied. The threshold may be defined as L, or it may depend on K or adaptively changed from block to block. In one example, the threshold may depend on the number of available motion candidate in the list before adding a new one from the look-up table
h. In one example, the checking order of K of motion candidates from the table which includes L candidates may be defined as: selecting those candidates with index

equal to a_0, a_0-T_0, a_0-T_0-T_1, a_0-T_0- T_1-T_2, … a_0-T_0-T_1-T_2-..-T_{K-1} in order wherein a_0 and T_i (*i* being 0 … K-1) are integer values.

i. In one example, a_0 is set to L-1 (i.e., the last entry of motion candidate in the table). Alternatively, a_0 is set to any integer between L-1-L/K and L-1.

ii. In one example, all the intervals T_i (*i* being 0 … K-1) are the same, such as L/K.

iii. In one example, (K, L, a_0, T_i) is set to (4, 16, L-1, 4), or (4, 12, L-1, 3) or (4, 8, L-1, 1) or (4, 16, L-4, 4) or (4, 12, L-3, 3), or (4, 8, L-2, 2). T_i are the same for all i.

iv. Such method may be only applied when K is smaller than L.

Alternatively, furthermore, when K is larger than or equal to a threshold, bullet 7.c. may be applied. The threshold may be defined as L, or it may depend on K or adaptively changed from block to block. In one example, the threshold may depend on the number of available motion candidate in the list before adding a new one from the look-up table.

i. How many and/or how to select motion candidates from a look table may depend on the coded information, such as block size/block shape.

i. In one example, for a smaller block size, instead of choosing the last K motion candidates, the other K motion candidates (starting not from the last one) may be chosen.

ii. In one example, the coded information may be the AMVP or merge mode.

iii. In one example, the coded information may be the affine mode or non-affine AMVP mode or non-affine merge mode.

iv. In one example, the coded information may be the affine AMVP (inter) mode affine merge mode or non-affine AMVP mode or non-affine merge mode.

v. In one example, the coded information may be Current Picture Reference (CPR) mode or not CPR mode.

vi. Alternatively, how to select motion candidates from a look-up table may further depend on the number of motion candidates in the look-up table, and/or number of available motion candidates in the list before adding a new one from the look-up table.

j. In one example, maximum number of motion candidates in a look up table to be checked (i.e., which may be added to the merge/amvp candidate list) may depend on the number of available motion candidates (denoted by $N_{avaiMCinLUT}$) in a look up table, and/or maximally allowed motion candidates (denoted by NUM_{maxMC}) to be added (which may be pre-defined or signaled), and/or number of available candidates (denoted by N_{avaiC}) in a candidate list before checking the candidates from the look up table.
   i. In one example, maximum number of motion candidates in the look up table to be checked is set to minimum value of ($N_{avaiMCinLUT}$, NUM_{maxMC} , N_{avaiC}).
   ii. Alternatively, maximum number of motion candidates in the look up table to be checked is set to minimum value of ($N_{avaiMCinLUT}$, NUM_{maxMC} - N_{avaiC}).
   iii. In one example, N_{avaiC} denotes the number of inserted candidates derived from spatial or temporal (adjacent and/or non-adjacent) neighboring blocks. Alternatively, furthermore, the number of sub-block candidates (like AMTVP, STMVP) is not counted in N_{avaiC}.
   iv. NUM_{maxMC} may depend on the coded mode, e.g., for merge mode and AMVP mode, NUM_{maxMC} may be set to different values. In one example, for merge mode, NUM_{maxMC} may be set to 4, 6, 8, 10, etc. al. for AMVP mode, NUM_{maxMC} may be set to 1, 2, 4, etc. al.
   v. Alternatively, NUM_{maxMC} may depend on other coded information, like block size, block shape, slice type etc. al.

k. The checking order of different look up tables is defined in usage of look up tables in the next subsection.

l. The checking process will terminate once the merge/AMVP candidate list reaches the maximumly allowed candidate numbers.

m. The checking process will terminate once the merge/AMVP candidate list reaches the maximumly allowed candidate numbers minus a threshold (Th). In one example, Th may be pre-defined as a positive integer value, e.g., 1, or 2, or 3. Alternatively, Th may be adaptively changed from block to block. Alternatively, Th may be signaled in the SPS/PPS/slice header etc. al. Alternatively, Th may further depend on block shape/block size/coded modes etc. Alternatively, Th may depend on how many available candidates before adding the motion candidates from LUTs.

n. Alternatively, it will terminate once the number of added motion candidates reaches the maximumly allowed motion candidate numbers. The maximumly allowed motion candidate numbers may be signaled or pre-defined. Alternatively, the maximumly allowed motion candidate numbers may further depend on block shape/block size/coded modes etc.

o. One syntax element to indicate the table size as well as the number of motion candidates (i.e., K=L) allowed to be checked may be signaled in SPS, PPS, Slice header, tile header.

[00222] Example C5: Enabling/disabling the usage look up tables for motion information coding of a block may be signalled in SPS, PPS, Slice header, tile header, CTU, CTB, CU or PU, region covering multiple CTU/CTB/CU/PUs.

[00223] Example C6: Whether to apply prediction from look up tables may further depend on the coded information. When it is inferred not to apply for a block, additional signaling of indications of the prediction is skipped. Alternatively, when it is inferred not to apply for a block, there is no need to access motion candidates of look up tables, and the checking of related motion candidates is omitted.

a. Whether to apply prediction from look up tables may depend on block size/block shape. In one example, for smaller blocks, such as 4x4, 8x4 or 4x8 blocks, it is disallowed to perform prediction from look up tables.
b. Whether to apply prediction from look up tables may depend on whether the block is coded with AMVP or merge mode. In one example, for the AMVP mode, it is disallowed to perform prediction from look up tables.
c. Whether to apply prediction from look up tables may depend on the block is coded with affine motion or other kinds of motion (such as translational motion). In one example, for the affine mode, it is disallowed to perform prediction from look up tables.

[00224] Example C7: Motion candidates of a look up table in previously coded frames/slices/tiles may be used to predict motion information of a block in a different frame/slice/tile.

a. In one example, only look up tables associated with reference pictures of current block may be utilized for coding current block.

b. In one example, only look up tables associated with pictures with the same slice type and/or same quantization parameters of current block may be utilized for coding current block.

[00225] Update of look up tables

[00226] Example D1: After coding a block with motion information (i.e., IntraBC mode, inter coded mode), one or multiple look up tables may be updated.

a. In one example, whether to update a look up table may reuse the rules for selecting look up tables, e.g., when a look up table could be selected for coding the current block, after coding/decoding the block, the selected look up table may further be updated.
b. Look up tables to be updated may be selected based on coded information, and/or positions of the block/LCU.
c. If the block is coded with motion information directly signaled (such as AMVP mode, MMVD mode for normal/affine inter mode, AMVR mode for normal/affine inter mode), the motion information for the block may be added to a look up table.
    i. Alternatively, if the block is coded with motion information directly inherited from a spatial neighboring block without any refinement (e.g., spatial merge candidate without refinement), the motion information for the block shouldn't be added to a look up table.
    ii. Alternatively, if the block is coded with motion information directly inherited from a spatial neighboring block with refinement (such as DMVR, FRUC), the motion information for the block shouldn't be added to any look up table.
    iii. Alternatively, if the block is coded with motion information directly inherited from a motion candidate stored in a look up table, the motion information for the block shouldn't be added to any look up table.
    iv. In one example, such motion information may be directly added to the look up table, such as to the last entry of the table or to the entry which is used for storing the next available motion candidate.

v. Alternatively, such motion information may be directly added to the look up table without pruning, e.g., without any pruning.

vi. Alternatively, such motion information may be used to reorder the look up table.

vii. Alternatively, such motion information may be used to update the look up table with limited pruning (e.g., compared to the latest one in the look up table).

d. M (M>=1) representative position within the block is chosen and the motion information associated with the representative is used to update look up tables.

i. In one example, the representative position is defined as one of the four corner positions (e.g., C0-C3 in FIG. 26) within the block.

ii. In one example, the representative position is defined as the center position (e.g., Ca – Cd in FIG. 26) within the block.

iii. When sub-block prediction is disallowed for block, M is set to 1.

iv. When sub-block prediction is allowed for block, M could be set to 1 or total number of sub-blocks or any other value between [1, number of sub-blocks] exclusively.

v. Alternatively, when sub-block prediction is allowed for block, M could be set to 1 and the selection of a representative sub-block is based on

1. the frequency of utilized motion information,
2. whether it is a bi-prediction block
3. based on the reference picture index/reference picture
4. motion vector differences compared to other motion vectors (e.g., selecting the maximum MV differences)
5. other coded information.

e. When M (M>=1) sets of representative positions are selected to update look up tables, further conditions may be checked before adding them as additional motion candidates to look up tables.

i. Pruning may be applied to the new sets of motion information to the existing motion candidates in the look up table.

ii. In one example, a new set of motion information shouldn't be identical to any or partial of existing motion candidates in the look up table.

iii. Alternatively, for same reference pictures from a new set of motion information and one existing motion candidate, the MV difference should be no smaller than one/multiple thresholds. For example, horizontal and/or vertical component of the MV difference should be larger than 1-pixel distance.

iv. Alternatively, the new sets of motion information are only pruned with the last K candidates or the first K%L existing motion candidates when K > L to allow reactivating the old motion candidates.

v. Alternatively, no pruning is applied.

f. If M sets of motion information are used to update a look up table, the corresponding counter should be increased by M.

g. Suppose a counter of a look up table to be updated is denoted by K before coding the current block, after coding the block, for one selected set of motion information (with methods mentioned above0, it is added as an additional motion candidate with index equal to K%L (wherein L is the look up table size). Examples are shown in FIG. 27A and 27B

i. Alternatively, it is added as an additional motion candidate with index equal to min(K+1, L-1). Alternatively, furthermore, if K >= L, the first motion candidate (index equal to 0) is removed from the look-up table, and the following K candidates indices are reduced by 1.

ii. For above both methods (either adding the new motion candidate to entry index equal to K%L or adding it with index equal to min(K+1, L-1)), they are trying to keep the latest few sets of motion information from previously coded blocks regardless whether there are identical/similar motion candidates.

iii. Alternatively, when adding a new set of motion information as a motion candidate to a LUT, redundancy checking is firstly applied. In this case, the LUT will keep the latest several sets of motion information from previously coded blocks, however, redundant ones may be removed from

LUTs. Such a method is called redundancy-removal based LUT updating method.

1. If there are redundant motion candidates in the LUT, the counter associated with the LUT may be not increased or decreased.
2. The redundant checking may be defined as the pruning process in merge candidate list construction process, e.g., checking whether the reference pictures/reference picture indices are the same, and motion vector differences are within a range or identical.
3. If there is a redundant motion candidate found in a LUT, the redundant motion candidate is moved from its current position to the last one of the LUT.
    a. Similarly, if there is a redundant motion candidate found in a LUT, this redundant motion candidate is removed from the LUT. In addition, all the motion candidates inserted to LUT after the redundant motion candidate move forward to refill the removed entry of the redundant motion candidate. After the shifting, the new motion candidate is added to the LUT.
    b. In this case, the counter is kept unchanged.
    c. Once a redundant motion candidate is identified in a LUT, the redundant checking process is terminated.
4. Multiple redundant motion candidates may be identified. In this case, all of them are removed from the LUT. In addition, all of the remaining motion candidates may move forward in order.
    a. In this case, the counter is decreased by (number of redundant motion candidates minus 1).
    b. The redundant checking process is terminated after identifying *maxR* redundant motion candidates (*maxR* is a positive integer variable).

5. The redundancy checking process may start from the first to the last motion candidate (i.e., in the order of added to LUTs, in the order of decoding process of blocks where motion information is from).
6. Alternatively, when there are redundant motion candidates in LUT, instead of removing one or multiple of redundant ones form LUTs, virtual motion candidates may be derived from redundant ones and the virtual motion candidates may be used to replace the redundant ones.
   a. Virtual motion candidates may be derived from a redundant motion candidate by adding offset(s) to horizontal and/or vertical component of one or multiple motion vectors; or average of two motion vectors if pointing to the same reference pictures. Alternatively, the virtual motion candidate may be derived from any function with motion vectors in the look up table as the input. Exemplary functions are: Adding two or motion vectors together; Averaging two or more motion vectors. The motion vectors may be scaled before being input into the function.
   b. Virtual motion candidates may be added to the same position as the redundant motion candidates.
   c. Virtual motion candidates may be added before all the other motion candidates (e.g., starting from smallest entry indices, like zero).
   d. In one example, it is applied only under certain conditions, such as when the current LUT is not full.
7. The redundancy-removal based LUT updating method may be invoked under certain conditions, such as
   a. the current block is coded with merge mode,

b. the current block is coded with AMVP mode but with at least one component of MV difference is non-zero;
c. the current block is or is not coded with sub-block based motion prediction/motion compensation methods (e.g., not coded with affine mode)
d. the current block is coded with merge mode and the motion information is associated with a certain type (e.g., from the spatial neighboring blocks, from the left neighboring block, from the temporal block)

h. After encoding/decoding one block, one or more look-up tables may be updated by just inserting the M sets of motion information to the end of the table, i.e., after all existing candidates.
    i. Alternatively, furthermore, some existing motion candidates in the table may be removed.
        1. In one example, if the table is full after inserting the M sets of motion information, the first several entries of motion candidates may be removed from the table.
        2. In one example, if the table is full before inserting the M sets of motion information, the first several entries of motion candidates may be removed from the table.
    ii. Alternatively, furthermore, if the block is coded with a motion candidate from a table, the motion candidates in the table may be reordered so that the selected motion candidate is put to the last entry of the table.

i. In one example, before encoding/decoding a block, a look-up table may include motion candidates denoted by $HMVP_0$, $HMVP_1$, $HMVP_2$, …, $HMVP_{K-1}$, $HMVP_K$, $HMVP_{K+1}$, …, $HMVP_{L-1}$ wherein $HMVP_i$ denotes the i-th entry in the look-up table. If the block is predicted from $HMVP_K$ (K is within the range [0, L-1], inclusively), after encoding/decoding this block, the look-up table is re-ordered to: $HMVP_0$, $HMVP_1$, $HMVP_2$, …, $HMVP_{K-1}$, ~~$HMVP_K$~~, $HMVP_{K+1}$, …, $HMVP_{L-1}$, $HMVP_K$.

j. The look-up table may be emptied after coding one intra-constrained block.

k. If an entry of motion information is added into the lookup table, more entries of motion information may also be added into the table by derivation from the motion inforamtion. In this case, the counter associated with the look up table may be increased more than 1.
    i. In one example, the MV of an entry of motion information is scaled and put into the table;
    ii. In one example, the MV of an entry of motion information is added by (dx, dy) and put into the table;
    iii. In one example, the average of MVs of two or more entries of motion information is calculated and put into the table.

[00227] Example D2: If one block is located at a picture/slice/tile border, updating of look up tables may be always disallowed.

[00228] Example D3: Motion information of above LCU rows may be disabled to code the current LCU row.

a. In this case, at the beginning of a new slice/tile/LCU row, the number of available motion candidates may be reset to 0.

[00229] Example D4: At the beginning of coding a slice/tile with a new temporal layer index, the number of available motion candidates may be reset to 0.

[00230] Example D5: The look up table may be continuously updated with one slice/tile/LCU row/slices with same temporal layer index.

a. Alternatively, the look up table may be updated only after coding/decoding each S (S>=1) CTUs/CTBs/CUs/CBs or after coding/decoding a certain region (e.g., size equal to 8x8 or 16x16).
b. Alternatively, the look up table may be updated only after coding/decoding each S (S>=1) blocks (e.g., CUs/CBs) with certain modes (e.g., S inter-coded blocks). Alternatively, the look up table may be updated only after coding/decoding each S (S>=1) inter-coded blocks (e.g., CUs/CBs) which are not coded with sub-block based motion prediction/motion compensation method (e.g., not coded with affine and/or ATMVP mode).
c. Alternatively, the look up table may be updated only when the left-top coordinate of the coded/decoded block satisfies some conditions. For example,

the look up table is updated only when (x&M==0)&&(y&M==0), where (x, y) is left-top coordinate of the coded/decoded block. M is an integer such as 2, 4, 8, 16, 32, or 64.

d. Alternatively, one look up table may stop updating once it reaches a maximumly allowed counter.
e. In one example, the counter may be predefined. Alternatively, it be signalled 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.

[00231] Example D6: Look up table updating process may be invoked within different procedures.

a. In one example, for a block coded with merge mode, the look up table updating process may be invoked after decoding the merge candidate or after constructing the merge list or after decoding the motion information with and/or without refinement.
b. In one example, for a block coded with AMVP mode, the look up table updating process may be invoked after decoding the motion information with and/or without refinement.
c. When and/or how to update the look up table may depend on the coded mode, block dimension, video processing data unit, low delay check, etc.
    i. In one example, when one block is coded with AMVP mode, look up table may be directly updated without pruning.
    ii. Alternatively, when one block is coded with merge mode, look up table may be updated with pruning.
    iii. Alternatively, when one block is coded with merge mode and its motion information is derived from spatial and/or temporal blocks, look up table may be updated with pruning.
    iv. Alternatively, when one block is coded with merge mode and its motion information is derived from motion candidates in a look up table, look up table may be reordered without pruning.

v. Alternatively, when one block is coded with merge mode and its motion information is derived from virtual candidates (e.g., combined bi, pairwise, zero motion vector candidates) in a look up table, look up table may not be updated.

vi. Alternatively, when one block is coded with sub-block merge mode and/or triangular merge mode, look up table may not be updated.

vii. Alternatively, when one block is coded with the merge with motion vector differences (MMVD) mode and its motion information is derived from spatial and/or temporal blocks, look up table may be directly updated.

viii. In one example, when one block is coded with illumination compensation (IC) mode and/or Overlapped Block Motion Compensation (OBMC) mode and/or Decode-side Motion Vector Derivation (DMVD) mode, look up table may not be updated. Alternatively, when one block is coded with such a mode, look up table may be updated.

