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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
23 January 2020 (23.01.2020)

WIPO | PCT

(10) International Publication Number
WO 2020/016745 A2

(51) International Patent Classification:
H04N 19/423 (2014.01) *H04N 19/52* (2014.01)
H04N 19/513 (2014.01)

(21) International Application Number:
PCT/IB2019/056020

(22) International Filing Date:
15 July 2019 (15.07.2019)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
PCT/CN2018/095716
14 July 2018 (14.07.2018) CN
PCT/CN2018/095719
15 July 2018 (15.07.2018) CN

(71) Applicants: **BEIJING BYTEDANCE NETWORK TECHNOLOGY CO., LTD.** [CN/CN]; Room B-0035, 2/F, No.3 Building, No.30, Shixing Road, Shijingshan District, Beijing 100041 (CN). **BYTEDANCE INC.** [US/US]; 12655 West Jefferson Boulevard, Sixth Floor, Suite No. 137, Los Angeles, California 90066 (US).

(72) Inventors: **ZHANG, Li**; 12655 West Jefferson Boulevard, Sixth Floor, Suite No. 137, Los Angeles, California 90066 (US). **ZHANG, Kai**; 12655 West Jefferson Boulevard, Sixth Floor, Suite No. 137, Los Angeles, California 90066 (US). **LIU, Hongbin**; Jinritoutiao Post Office, China Satellite Communications Tower, No.63, Zhichun Road, Haidian District, Beijing 100080 (CN). **WANG, Yue**; Jinritoutiao Post Office, China Satellite Communications Tower, No.63, Zhichun Road, Haidian District, Beijing 100080 (CN).

(74) Agent: **LIU, SHEN & ASSOCIATES**; 10th Floor, Building 1, 10 Caihefang Road, Haidian, District, Beijing 100080 (CN).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP,

(54) Title: EXTENSION OF LOOK-UP TABLE BASED MOTION VECTOR PREDICTION WITH TEMPORAL INFORMATION

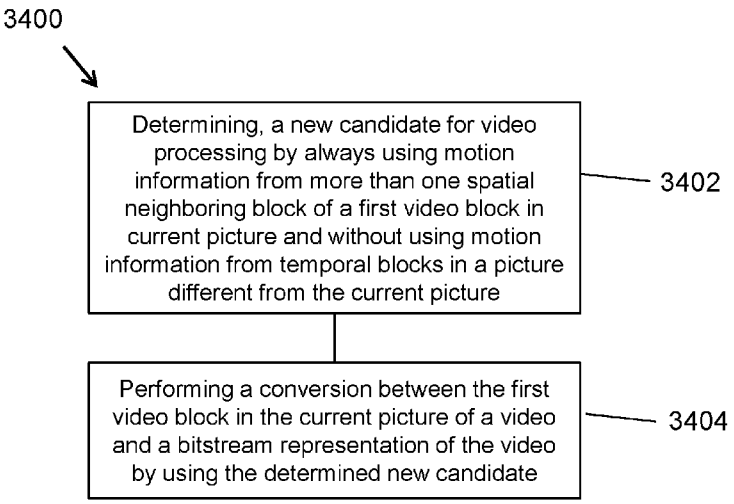


FIG. 34

(57) Abstract: The application provides a method for video processing. This method includes: determining, a new candidate for video processing by always using motion information from more than one spatial neighboring block of a first video block in current picture and without using motion information from temporal blocks in a picture different from the current picture; and performing a conversion between the first video block in the current picture of a video and a bitstream representation of the video by using the determined new candidate.

WO 2020/016745 A2

[Continued on next page]

KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

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

Published:

— *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

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

EXTENSION OF LOOK-UP TABLE BASED MOTION VECTOR PREDICTION WITH TEMPORAL INFORMATION

CROSS-REFERENCE TO RELATED APPLICATION

[0001] Under the applicable patent law and/or rules pursuant to the Paris Convention, this application is made to timely claim the priority to and benefits of International Patent Application No. PCT/CN2018/095716, filed on July 14, 2018, and International Patent Application No. PCT/CN2018/095719, filed on July 15, 2018. The entire disclosures of International Patent Application No. PCT/CN2018/095716 and No. PCT/CN2018/095719, are 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 by using new candidate for video processing.

[0005] In one example aspect, a method for video processing is disclosed. This method includes: determining a new candidate for video processing by averaging motion vectors of two or more selected motion candidates; adding the new candidate to a candidate list; and performing a conversion between a first video block of a video and a bitstream representation of the video by using the determined new candidate in the candidate list.