[00232] Example D7: Whether to reset the look up tables may further depend on the indication of enabling (or disabling) prediction crossing tiles.

a. In one example, if such an indication indicates prediction crossing tiles is disallowed, look up tables have to be reset (e.g., emptied) before coding a new tile.

b. In one example, such an indication indicates prediction crossing tiles is allowed, there is no need to reset look up tables (e.g., emptied) before coding a new tile. That is, the look up tables for coding a second tile may depend on those tables used for coding a first tile.

[00233] Example D8: Whether to enable adaptive loop filter (ALF) temporal prediction (or prediction/inheritance of ALF filters from a different tile) may further depend on the indication of enabling (or disabling) prediction crossing tiles.

a. In one example, if such an indication indicates prediction crossing tiles is disallowed, ALF temporal prediction is disallowed.

b. In one example, such an indication indicates prediction crossing tiles is allowed, ALF temporal prediction may be enabled.

[00234] Example D9: When shared merge list (or other kinds of shared motion candidates list/or shared intra mode information or other shared information) is enabled for a merge sharing node, the updating of look up tables may be performed in the following ways:

a. One of representative coded block (e.g., leaf coding unit) may be selected to update the look up tables.
    ii. In one example, it is defined as the last coded block in decoding order.
    iii. Alternatively, it is defined as the last coded block that satisfy the conditions for look up table updating (such as the last coded block with non-affine and non-ATMVP inter mode).
    iv. In one example, it is defined as the first coded block in decoding order under the parent node.
    v. Alternatively, it is defined as the first coded block that satisfy the conditions for look up table updating (such as the last coded block with non-affine and non-ATMVP, non-triangular inter mode) under the parent node.

b. More than one representative coded blocks (e.g., leaf coding unit) may be selected to update the look up tables.
    vi. In one example, multiple sets of coded information associated with those representative coded blocks may be used to update the look up tables wherein multiple entries of a look up table may be updated accordingly.
    vii. In one example, multiple sets of coded information associated with those representative coded blocks may be used to update the look up tables wherein multiple look up tables may be updated accordingly.
    viii. In one example, those representative coded blocks may be checked in a certain order to determinate whether to be used to update one or multiple look up tables.

ix. In one example, those representative coded blocks may be defined as the first and the last coded block under the parent node.

c. Updating of look up tables may be always disabled.

[00235] Example D10: For all above bullets, the look up tables indicate the coded information or information derived from coded information from previously coded blocks in a decoding order.

a. A look up table may include the translational motion information, or affine motion information, or affine model parameters, or intra mode information, or illumination compensation information, etc. al.

b. Alternatively, a look up table may include at least two kinds of information, such as translational motion information, or affine motion information, or affine model parameters, or intra mode information, or illumination compensation information, etc. al.

[00236] **Additional Example Embodiments**

[00237] A history-based MVP (HMVP) method is proposed wherein a HMVP candidate is defined as the motion information of a previously coded block. A table with multiple HMVP candidates is maintained during the encoding/decoding process. The table is emptied when a new slice is encountered. Whenever there is an inter-coded block, the associated motion information is added to the last entry of the table as a new HMVP candidate. The overall coding flow is depicted in FIG. 31.

[00238] In one example, the table size is set to be L (e.g., L = 16 or 6, or 44), which indicates up to L HMVP candidates may be added to the table.

[00239] In one embodiment (corresponding to example 11.g.i), if there are more than L HMVP candidates from the previously coded blocks, a First-In-First-Out (FIFO) rule is applied so that the table always contains the latest previously coded L motion candidates. FIG. 32 depicts an example wherein the FIFO rule is applied to remove a HMVP candidate and add a new one to the table used in the proposed method.

[00240] In another embodiment (corresponding to invention 11.g.iii), whenever adding a new motion candidate (such as the current block is inter-coded and non-affine mode), a redundancy checking process is applied firstly to identify whether there are identical or similar motion candidates in LUTs.

[00241] Some examples are depicted as follows:

[00242] FIG. 33A shows an example when the LUT is full before adding a new motion candidate.

[00243] FIG. 33B shows an example when the LUT is not full before adding a new motion candidate.

[00244] FIG. 33A and 33B together show an example of redundancy-removal based LUT updating method (with one redundancy motion candidate removed).

[00245] FIG. 34A and 34B show example implementation for two cases of the redundancy-removal based LUT updating method (with multiple redundancy motion candidates removed, 2 candidates in the figures)

[00246] FIG. 34A shows an example case of when the LUT is full before adding a new motion candidate.

[00247] FIG. 34B shows an example case of When the LUT is not full before adding a new motion candidate

[00248] HMVP candidates could be used in the merge candidate list construction process. All HMVP candidates from the last entry to the first entry (or the last *K*0 HMVP, e.g., *K*0 equal to 16 or 6) in the table are inserted after the TMVP candidate. Pruning is applied on the HMVP candidates. Once the total number of available merge candidates reaches the signaled maximally allowed merge candidates, the merge candidate list construction process is terminated. Alternatively, once the total number of added motion candidates reaches a given value, the fetching of motion candidates from LUTs is terminated.

[00249] Similarly, HMVP candidates could also be used in the AMVP candidate list construction process. The motion vectors of the last *K1* HMVP candidates in the table are inserted after the TMVP candidate. Only HMVP candidates with the same reference picture as the AMVP target reference picture are used to construct the AMVP candidate list. Pruning is applied on the HMVP candidates. In one example, *K1* is set to 4.

[00250] **References**

[1] "Overview of the High Efficiency Video Coding (HEVC) Standard", Gary J. Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 22, No. 12, December 2012.

[2] "Overview of the H.264/AVC video coding standard", Ajay Luthra, Pankaj Topiwala, Proceedings of SPIE Vol. 5203 Applications of Digital Image Processing XXVI.

[3] J. Chen, E. Alshina, G. J. Sullivan, J.-R. Ohm, J. Boyce, "Algorithm description of Joint Exploration Test Model 7 (JEM7)," JVET-G1001, Aug. 2017.

[4] JEM-7.0: https://jvet.hhi.fraunhofer.de/svn/svn_HMJEMSoftware/tags/ HM-16.6-JEM-7.0.

[5] H.265/HEVC, https://www.itu.int/rec/T-REC-H.265

[6] "Multi-Type-Tree", JVET-D0117

[7] International Patent Application WO2016/091161

[8] "Description of SDR, HDR and 360° video coding technology proposal by Qualcomm and Technicolor – low and high complexity versions", JVET-J0021.

[00251] FIG. 28A is a block diagram of a video processing apparatus 2800. The apparatus 2800 may be used to implement one or more of the methods described herein. The apparatus 2800 may be embodied in a smartphone, tablet, computer, Internet of Things (IoT) receiver, and so on. The apparatus 2800 may include one or more processors 2802, one or more memories 2804 and video processing hardware 2806. The processor(s) 2802 may be configured to implement one or more methods described in the present document. The memory (memories) 2804 may be used for storing data and code used for implementing the methods and techniques described herein. The video processing hardware 2806 may be used to implement, in hardware circuitry, some techniques described in the present document.

[00252] FIG. 28B is another example of a block diagram of a video processing system in which disclosed techniques may be implemented. FIG. 28B is a block diagram showing an example video processing system 3100 in which various techniques disclosed herein may be implemented. Various implementations may include some or all of the components of the system 3100. The system 3100 may include input 3102 for receiving video content. The video content may be received in a raw or uncompressed format, e.g., 8 or 10 bit multi-component pixel values, or may be in a compressed or encoded format. The input 3102 may represent a network interface, a peripheral bus interface, or a storage interface. Examples of network interface include wired interfaces such as Ethernet, passive optical network (PON), etc. and wireless interfaces such as Wi-Fi or cellular interfaces.

[00253] The system 3100 may include a coding component 3104 that may implement the various coding or encoding methods described in the present document. The coding component

3104 may reduce the average bitrate of video from the input 3102 to the output of the coding component 3104 to produce a coded representation of the video. The coding techniques are therefore sometimes called video compression or video transcoding techniques. The output of the coding component 3104 may be either stored, or transmitted via a communication connected, as represented by the component 3106. The stored or communicated bitstream (or coded) representation of the video received at the input 3102 may be used by the component 3108 for generating pixel values or displayable video that is sent to a display interface 3110. The process of generating user-viewable video from the bitstream representation is sometimes called video decompression. Furthermore, while certain video processing operations are referred to as "coding" operations or tools, it will be appreciated that the coding tools or operations are used at an encoder and corresponding decoding tools or operations that reverse the results of the coding will be performed by a decoder.

[00254] Examples of a peripheral bus interface or a display interface may include universal serial bus (USB) or high definition multimedia interface (HDMI) or Displayport, and so on. Examples of storage interfaces include SATA (serial advanced technology attachment), PCI, IDE interface, and the like. The techniques described in the present document may be embodied in various electronic devices such as mobile phones, laptops, smartphones or other devices that are capable of performing digital data processing and/or video display.

[00255] 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. 28A or 28B.

[00256] FIG. 29 is a flowchart for an example of a video decoding method 2900. The method 2900 includes, at step 2902, determining, whether to use a temporal prediction for obtaining an adaptive loop filter for a conversion between a current video block of a video and a coded representation of the video, based on whether a cross-tile prediction is enabled. The method 2900 further includes, at step 2904, performing the conversion using an adaptive loop filter that is obtained based on the determining.

[00257] FIG. 30 shows a flowchart for an example of video decoding method 3000. The method 3000 includes, at step 3002, determining whether a sharing of merge list information is enabled for a merge sharing node that corresponds to an ancestor node in a coding unit split tree to allow leaf coding units of the ancestor node in the coding unit split tree to use the merge list information. The method 3000 further includes, at step 3004, performing a conversion between a current video block of a video and a coded representation of the video based on the determining.

[00258] With respect to methods 2900 and 3000, in some embodiments, the motion information includes at least one of: a prediction direction, a reference picture index, motion vector values, intensity compensation flag, affine flag, motion vector difference precision, and motion vector difference value. Further, the motion information may further include block position information indicating source of the motion information. In some embodiments, the video block may be a CU or a PU and the portion of video may correspond to one or more video slices or one or more video pictures.

[00259] In some embodiments, each LUT includes an associated counter, wherein the counter is initialized to a zero value at beginning of the portion of video and increased for each encoded video region in the portion of the video. The video region comprises one of a coding tree unit, a coding tree block, a coding unit, a coding block or a prediction unit. In some embodiments, the counter indicates, for a corresponding LUT, a number of motion candidates that were removed from the corresponding LUT. In some embodiments, the set of motion candidates may have a same size for all LUTs. In some embodiments, the portion of video corresponds to a slice of video, and wherein the number of LUTs is equal to N*P, wherein N is an integer representing LUTs 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. Additional details of the methods 2900 and 3000 are described in the examples provided in Section 4 and the examples listed below.

[00260] 1. A method for video processing, comprising: determining, whether to use a temporal prediction for obtaining an adaptive loop filter for a conversion between a current video block of a video and a coded representation of the video, based on whether a cross-tile prediction is enabled; and performing the conversion using an adaptive loop filter that is obtained based on the determining.

[00261] 2. The method of example 1, wherein the temporal prediction is not used for obtaining the adaptive loop filter due to the cross-tile prediction not being enabled.

[00262] 3. The method of example 1, wherein the temporal prediction is used for obtaining the adaptive loop filter due to the cross-tile prediction being allowed.

[00263] 4. A method for video processing, comprising: determining whether a sharing of merge list information is enabled for a merge sharing node that corresponds to an ancestor node in a coding unit split tree to allow leaf coding units of the ancestor node in the coding unit split tree to use the merge list information; and performing a conversion between a current video block of a video and a coded representation of the video based on the determining.

[00264] 5. The method of example 4, wherein updating a motion candidate table is enabled or disabled during the conversion.

[00265] 6. The method of example 4, wherein the merge list information includes at least one of a merge list, a motion candidate list, an intra mode information, or any typed information.

[00266] 7. The method of example 1, wherein a single representative coded block is selected to update a motion candidate table.

[00267] 8. The method of example 7, wherein the single representative coded block is selected as a last coded block according to a decoding order.

[00268] 9. The method of example 7, wherein the single representative coded block is selected as a last coded block that that satisfies a condition for updating the motion candidate table.

[00269] 10. The method of example 9, wherein the condition for updating the motion candidate table requires that the last coded block is coded with a non-affine and non-ATMVP (alternative temporal motion vector prediction) inter mode, wherein the ATMVP mode modifies motion vectors temporal motion vector prediction (TMVP) by fetching multiple sets of motion information from blocks smaller than a current coding unit including the current video block.

[00270] 11. The method of example 7, wherein the single representative coded block is selected as a first coded block according to a decoding order under a parent node.

[00271] 12. The method of example 7, wherein the single representative coded block is selected as a first coded block that that satisfies a condition for updating the motion candidate table.

[00272] 13. The method of example 12, wherein the condition for updating the motion candidate table requires that the first coded block is coded with a non-affine and non-ATMVP (alternative temporal motion vector prediction) inter mode, wherein the ATMVP mode modifies motion vectors temporal motion vector prediction (TMVP) by fetching multiple sets of motion information from blocks smaller than a current coding unit including the current video block.

[00273] 14. The method of example 1, wherein multiple representative coded blocks are selected to update a motion candidate table.

[00274] 15. The method of example 14, wherein multiple sets of coded information associated with the multiple representative coded blocks are used to update multiple entries of the motion candidate table.

[00275] 16. The method of example 14, wherein the performing of the conversion further includes updating an additional motion candidate table and wherein multiple sets of coded information associated with the multiple representative coded blocks are used to update the motion candidate table and the additional motion candidate table.

[00276] 17. The method of example 14, wherein the multiple representative coded blocks are checked in a certain order to determine whether to be used to update the motion candidate table.

[00277] 18. The method of example 14, wherein the multiple representative coded blocks are selected as a first or last coded block under a parent node.

[00278] 19. The method of example 2, wherein updating of a motion candidate table is always disabled.

[00279] 20. The method of any of examples 2-16, wherein the motion candidate table indicates coded information or information derived from coded information from previously coded blocks in a decoding order.

[00280] 21. The method of example 20, wherein the motion candidate table includes at least one of translational motion information, affine motion information, affine model parameters, intra mode information, or illumination compensation information.

[00281] 22. The method of any of examples 1 to 21, wherein the performing of the conversion includes generating the coded representation from the current video block.

[00282] 23. The method of any of examples 1 to 21, wherein the performing of the conversion includes generating the current video block from the coded representation.

[00283] 24. 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 examples 1 to 23.

[00284] 25. 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 examples 1 to 23.

[00285] 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.

[00286] The disclosed and other embodiments, modules and the functional operations described in this document can be implemented in digital electronic circuitry, or in computer software, firmware, or hardware, including the structures disclosed in this document and their structural equivalents, or in combinations of one or more of them. The disclosed and other embodiments can be implemented as one or more computer program products, i.e., one or more modules of computer program instructions encoded on a 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 them. The term “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. A propagated signal is an artificially generated signal, e.g., a machine-generated electrical, optical, or electromagnetic signal, that is generated to encode information for transmission to suitable receiver apparatus.

[00287] 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.

[00288] The processes and logic flows described in this document 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).

[00289] 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 non-volatile memory, media and memory devices, including by way of example semiconductor memory devices, e.g., EPROM, EEPROM, and flash memory devices; magnetic disks, e.g., internal hard disks or removable disks; magneto optical disks; and CD ROM and DVD-ROM disks. The processor and the memory can be supplemented by, or incorporated in, special purpose logic circuitry.

[00290] 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.

[00291] 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.

[00292] 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 method for video processing, comprising:

determining, whether to use a temporal prediction for obtaining an adaptive loop filter for a conversion between a current video block of a video and a coded representation of the video, based on whether a cross-tile prediction is enabled; and

performing the conversion using an adaptive loop filter that is obtained based on the determining.

2. The method of claim 1, wherein the temporal prediction is not used for obtaining the adaptive loop filter due to the cross-tile prediction not being enabled.

3. The method of claim 1, wherein the temporal prediction is used for obtaining the adaptive loop filter due to the cross-tile prediction being allowed.

4. A method for video processing, comprising:

determining whether a sharing of merge list information is enabled for a merge sharing node that corresponds to an ancestor node in a coding unit split tree to allow leaf coding units of the ancestor node in the coding unit split tree to use the merge list information; and

performing a conversion between a current video block of a video and a coded representation of the video based on the determining.

5. The method of claim 4, wherein updating a motion candidate table is enabled or disabled during the conversion.

6. The method of claim 4, wherein the merge list information includes at least one of a merge list, a motion candidate list, an intra mode information, or any typed information.

7. The method of claim 1, wherein a single representative coded block is selected to update a motion candidate table.

8. The method of claim 7, wherein the single representative coded block is selected as a last coded block according to a decoding order.

9. The method of claim 7, wherein the single representative coded block is selected as a last coded block that that satisfies a condition for updating the motion candidate table.

10. The method of claim 9, wherein the condition for updating the motion candidate table requires that the last coded block is coded with a non-affine and non-ATMVP (alternative temporal motion vector prediction) inter mode, wherein the ATMVP mode modifies motion vectors temporal motion vector prediction (TMVP) by fetching multiple sets of motion information from blocks smaller than a current coding unit including the current video block.

11. The method of claim 7, wherein the single representative coded block is selected as a first coded block according to a decoding order under a parent node.

12. The method of claim 7, wherein the single representative coded block is selected as a first coded block that that satisfies a condition for updating the motion candidate table.

13. The method of claim 12, wherein the condition for updating the motion candidate table requires that the first coded block is coded with a non-affine and non-ATMVP (alternative temporal motion vector prediction) inter mode, wherein the ATMVP mode modifies motion vectors temporal motion vector prediction (TMVP) by fetching multiple sets of motion information from blocks smaller than a current coding unit including the current video block.

14. The method of claim 1, wherein multiple representative coded blocks are selected to update a motion candidate table.

15. The method of claim 14, wherein multiple sets of coded information associated with the multiple representative coded blocks are used to update multiple entries of the motion candidate table.

16. The method of claim 14, wherein the performing of the conversion further includes updating an additional motion candidate table and wherein multiple sets of coded information associated with the multiple representative coded blocks are used to update the motion candidate table and the additional motion candidate table.

17. The method of claim 14, wherein the multiple representative coded blocks are checked in a certain order to determine whether to be used to update the motion candidate table.

18. The method of claim 14, wherein the multiple representative coded blocks are selected as a first or last coded block under a parent node.

19. The method of claim 2, wherein updating of a motion candidate table is always disabled.

20. The method of any of claims 2-16, wherein the motion candidate table indicates coded information or information derived from coded information from previously coded blocks in a decoding order.

21. The method of claim 20, wherein the motion candidate table includes at least one of translational motion information, affine motion information, affine model parameters, intra mode information, or illumination compensation information.

22. The method of any of claims 1 to 21, wherein the performing of the conversion includes generating the coded representation from the current video block.

23. The method of any of claims 1 to 21, wherein the performing of the conversion includes generating the current video block from the coded representation.

24. 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 to 23.

25. 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 to 23.

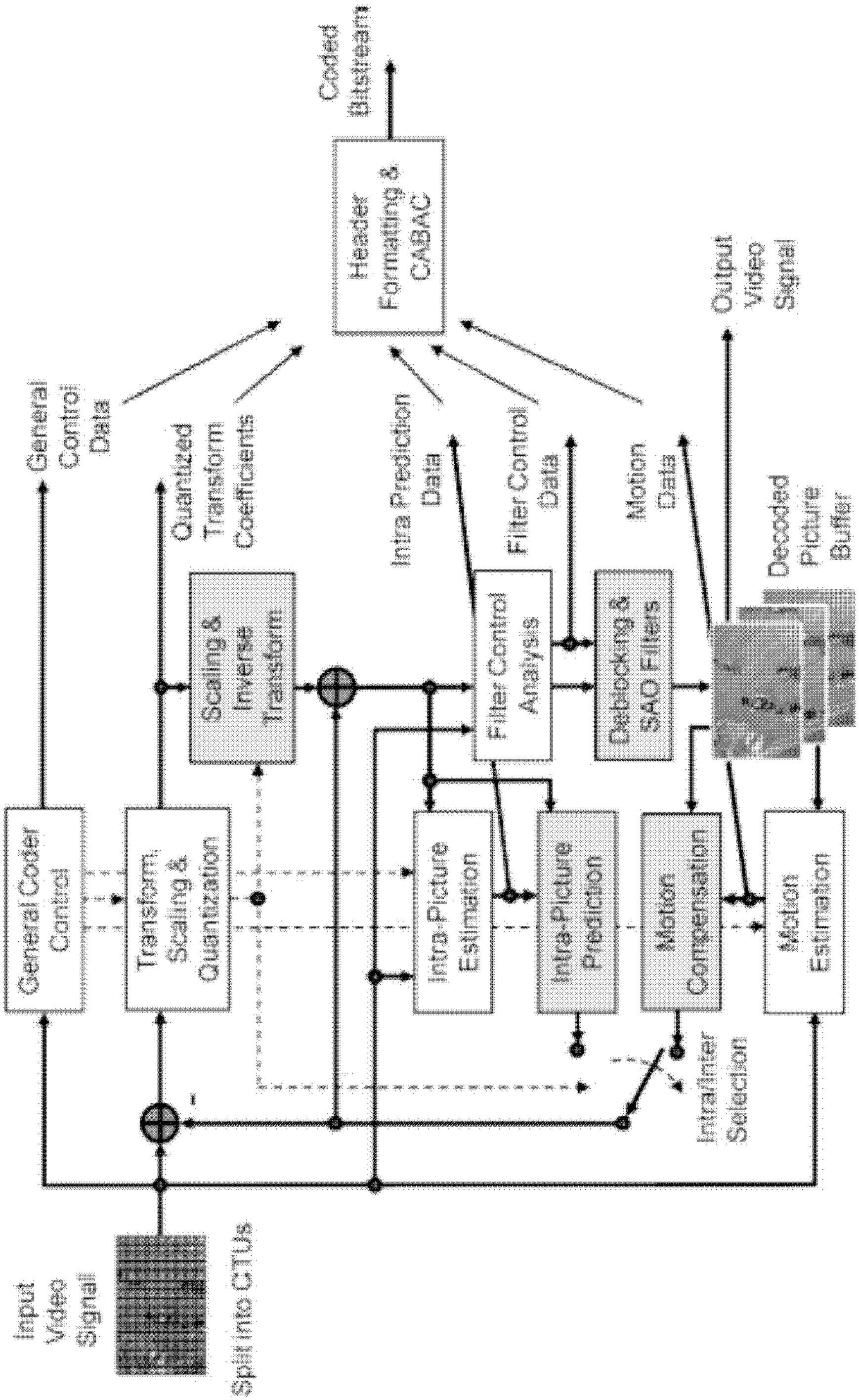


Fig. 1. Typical HEVC video encoder (with decoder modeling elements shaded in light gray).

FIG. 1

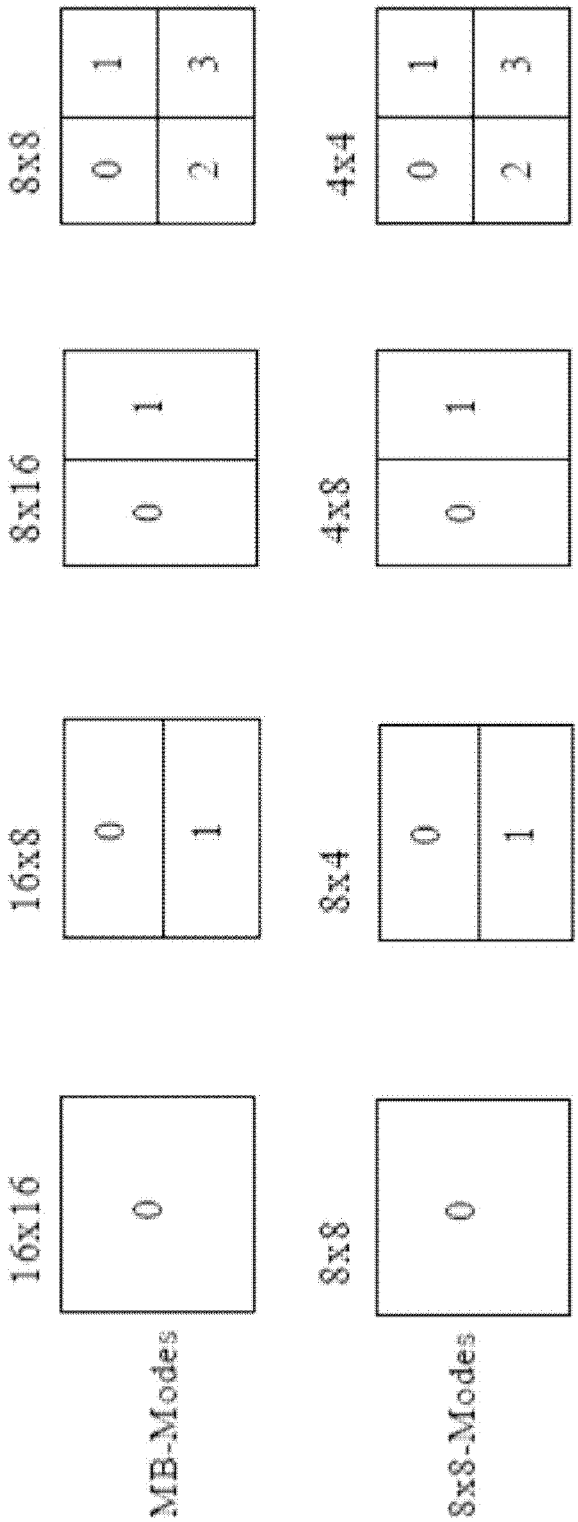

MB-Modes
16x16
0
16x8
0
1
8x16
0
1
8x8
0
1
2
3
8x8-Modes
8x8
0
8x4
0
1
4x8
0
1
4x4
0
1
2
3

FIG. 2

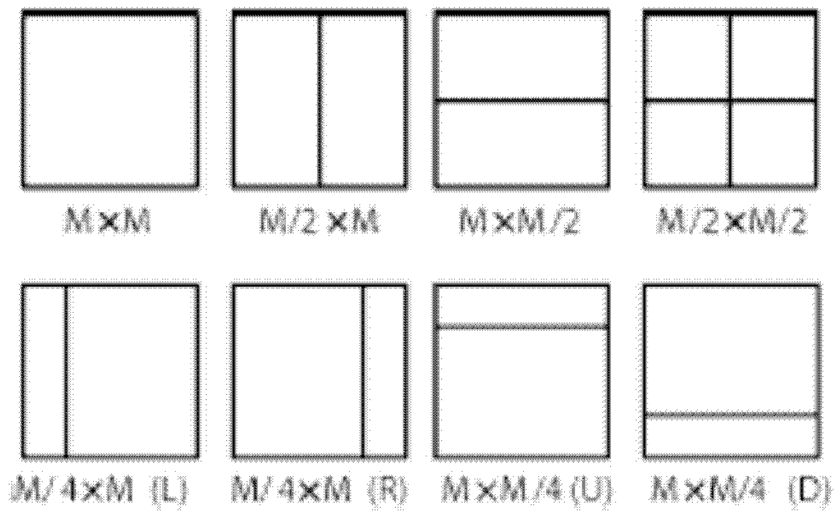


Fig. 3. Modes for splitting a CB into PBs, subject to certain size constraints. For intrapicture-predicted CBs, only $M \times M$ and $M/2 \times M/2$ are supported.