[0006] In one example aspect, a method of video processing is disclosed. The method of video processing, comprising: determining a new motion candidate for video processing by using

one or multiple motion candidate from one or multiple tables, wherein a table includes one or multiple of motion candidates and each motion candidate is associated motion information; and performing a conversion between a video block and a coded representation of the video block based on the new candidate.

[0007] In yet another representative aspect, a video processing method is disclosed. The video processing method comprises: maintaining a set of tables, wherein each table includes motion candidates and each motion candidate is associated with corresponding motion information; performing a conversion between a first video block and a bitstream representation of a video including the first video block; and updating one or multiple tables by selectively performing pruning with existing motion candidates in the one or multiple tables based on an encoding/decoding mode of the first video block.

[0008] In yet another representative aspect, a video processing method is disclosed. The video processing method comprises: maintaining a set of tables, wherein each table includes motion candidates and each motion candidate is associated with corresponding motion information; performing a conversion between a first video block and a bitstream representation of a video including the first video block; and updating one or multiple tables to include motion information from temporal neighbor block(s) of the first video block as a new motion candidate.

[0009] In yet another representative aspect, a method of updating a table of motion candidates is disclosed. The method comprises: selectively performing pruning with existing motion candidates in the table based on an encoding/decoding mode of a video block being processed, each motion candidate being associated with corresponding motion information; and updating the table to include motion information of the video block as a new motion candidate.

[0010] In yet another representative aspect, a method of updating a table of motion candidates is disclosed. The method comprises: maintaining the table of motion candidates, each motion candidate being associated with corresponding motion information; and updating the table to include motion information from temporal neighbor block(s) of a video block being processed as a new motion candidate.

[0011] In yet another representative aspect, a method for video processing is disclosed. The method for video processing, comprising: determining, a new candidate for video processing by always using motion information from more than one spatial neighboring block of a first video block in current picture and without using motion information from temporal blocks in a picture

different from the current picture; and performing a conversion between the first video block in the current picture of a video and a bitstream representation of the video by using the determined new candidate.

[0012] In yet another representative aspect, a method for video processing is disclosed. The method for video processing, comprising: determining, a new candidate for video processing by using motion information from at least one spatial non-adjacent block of a first video block in current picture and other candidates derived from spatial non-adjacent or not from spatial non-adjacent block of the first video block; and performing a conversion between the first video block of a video and a bitstream representation of the video by using the determined new candidate.

[0013] In yet another representative aspect, a method for video processing is disclosed. The method for video processing, comprising: determining, a new candidate for video processing by using motion information from one or more tables of a first video block in current picture and motion information from temporal blocks in a picture different from the current picture; and performing a conversion between the first video block in the current picture of a video and a bitstream representation of the video by using the determined new candidate.

[0014] In yet another representative aspect, a method for video processing is disclosed. The method for video processing, comprising: determining, a new candidate for video processing by using motion information from one or more tables of a first video block and motion information from one or more spatial neighboring block of the first video block; and performing a conversion between the first video block in the current picture of a video and a bitstream representation of the video by using the determined new candidate.

[0015] In yet another representative aspect, a method for video processing is disclosed. The method of video processing, comprising: determining a new motion candidate for video processing by using one or multiple motion candidate from one or multiple tables, wherein a table includes one or multiple of motion candidates and each motion candidate is associated motion information; and performing a conversion between a video block and a coded representation of the video block based on the new candidate.

[0016] In yet another representative aspect, an apparatus in a video system is disclosed. The apparatus comprises a processor and a non-transitory memory with instructions thereon, wherein the instructions upon execution by the processor, cause the processor to implement the various

method described herein.

[0017] 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.

[0018] 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

[0019] FIG. 1 is a block diagram showing an example of a video encoder implementation

[0020] FIG. 2 illustrates macroblock partitioning in the H.264 video coding standard.

[0021] FIG. 3 illustrates an example of splitting coding blocks (CB) into prediction blocks (PU).

[0022] 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.

[0023] FIG. 5 shows an example of a Quad Tree Binary Tree (QTBT) structure for partitioning video data.

[0024] FIG. 6 shows an example of video block partitioning.

[0025] FIG. 7 shows an example of quad-tree partitioning.

[0026] FIG. 8 shows an example of tree-type signaling.

[0027] FIG. 9 shows an example of a derivation process for merge candidate list construction.

[0028] FIG. 10 shows example positions of spatial merge candidates.

[0029] FIG. 11 shows examples of candidate pairs considered for redundancy check of spatial merge candidates.

[0030] FIG. 12 shows examples of positions for the second PU of Nx2N and 2NxN partitions.

[0031] FIG. 13 illustrates motion vector scaling for temporal merge candidates.

[0032] FIG. 14 shows candidate positions for temporal merge candidates, and their co-

located picture.

[0033] FIG. 15 shows an example of a combined bi-predictive merge candidate.

[0034] FIG. 16 shows an example of a derivation process for motion vector prediction candidates.

[0035] FIG. 17 shows an example of motion vector scaling for spatial motion vector candidates.

[0036] FIG. 18 shows an example Alternative Temporal Motion Vector Prediction (ATMVP) for motion prediction of a CU.

[0037] FIG. 19 pictorially depicts an example of identification of a source block and a source picture.

[0038] FIG. 20 shows an example of one CU with four sub-blocks and neighboring blocks.

[0039] FIG. 21 illustrates an example of bilateral matching.

[0040] FIG. 22 illustrates an example of template matching.

[0041] FIG. 23 depicts an example of unilateral Motion Estimation (ME) in Frame Rate UpConversion (FRUC).

[0042] FIG. 24 shows an example of DMVR based on bilateral template matching.

[0043] FIG. 25 shows an example of spatially neighboring blocks used to derive intensity compensation IC parameters.

[0044] FIG. 26 shows an example of spatially neighboring blocks used to derive spatial merge candidates.

[0045] FIG. 27 illustrates examples of using neighboring inter-prediction blocks.

[0046] FIG. 28 shows an example of a planar motion vector prediction process.

[0047] FIG. 29 shows examples of locations next to the current coding unit (CU) line.

[0048] FIG. 30 is a block diagram illustrating an example of the architecture for a computer system or other control device that can be utilized to implement various portions of the presently disclosed technology.

[0049] FIG. 31 shows a block diagram of an example embodiment of a mobile device that can be utilized to implement various portions of the presently disclosed technology.

[0050] FIG. 32 shows a flowchart of an example method for video processing in accordance with the presently disclosed technology.

[0051] FIG. 33 shows a flowchart of an example method for video processing in accordance

with the presently disclosed technology.

[0052] FIG. 34 shows a flowchart of an example method for video processing in accordance with the presently disclosed technology.

[0053] FIG. 35 shows a flowchart of an example method for video processing in accordance with the presently disclosed technology.

[0054] FIG. 36 shows a flowchart of an example method for video processing in accordance with the presently disclosed technology.

[0055] FIG. 37 shows a flowchart of an example method for video processing in accordance with the presently disclosed technology.

[0056] FIG. 38 shows a flowchart of an example method for video processing in accordance with the presently disclosed technology.

[0057] FIG. 39 shows a flowchart of an example method for video processing in accordance with the presently disclosed technology.

[0058] FIG. 40 shows a flowchart of an example method for updating a table of motion candidates in accordance with the presently disclosed technology.

[0059] FIG. 41 shows a flowchart of an example method for updating a table of motion candidates in accordance with the presently disclosed technology.

[0060] FIG. 42 shows a flowchart of an example method for video processing in accordance with the presently disclosed technology.

DETAILED DESCRIPTION

[0061] To improve compression ratio of video, researchers are continually looking for new techniques by which to encode video.

[0062] **1. Introduction**

[0063] 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.

[0064] **Brief discussion**

[0065] 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.

[0066] **2.1 Partition Structure**

[0067] **2.1.1 Partition tree structure in H.264/AVC**

[0068] 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.

[0069] An intra-coded block uses spatial prediction to exploit spatial correlation among pixels. Two partitions are defined: 16x16 and 4x4.

[0070] 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.

[0071] **2.1.2 Partition tree structure in HEVC**

[0072] 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.

[0073] In the following, the various features involved in hybrid video coding using HEVC are highlighted as follows.

[0074] 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.

[0075] 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).

[0076] 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.

[0077] 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.

[0078] 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.

[0079] **2.1.2.1 Tree-Structured Partitioning into Transform Blocks and Units**

[0080] 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.

[0081] 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.

[0082] **2.1.2.2 Parent and child nodes**

[0083] 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.

[0084] **2.1.3 Quadtree plus binary tree block structure with larger CTUs in JEM**

[0085] 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).