FIG. 3

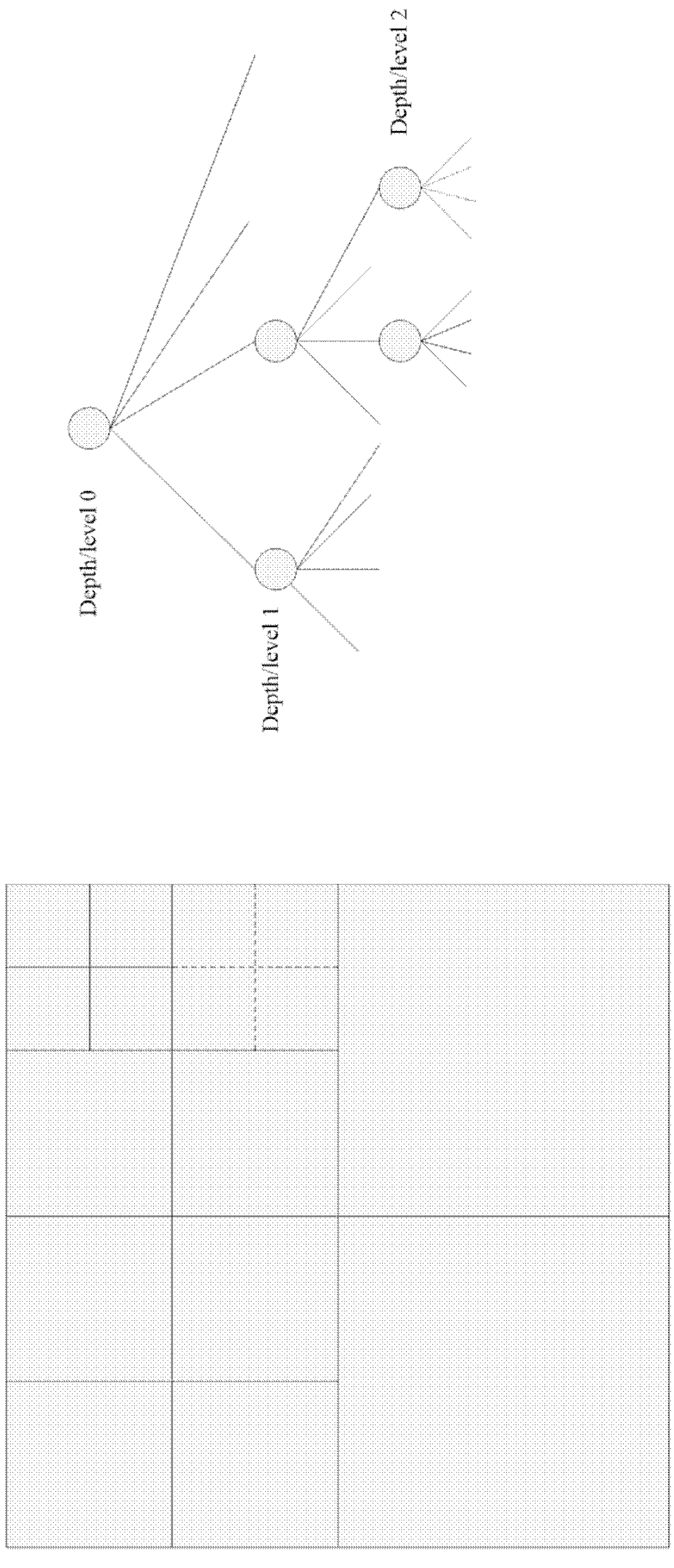


FIG. 4

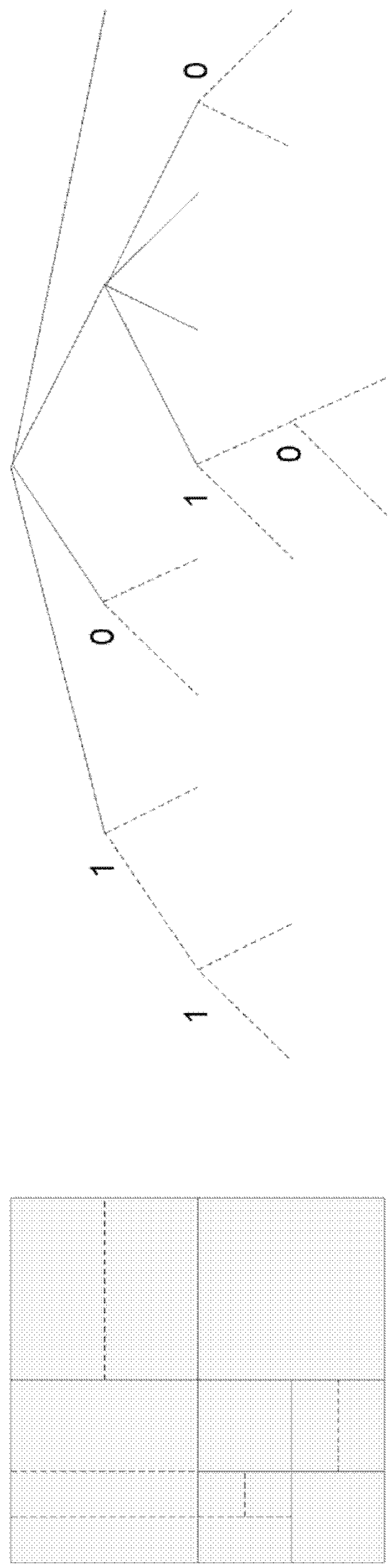


FIG. 5

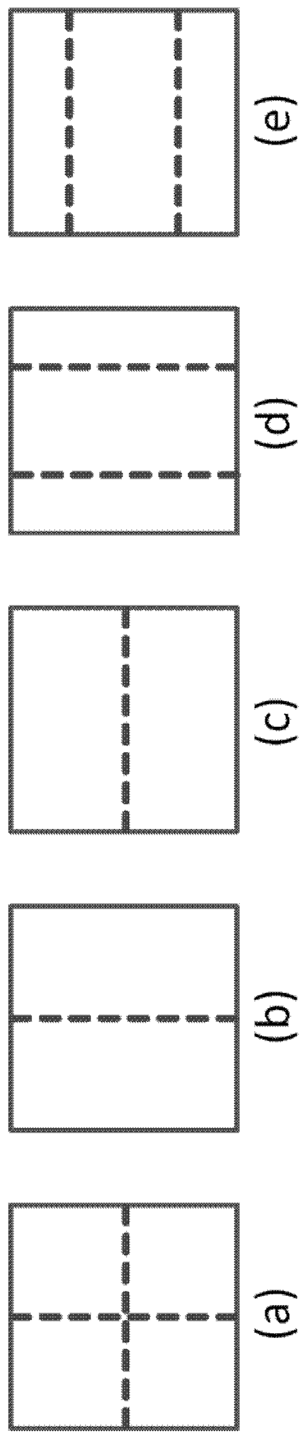


FIG. 6

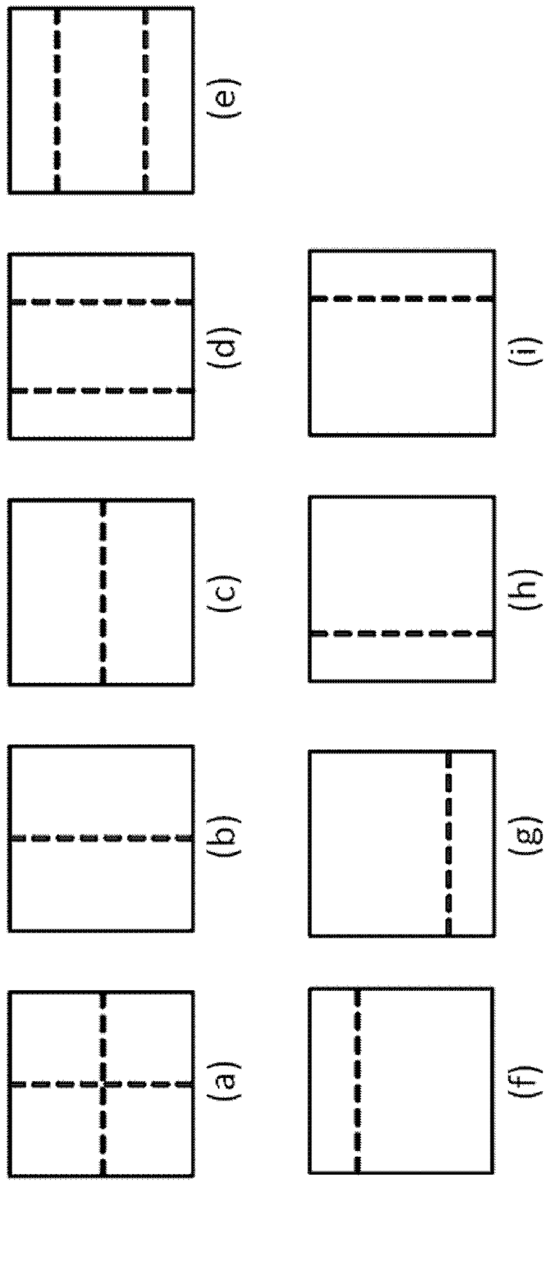


FIG. 7

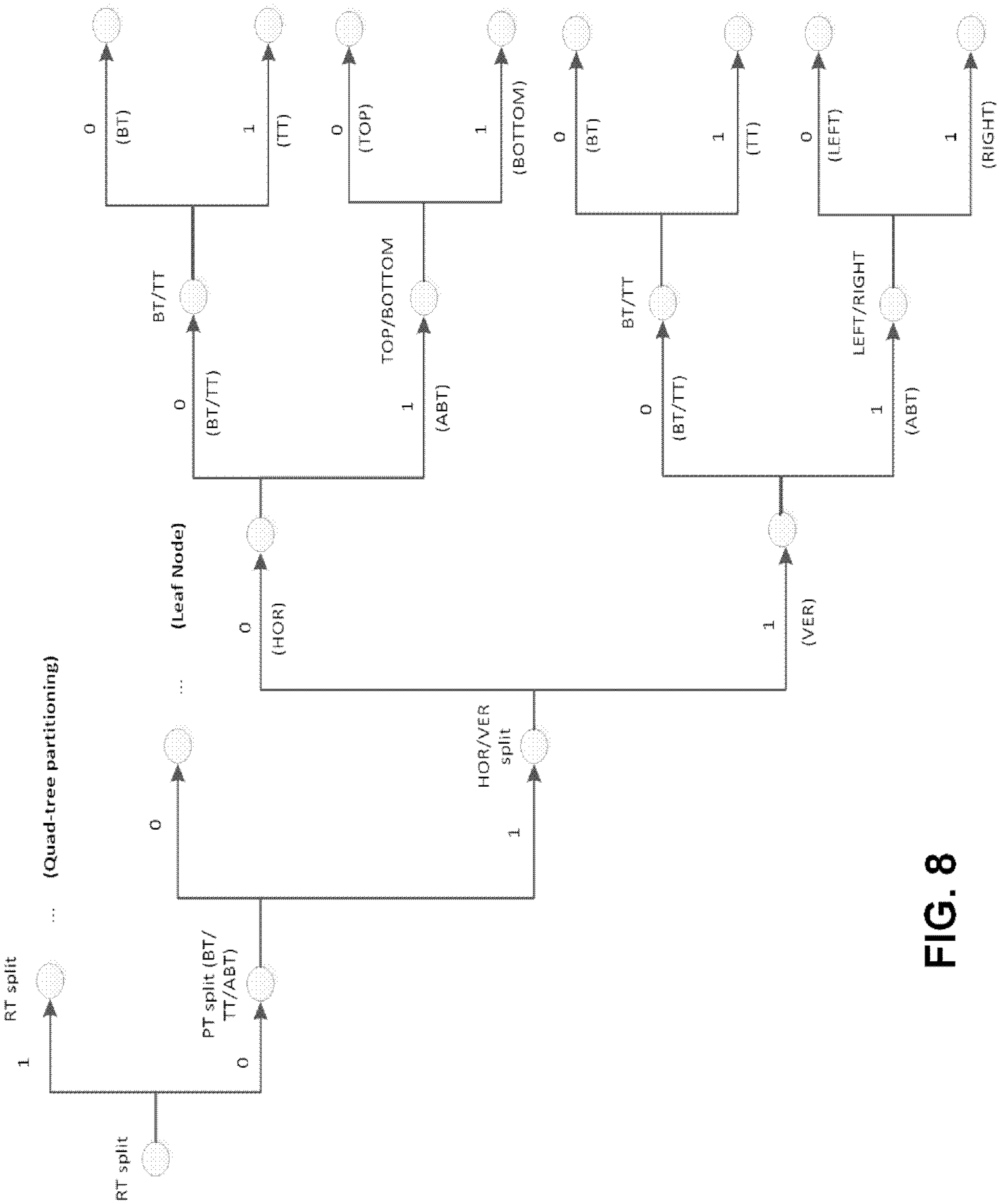

RT split
1
RT split
...
(Quad-tree partitioning)
0
PT split (BT/TT/ABT)
0
...
1
HOR/VER split
(Leaf Node)
0
(HOR)
1
(VER)
0
(BT/TT)
1
(ABT)
0
(BT/TT)
1
(ABT)
BT/TT
TOP/BOTTOM
BT/TT
LEFT/RIGHT
0
(BT)
1
(TT)
0
(TOP)
1
(BOTTOM)
0
(BT)
1
(TT)
0
(LEFT)
1
(RIGHT)

FIG. 8

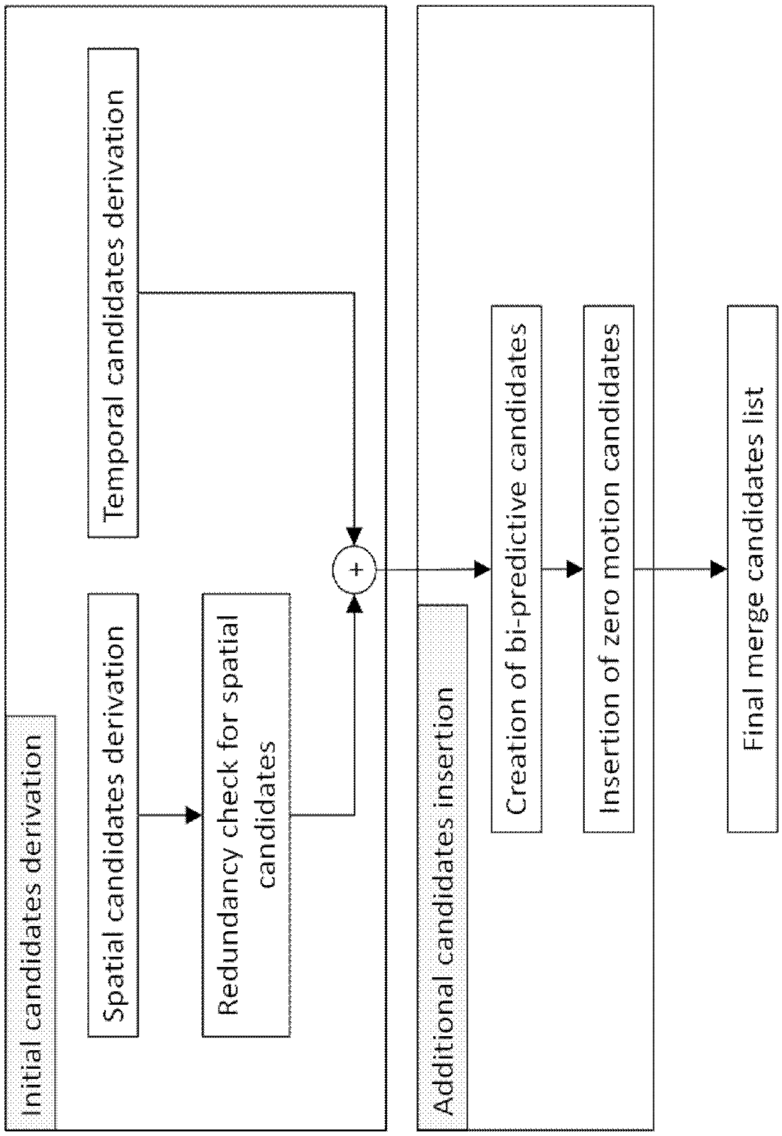

Initial candidates derivation
Spatial candidates derivation
Temporal candidates derivation
Redundancy check for spatial candidates
+
Additional candidates insertion
Creation of bi-predictive candidates
Insertion of zero motion candidates
Final merge candidates list

FIG. 9

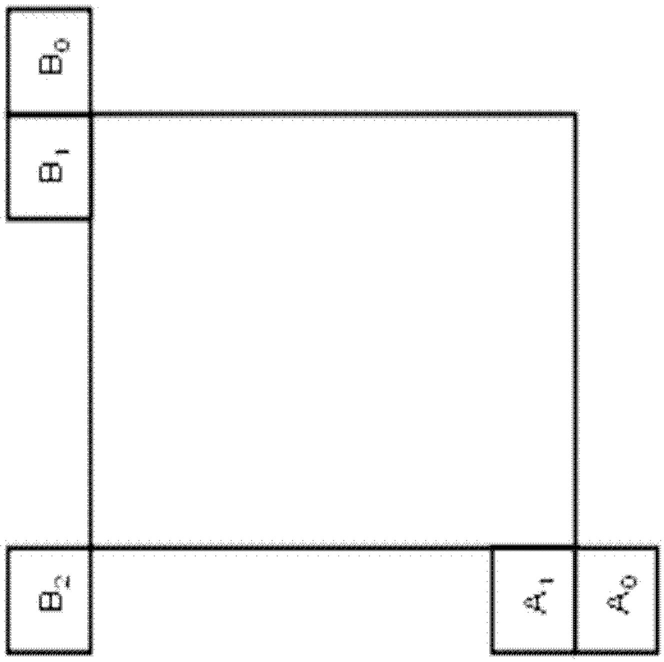


FIG. 10

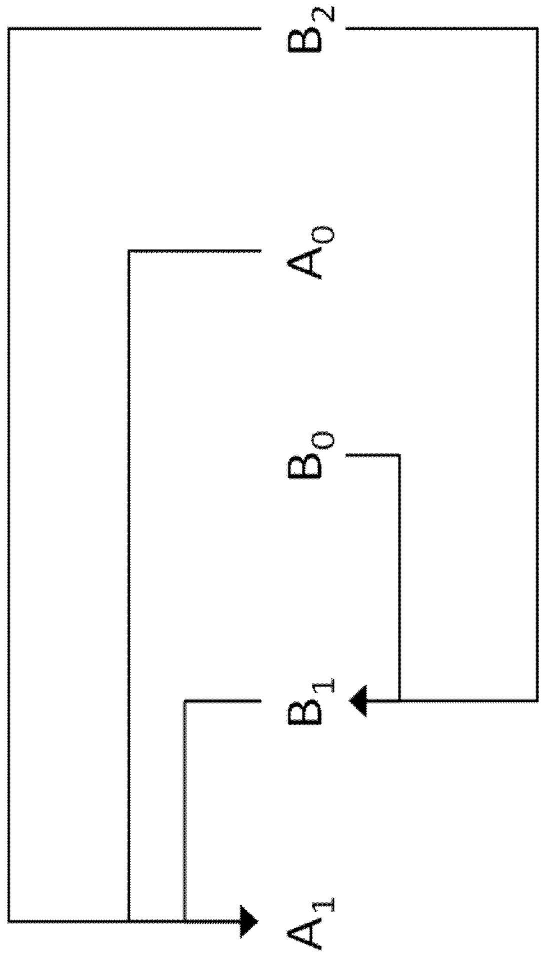
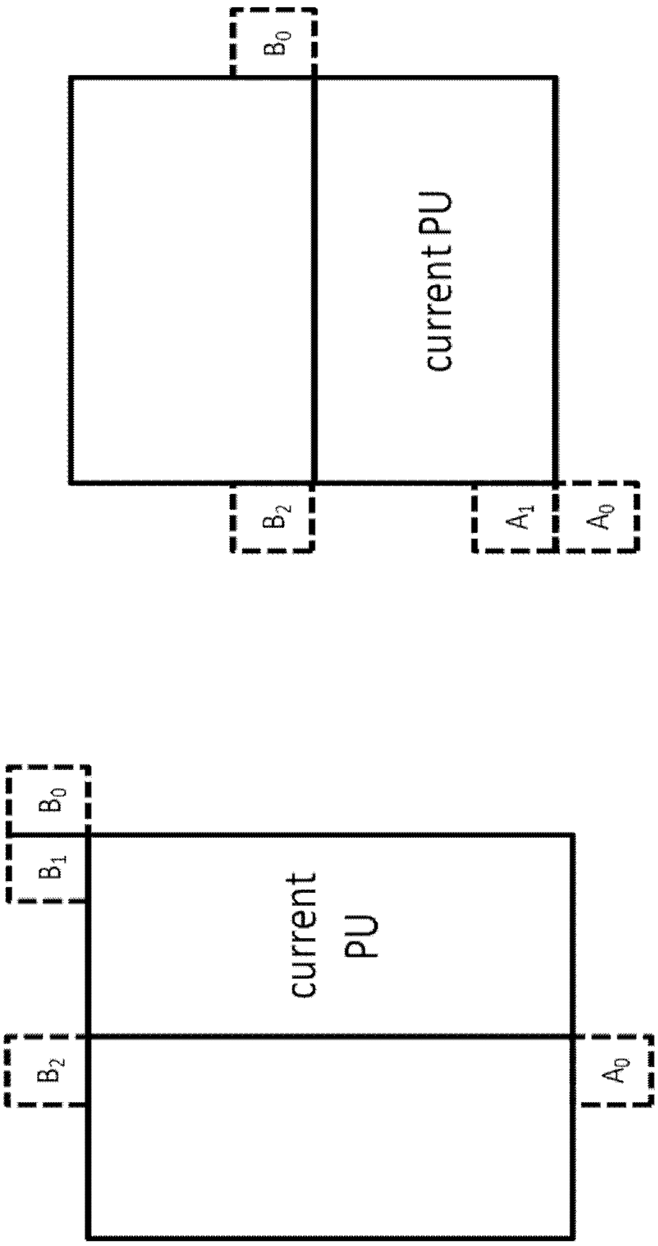


FIG. 11



(a) second PU of Nx2N

(b)second PU of 2NxN

FIG. 12

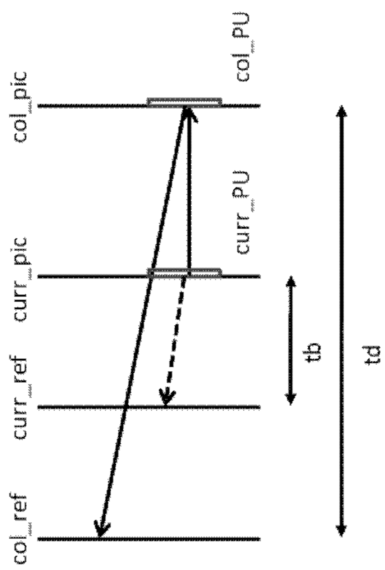


FIG. 13

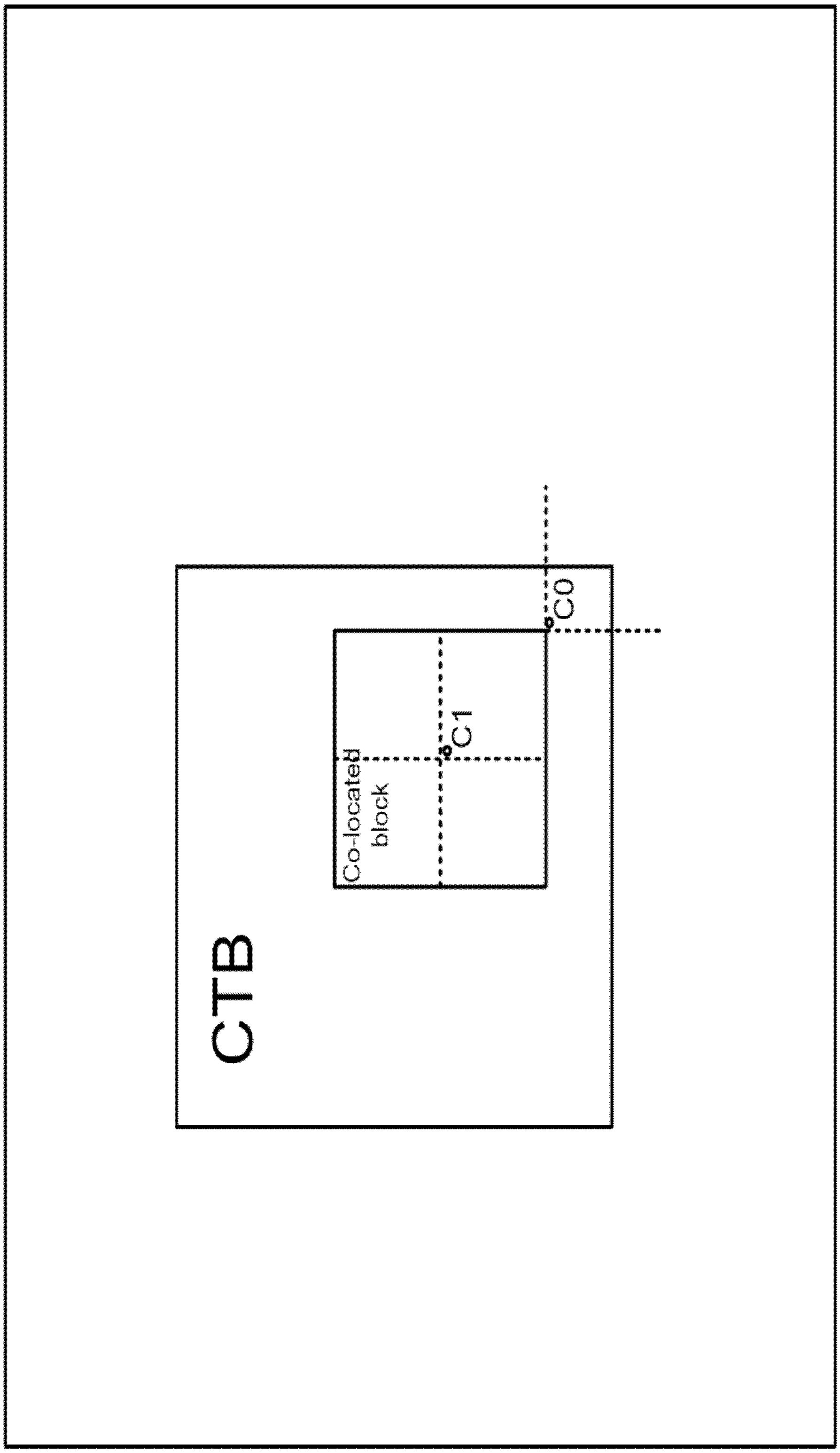


FIG. 14

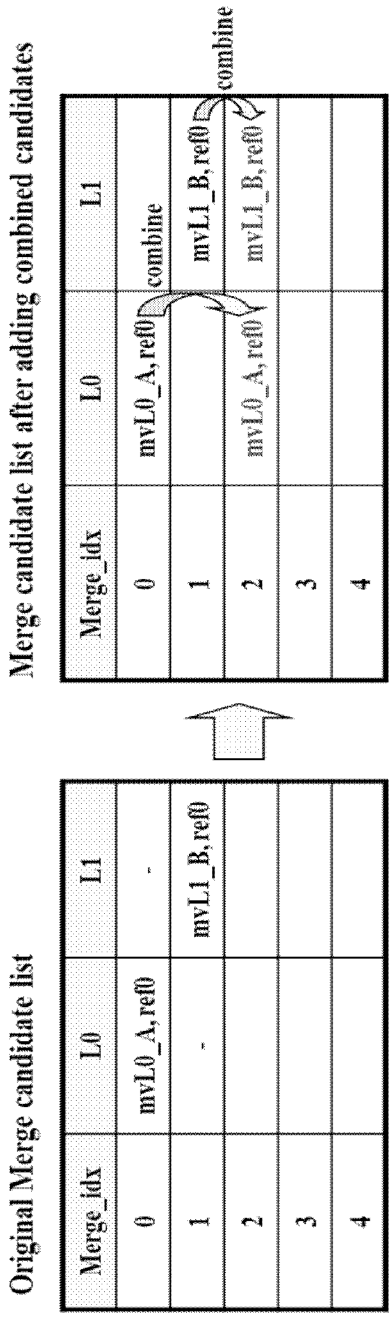

Original Merge candidate list
Merge_idx
L0
L1
0
mvL0_A, ref0
-
1
-
mvL1_B, ref0
2
3
4
Merge candidate list after adding combined candidates
Merge_idx
L0
L1
0
mvL0_A, ref0
combine
1
mvL1_B, ref0
combine
2
mvL0_A, ref0
mvL1_B, ref0
3
4

FIG. 15

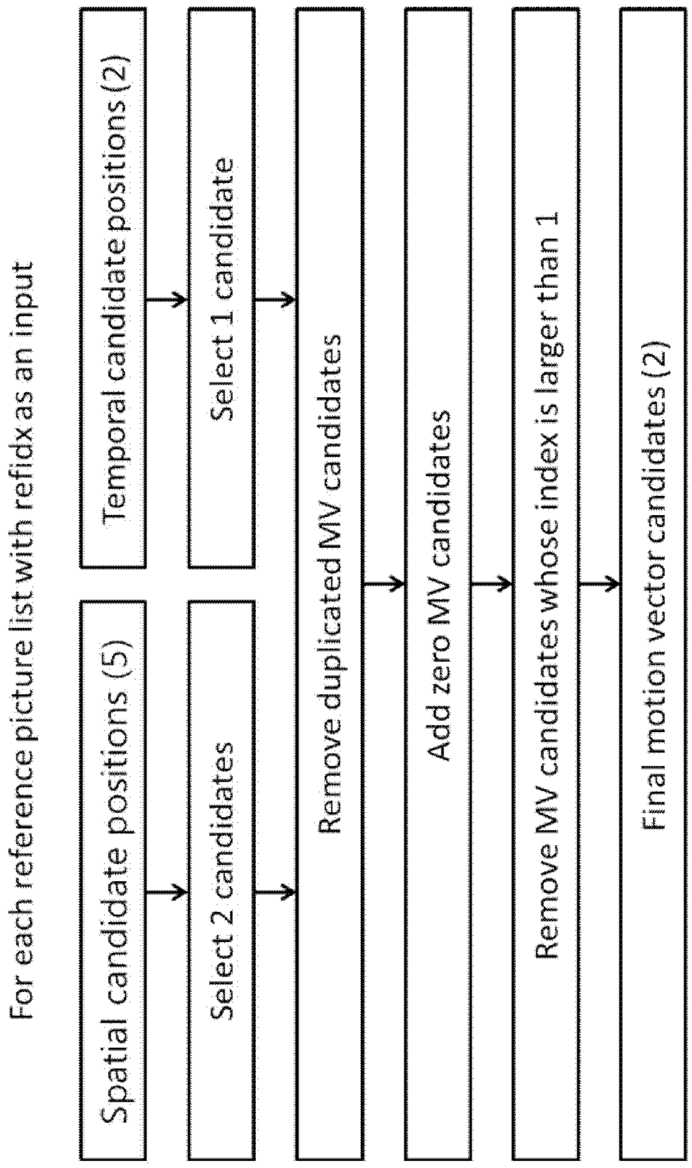

For each reference picture list with refidx as an input
Spatial candidate positions (5)
Temporal candidate positions (2)
Select 2 candidates
Select 1 candidate
Remove duplicated MV candidates
Add zero MV candidates
Remove MV candidates whose index is larger than 1
Final motion vector candidates (2)

FIG. 16

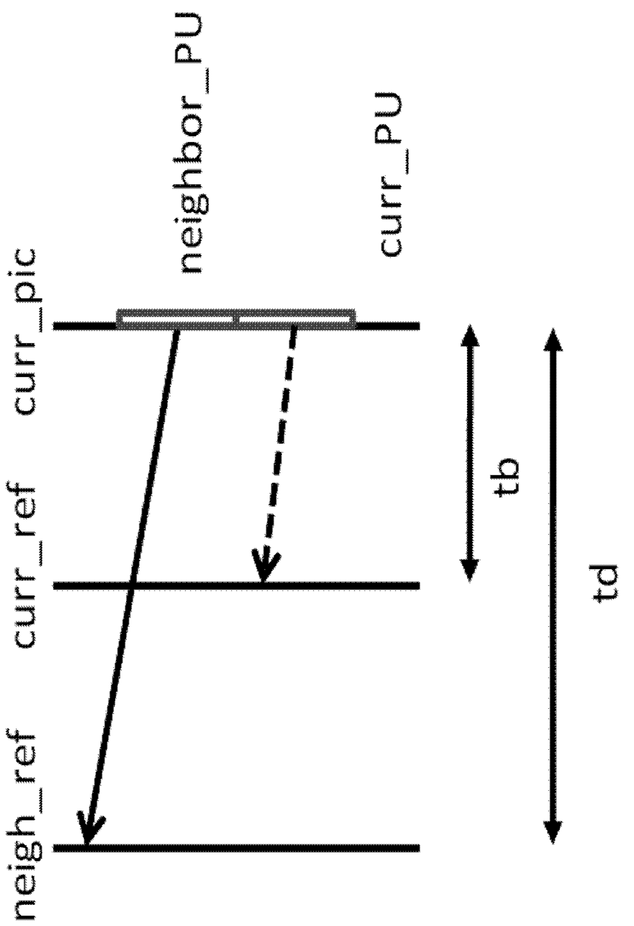


FIG. 17

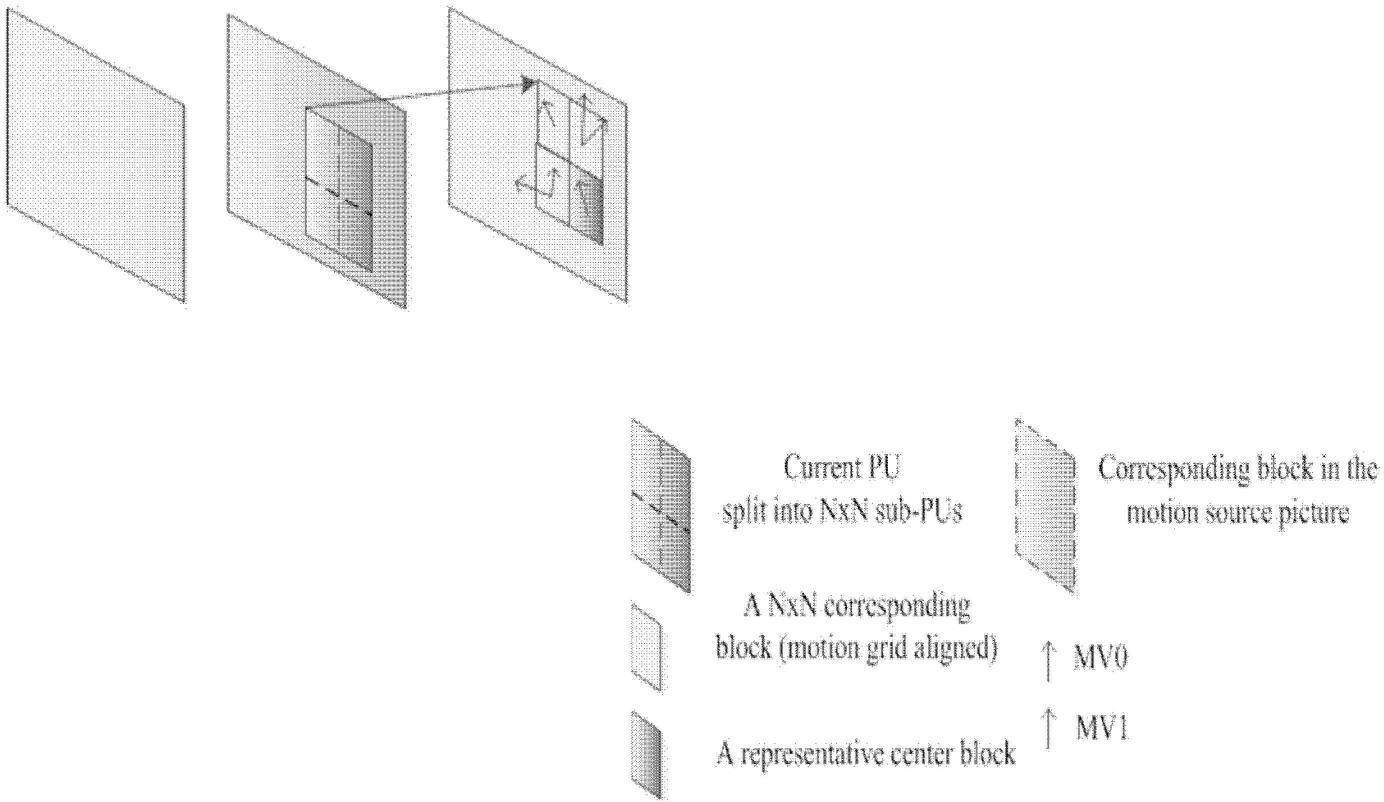

Current PU
split into NxN sub-PUs
Corresponding block in the
motion source picture
A NxN corresponding
block (motion grid aligned)
MV0
MV1
A representative center block

FIG. 18

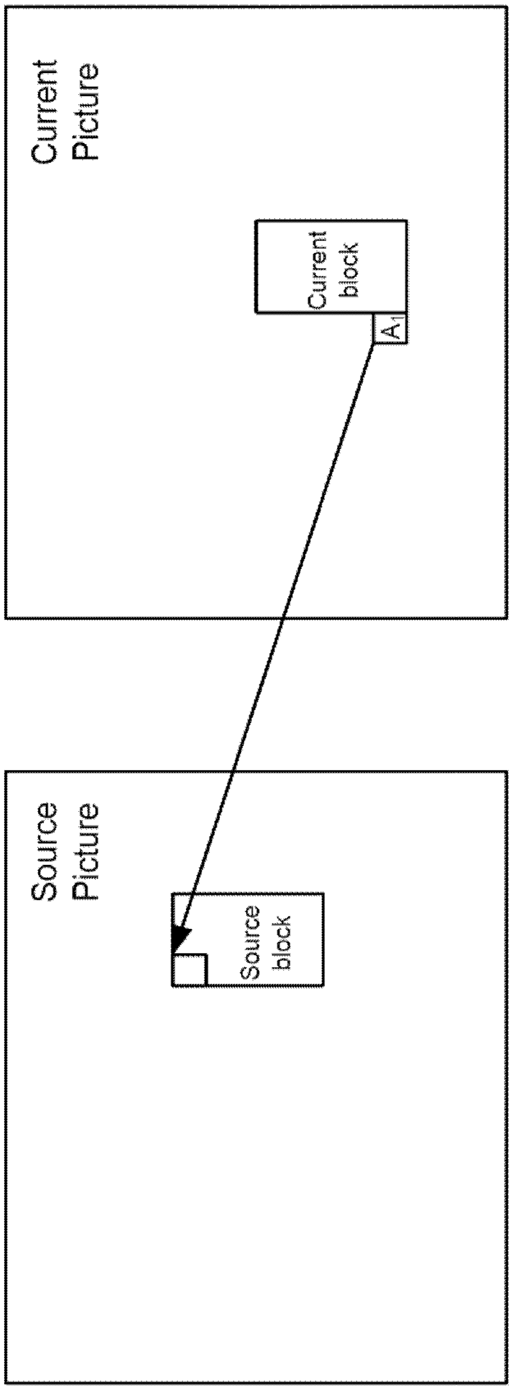

Current Picture
Current block
A_1
Source Picture
Source block

FIG. 19

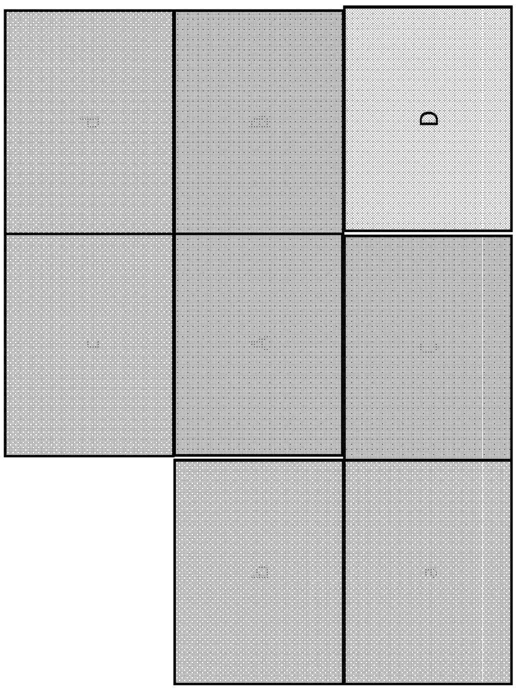


FIG. 20

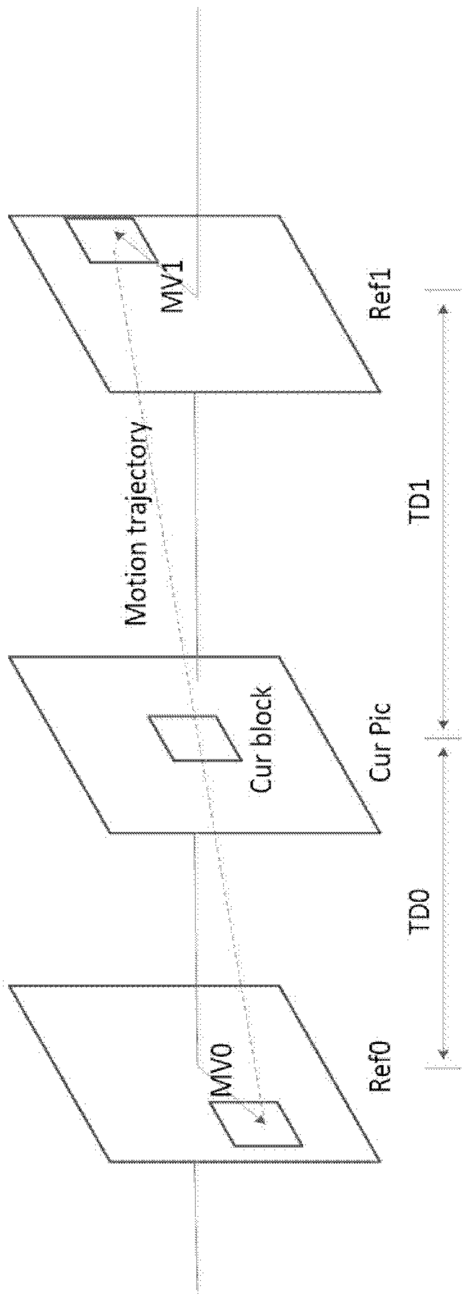

Motion trajectory
MV1
Ref1
TD1
Cur block
Cur Pic
TD0
MV0
Ref0

FIG. 21

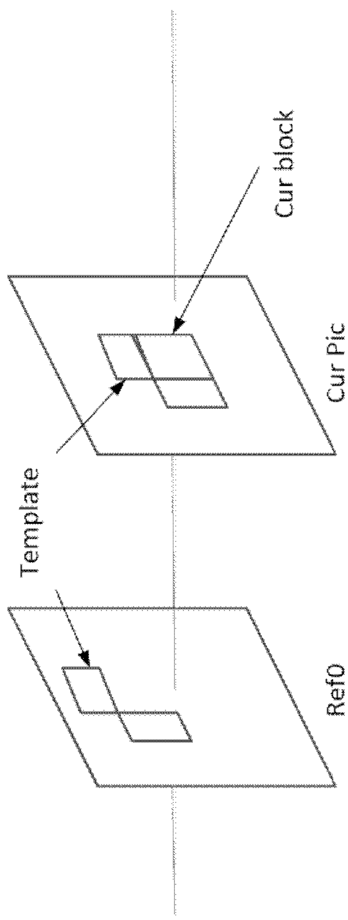


FIG. 22

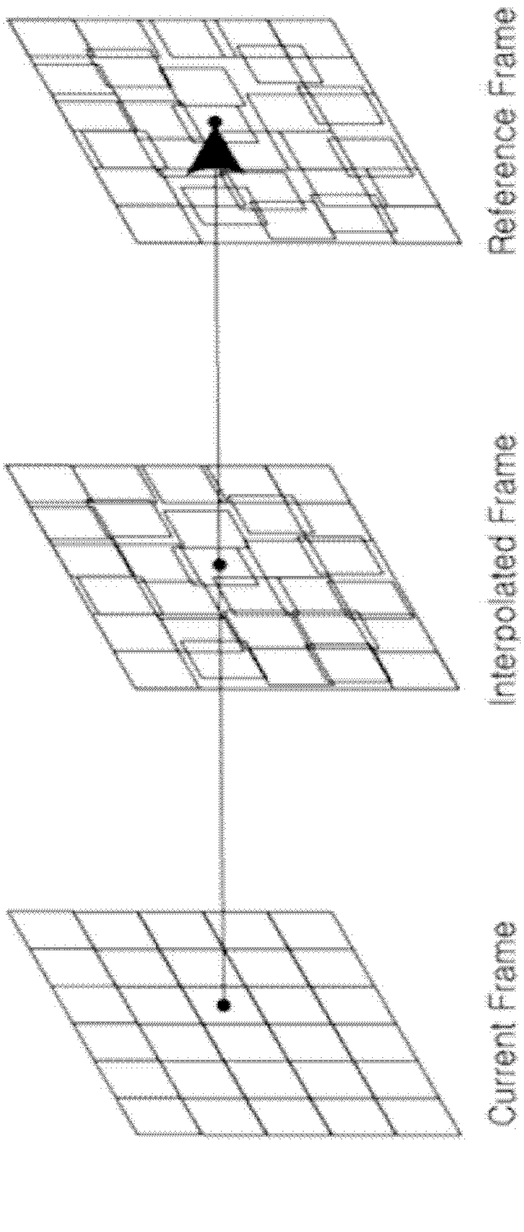


FIG. 23

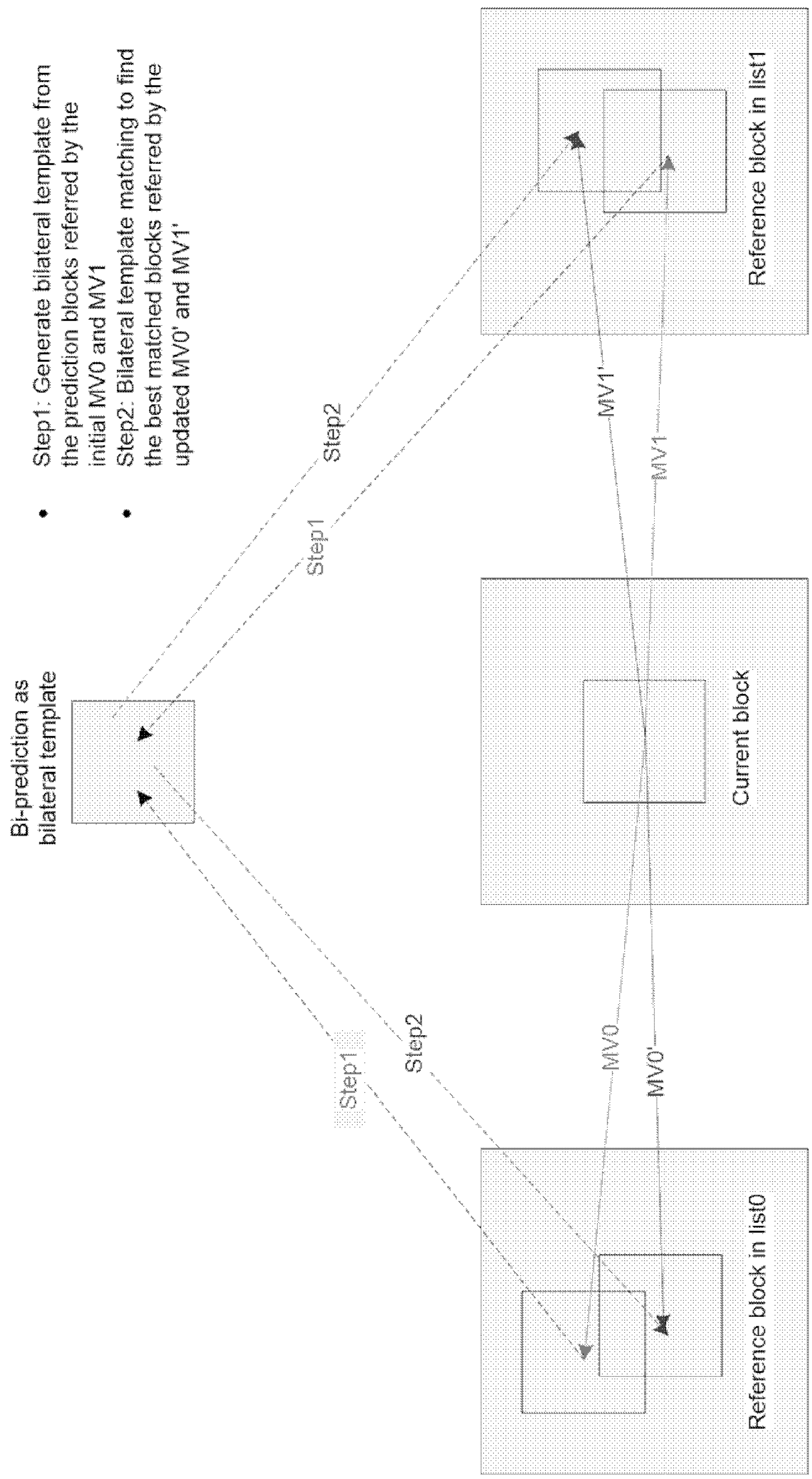

Bi-prediction as bilateral template
Step1: Generate bilateral template from the prediction blocks referred by the initial MV0 and MV1
Step2: Bilateral template matching to find the best matched blocks referred by the updated MV0' and MV1'
Step1
Step2
Step1
Step2
MV0
MV0'
MV1
MV1'
Reference block in list0
Current block
Reference block in list1

FIG. 24

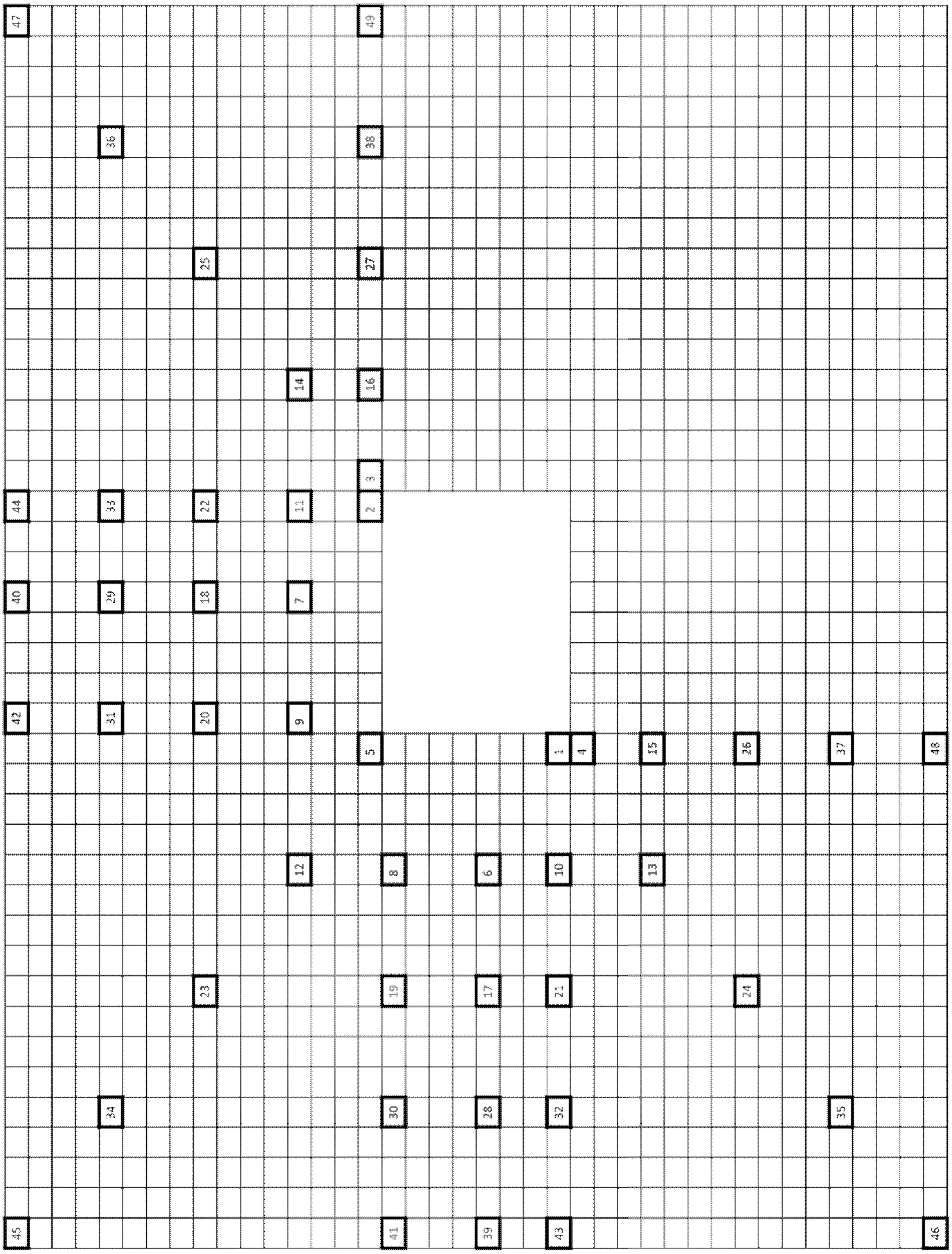


FIG. 25

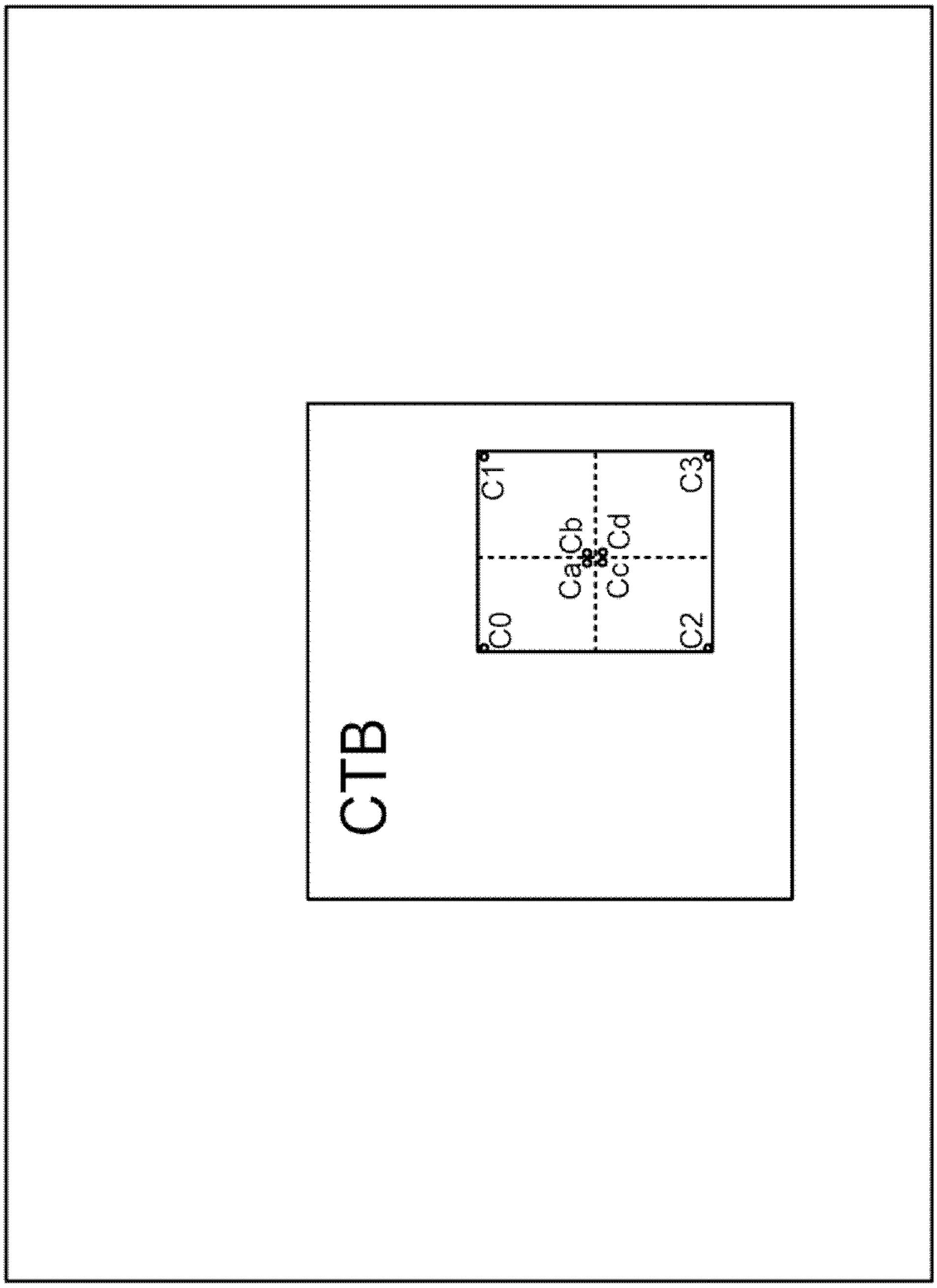


FIG. 26

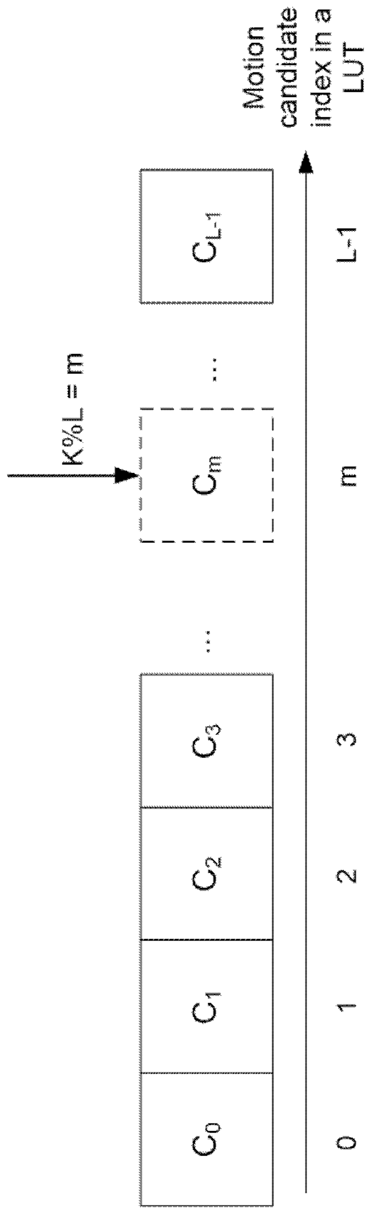


FIG. 27A

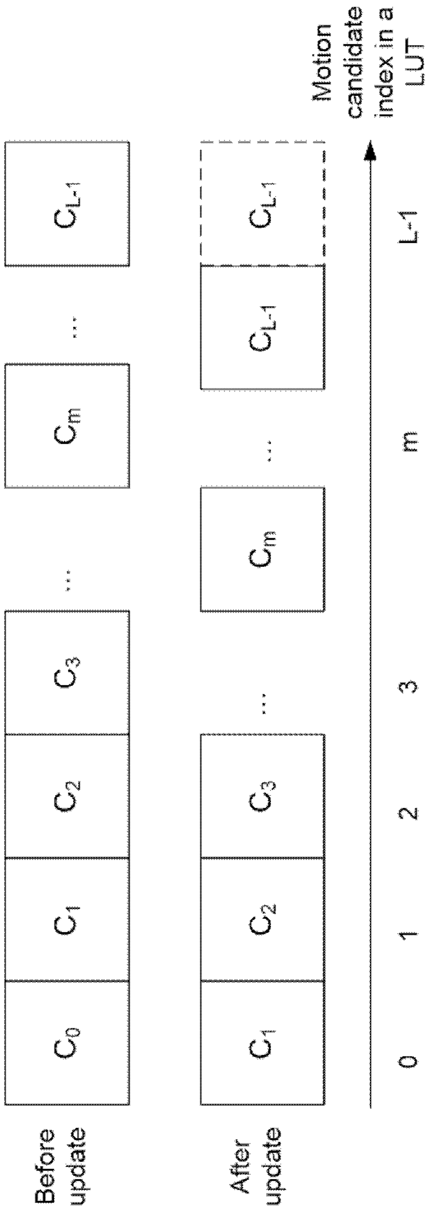

Before update
After update
C0
C1
C2
C3
...
Cm
...
CL-1
C1
C2
C3
...
Cm
...
CL-1
CL-1
0
1
2
3
m
L-1
Motion candidate index in a LUT

FIG. 27B

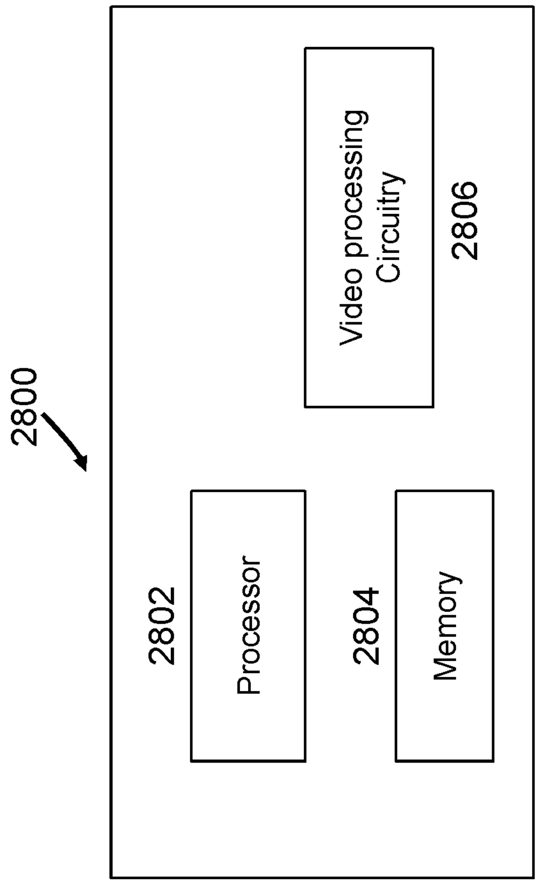

2800
2802
Processor
2804
Memory
Video processing
Circuitry
2806

FIG. 28A

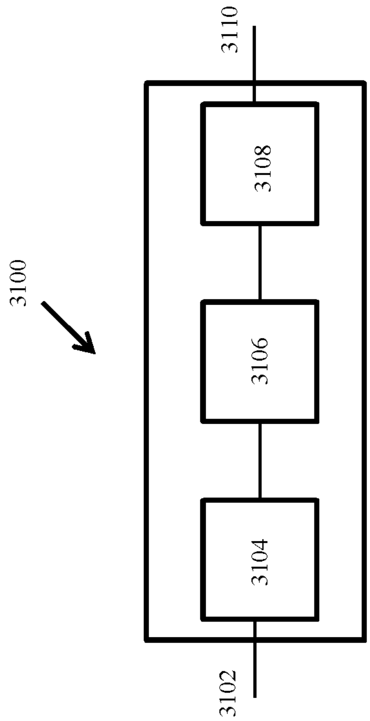


FIG. 28B

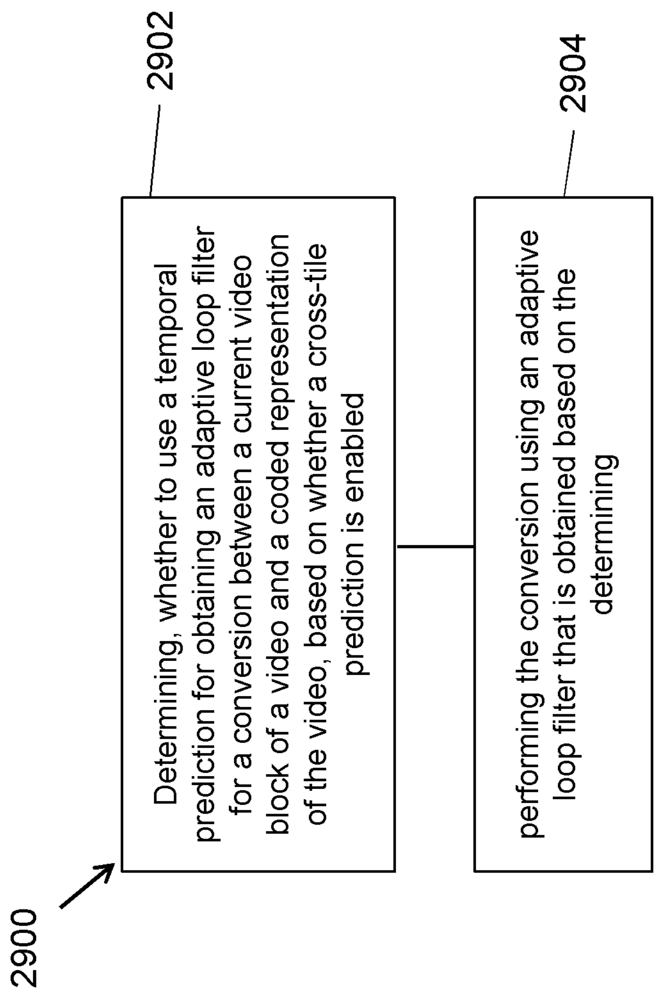

2900
2902
Determining, whether to use a temporal prediction for obtaining an adaptive loop filter for a conversion between a current video block of a video and a coded representation of the video, based on whether a cross-tile prediction is enabled
2904
performing the conversion using an adaptive loop filter that is obtained based on the determining

FIG. 29

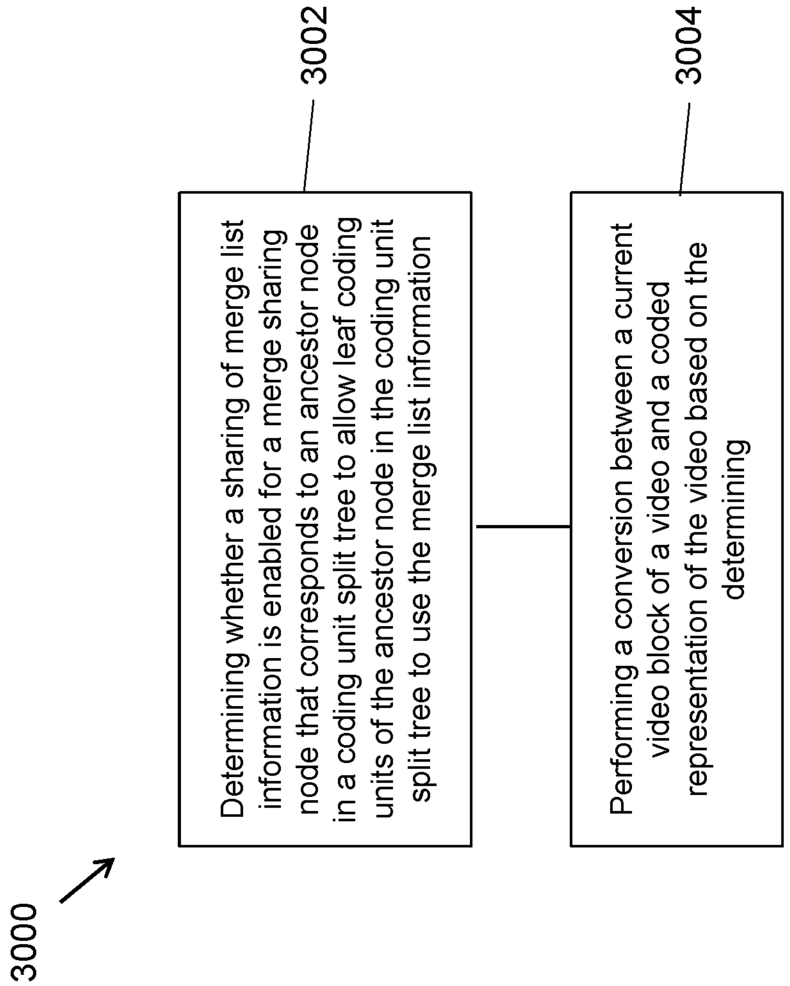

3000
Determining whether a sharing of merge list information is enabled for a merge sharing node that corresponds to an ancestor node in a coding unit split tree to allow leaf coding units of the ancestor node in the coding unit split tree to use the merge list information
3002
Performing a conversion between a current video block of a video and a coded representation of the video based on the determining
3004

FIG. 30

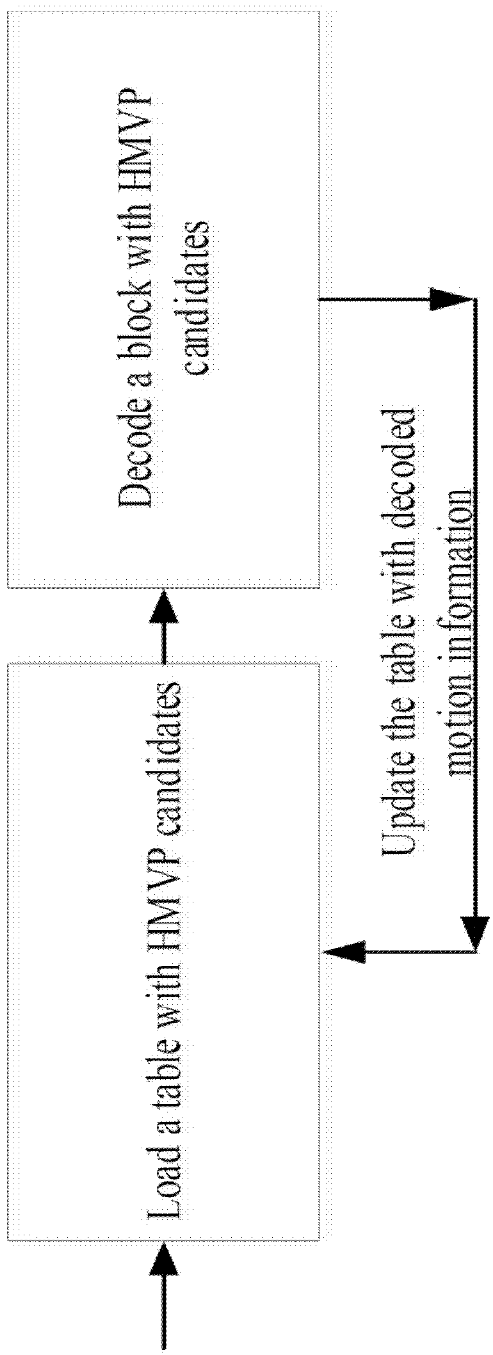

Load a table with HMVP candidates
Decode a block with HMVP candidates
Update the table with decoded motion information

FIG. 31

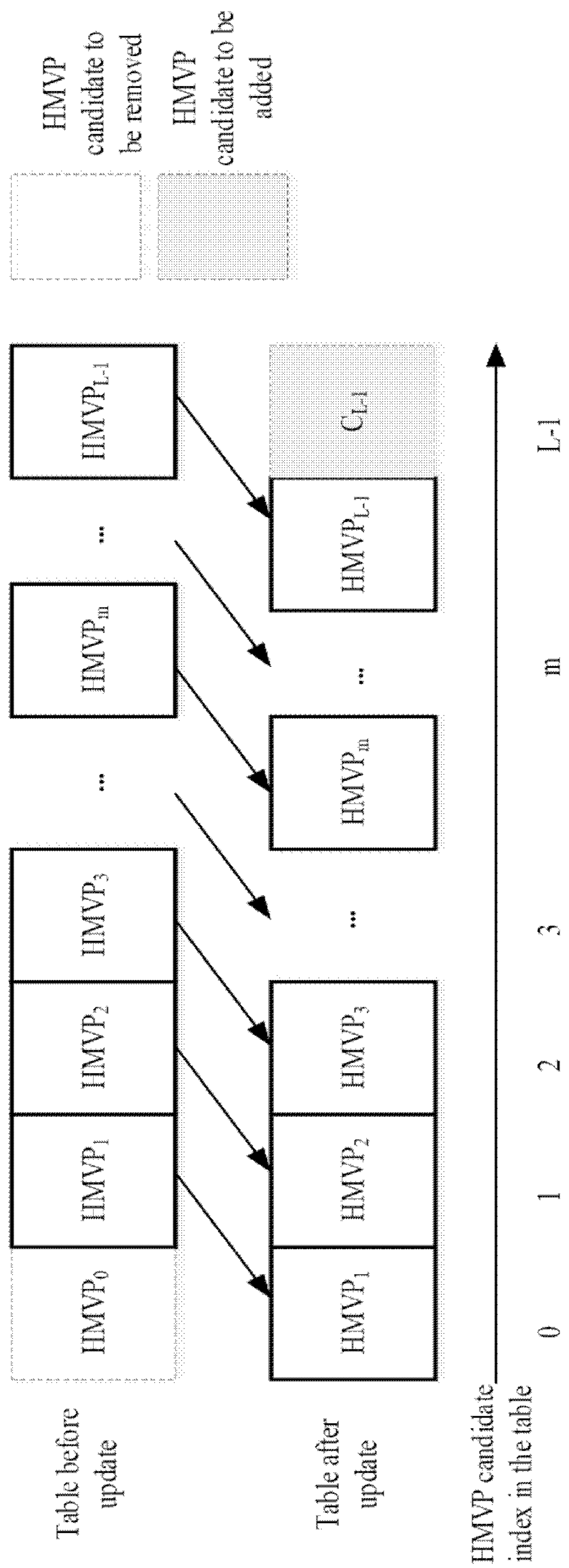

HMVP candidate to be removed
HMVP candidate to be added
Table before update
Table after update
HMVP candidate index in the table
HMVP0
HMVP1
HMVP2
HMVP3
...
HMVPm
...
HMVPL-1
HMVP1
HMVP2
HMVP3
...
HMVPm
...
HMVPL-1
CL-1
0
1
2
3
m
L-1

FIG. 32

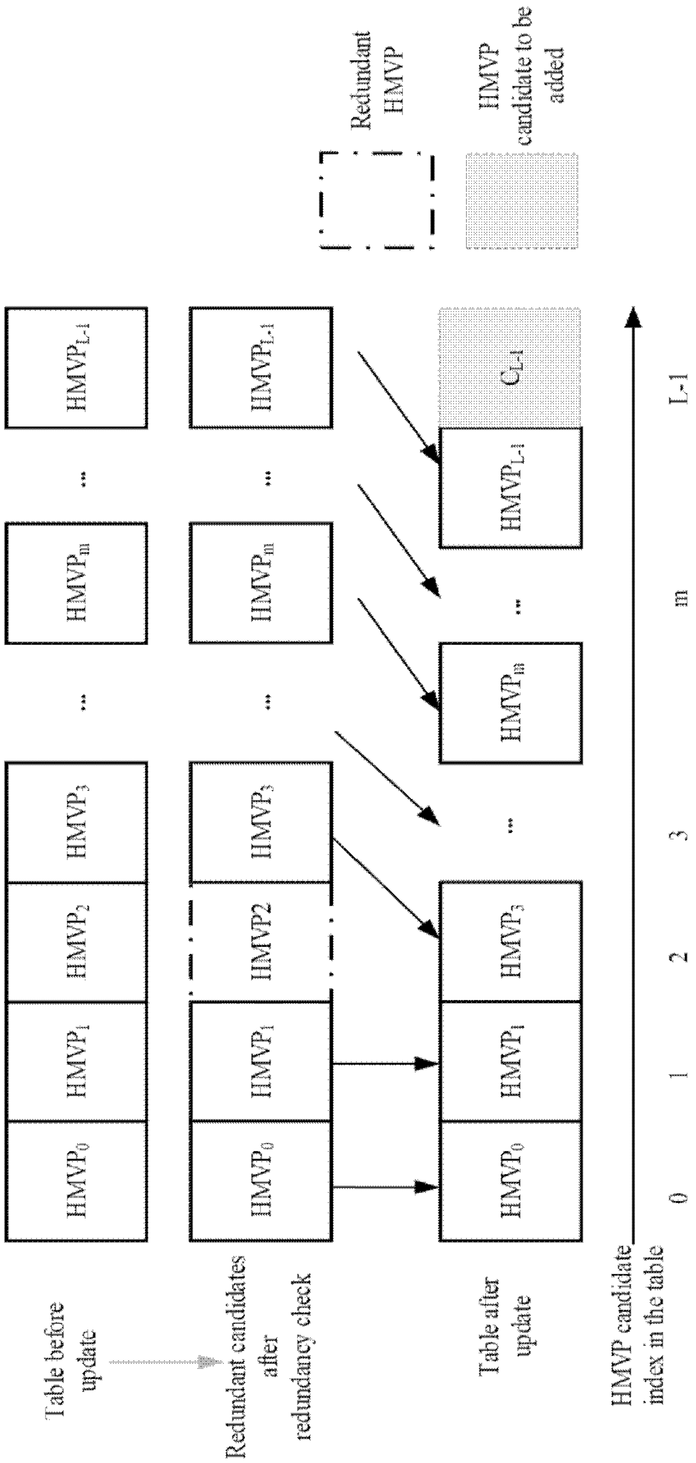


Table before update
Redundant candidates after redundancy check
Table after update
HMVP candidate index in the table
HMVP0
HMVP1
HMVP2
HMVP3
HMVPm
HMVPL-1
CL-1
0
1
2
3
m
L-1
Redundant HMVP
HMVP candidate to be added

FIG. 33A

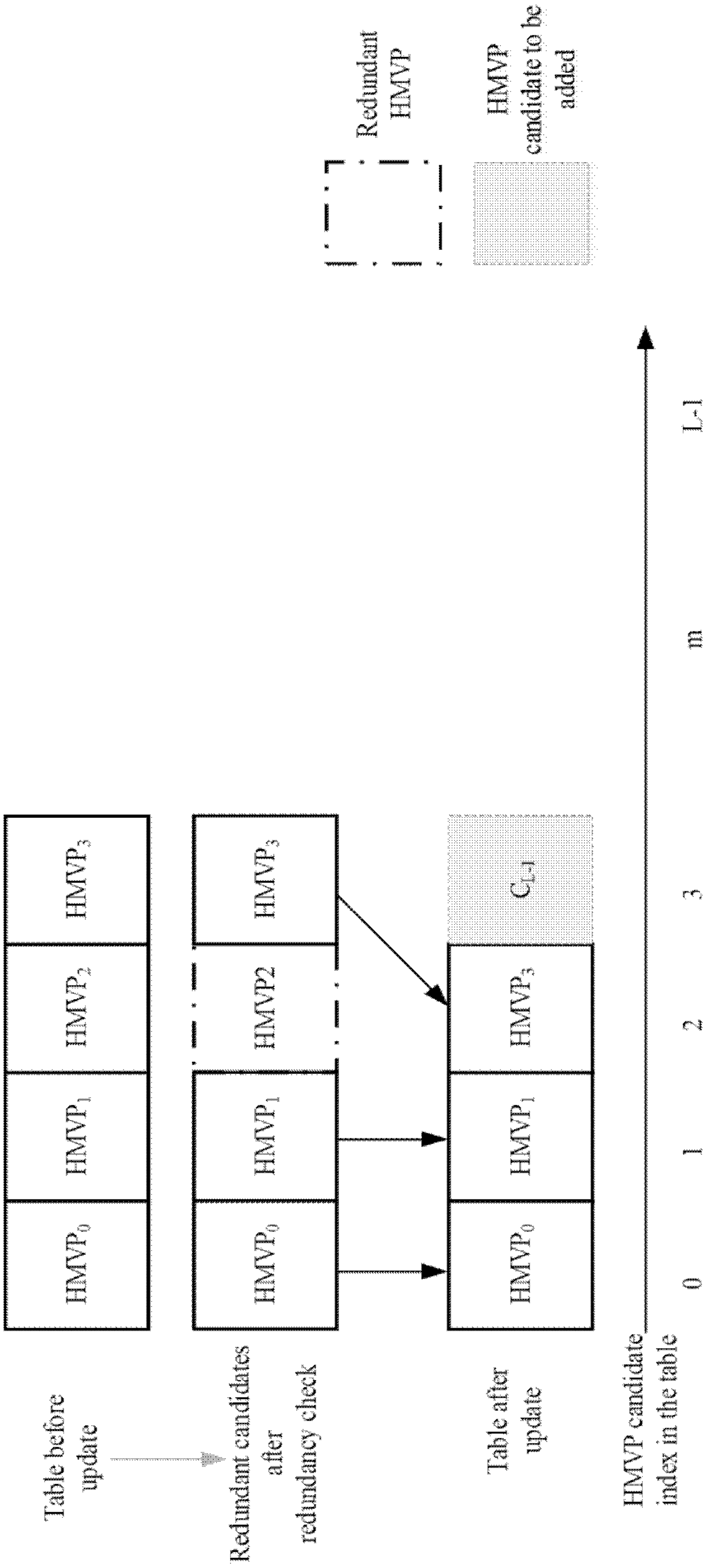


Table before update
Redundant candidates after redundancy check
Table after update
HMVP candidate index in the table
$HMVP_0$
$HMVP_1$
$HMVP_2$
$HMVP_3$
HMVP2
C_{L-1}
0
1
2
3
m
L-1
Redundant HMVP
HMVP candidate to be added

FIG. 33B

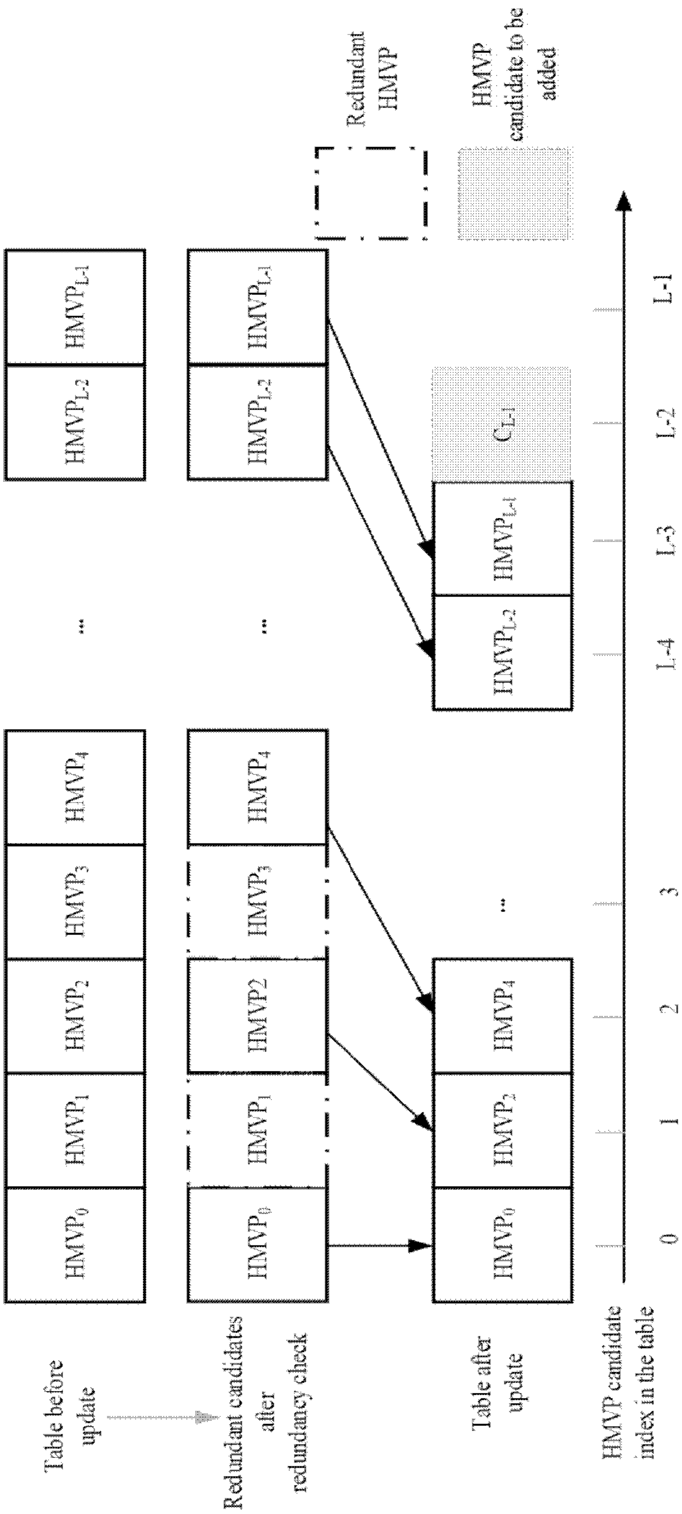


Table before update
Redundant candidates after redundancy check
Table after update
HMVP candidate index in the table
Redundant HMVP
HMVP candidate to be added
$HMVP_0$
$HMVP_1$
$HMVP_2$
$HMVP_3$
$HMVP_4$
$HMVP_{L-2}$
$HMVP_{L-1}$
HMVP2
C_{L-1}
0
1
2
3
L-4
L-3
L-2
L-1

FIG. 34A

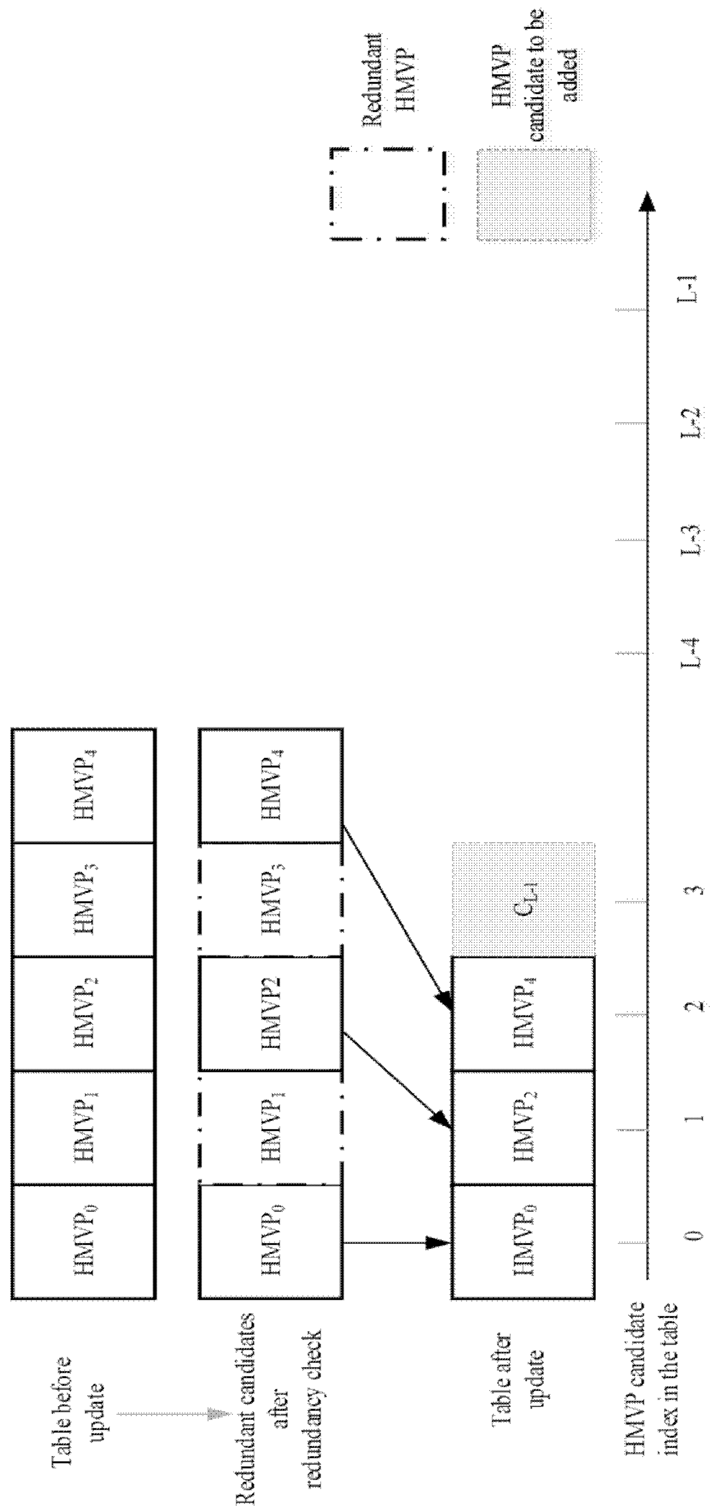


Table before update
Redundant candidates after redundancy check
Table after update
HMVP candidate index in the table
HMVP0
HMVP1
HMVP2
HMVP3
HMVP4
HMVP0
HMVP1
HMVP2
HMVP3
HMVP4
HMVP0
HMVP2
HMVP4
CL-1
0
1
2
3
L-4
L-3
L-2
L-1
Redundant HMVP
HMVP candidate to be added

FIG. 34B

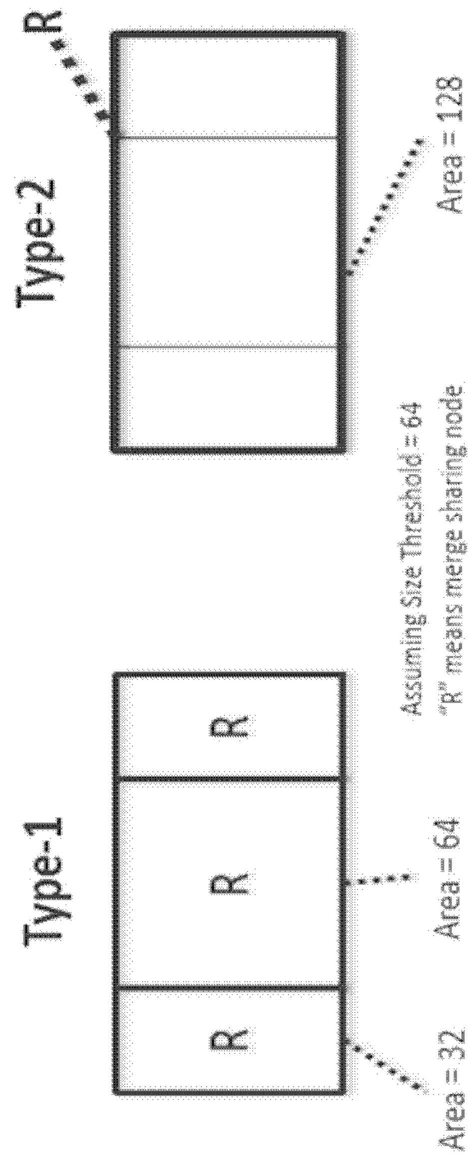

Type-2
R
Area = 128
Type-1
R
R
R
Area = 32
Area = 64
Assuming Size Threshold = 64
"R" means merge sharing node

FIG. 35

INTERNATIONAL SEARCH REPORT

International application No.

PCT/CN2020/071656

A. CLASSIFICATION OF SUBJECT MATTER

H04N 19/82(2014.01)i; H04N 19/117(2014.01)i; H04N 19/176(2014.01)i; H04N 19/157(2014.01)i

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

H04N

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)

CNABS;CNTXT;VEN;JVET: video, image, inter, temporal, predict+, loop, filter, conver+, merg+, candidate, shar+, motion, node, current, block, reference

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2017223352 A1 (SAMSUNG ELECTRONICS CO., LTD.) 03 August 2017 (2017-08-03) the description paragraphs 0103-0109	1-3
A	US 2017078699 A1 (SAMSUNG ELECTRONICS CO., LTD.) 16 March 2017 (2017-03-16) the whole document	1-25
A	CN 109076218 A (QUALCOMM INC.) 21 December 2018 (2018-12-21) the whole document	1-25
A	CN 104247434 A (QUALCOMM INC.) 24 December 2014 (2014-12-24) the whole document	1-25

☐ 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	Date of mailing of the international search report
19 March 2020	**03 April 2020**

Name and mailing address of the ISA/CN

National Intellectual Property Administration, PRC
6, Xitucheng Rd., Jimen Bridge, Haidian District, Beijing 100088
China

Facsimile No. **(86-10)62019451**

Authorized officer

MA,Hui

Telephone No. **86-(010)-62411524**

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2020/071656

Patent document cited in search report	Publication date (day/month/year)	Patent family member(s)	Publication date (day/month/year)
US 2017223352 A1	03 August 2017	CN 106716997 B	04 February 2020
		US 10250879 B2	02 April 2019
		CN 106716997 A	24 May 2017
		US 2019222840 A1	18 July 2019
		EP 3177014 A4	28 February 2018
		KR 20160015799 A	15 February 2016
		WO 2016017937 A1	04 February 2016
		EP 3177014 A1	07 June 2017
		SG 11201700676T A	27 February 2017
US 2017078699 A1	16 March 2017	EP 3107290 A4	20 September 2017
		MX 366439 B	09 July 2019
		US 10187657 B2	22 January 2019
		MX 2016011980 A	09 December 2016
		WO 2015137783 A1	17 September 2015
		KR 20160132859 A	21 November 2016
		EP 3107290 A1	21 December 2016
CN 109076218 A	21 December 2018	KR 20190008230 A	23 January 2019
		CA 3020233 A1	23 November 2017
		US 10419755 B2	17 September 2019
		US 2017332075 A1	16 November 2017
		BR 112018073380 A2	06 March 2019
		TW 201743619 A	16 December 2017
		JP 2019519982 A	11 July 2019
		WO 2017201011 A1	23 November 2017
		EP 3459245 A1	27 March 2019
CN 104247434 A	24 December 2014	US 2013101016 A1	25 April 2013
		ES 2677100 T3	30 July 2018
		EP 2769545 B1	18 April 2018
		CN 104247434 B	02 October 2018
		KR 20140081882 A	01 July 2014
		JP 5876156 B2	02 March 2016
		EP 2769545 A1	27 August 2014
		KR 101607788 B1	30 March 2016
		WO 2013058876 A1	25 April 2013
		US 9462298 B2	04 October 2016
		HU E038103 T2	28 September 2018
		JP 2014533008 A	08 December 2014