[0086] **2.1.3.1 QTBT block partitioning structure**

[0087] 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.

[0088] 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

[0089] 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.

[0090] 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.

[0091] 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.

[0092] 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.

[0093] **2.1.4 Ternary-tree for VVC**

[0094] As proposed in JVET-D0117, 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).

[0095] 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.

[0096] In JVET-D0117, 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'.

[0097] **2.1.5 Partitioning structure in JVET-J0021**

[0098] 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.

[0099] 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.

[00100] 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.

[00101] 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.

[00102] **2.2 Inter prediction in HEVC/H.265**

[00103] 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.

[00104] 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.

[00105] 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.

[00106] 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.

[00107] The following text provides the details on the inter prediction modes specified in HEVC. The description will start with the merge mode.

[00108] 2.2.1 Merge mode

[00109] 2.2.1.1 Derivation of candidates for merge mode

[00110] 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

[00111] 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.

[00112] In the following, the operations associated with the aforementioned steps are detailed.

[00113] **2.2.1.2 Spatial candidates derivation**

[00114] 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.

[00115] **2.2.1.3 Temporal candidate derivation**

[00116] 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.

[00117] 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.

[00118] **2.2.1.4 Additional candidates insertion**

[00119] 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, defined in HEVC.

[00120] 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.

[00121] **2.2.1.5 Motion estimation regions for parallel processing**

[00122] 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.

Picture parameter set RBSP syntax

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.

[00123] 2.2.2 Motion vector prediction in AMVP mode

[00124] 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.

[00125] 2.2.2.1 Derivation of motion vector prediction candidates

[00126] FIG. 16 summarizes derivation process for motion vector prediction candidate.

[00127] 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.

[00128] 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.

[00129] 2.2.2.2 Spatial motion vector candidates

[00130] 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)

[00131] 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.

[00132] 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.

[00133] **2.2.2.3 Temporal motion vector candidates**

[00134] 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.

[00135] **2.2.2.4 Signaling of AMVP information**

[00136] For the AMVP mode, four parts may be signalled in the bitstream, i.e., prediction direction, reference index, MVD and mv predictor candidate index.

Syntax tables:

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)
}	
}	
}	
}	

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)
}	
}	

[00137] **2.3 New inter prediction methods in JEM (Joint Exploration Model)**

[00138] **2.3.1 Sub-CU based motion vector prediction**

[00139] 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.

[00140] To preserve more accurate motion field for sub-CU motion prediction, the motion compression for the reference frames is currently disabled.

[00141] **2.3.1.1 Alternative temporal motion vector prediction**

[00142] 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).

[00143] 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.

[00144] 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.

[00145] FIG. 19 shows an example of the identification of source block and source picture

[00146] 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.

[00147] 2.3.1.2 Spatial-temporal motion vector prediction

[00148] 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.

[00149] 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.

[00150] FIG. 20 shows an example of one CU with four sub-blocks (A-D) and its neighbouring blocks (a–d).

[00151] 2.3.1.3 Sub-CU motion prediction mode signalling

[00152] 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.

[00153] 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.

[00154] 2.3.2 Adaptive motion vector difference resolution

[00155] 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.

[00156] 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.

[00157] 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.

[00158] 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.

[00159] 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.

[00160] 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.

[00161] **2.3.3 Pattern matched motion vector derivation**

[00162] 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.

[00163] 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.

[00164] 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.

[00165] 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.

[00166] 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.

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

[00167] 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.

[00168] 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.

[00169] **2.3.3.1 CU level MV candidate set**

[00170] 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

[00171] 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.

[00172] 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.

[00173] When FRUC is applied in AMVP mode, the original AMVP candidates are also added to CU level MV candidate set.

[00174] 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.

[00175] **2.3.3.2 Sub-CU level MV candidate set**

[00176] 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

[00177] 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.

[00178] ATMVP and STMVP candidates are limited to the four first ones.

[00179] At the sub-CU level, up to 17 MVs are added to the candidate list.

[00180] **2.3.3.3 Generation of interpolated MV field**

[00181] 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.

[00182] 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.

[00183] **2.3.3.4 Interpolation and matching cost**

[00184] 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.

[00185] 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:

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

[00186] 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.

[00187] 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.

[00188] 2.3.3.5 MV refinement

[00189] 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.

[00190] 2.3.3.6 Selection of prediction direction in template matching FRUC merge mode

[00191] 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;

[00192] 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.

[00193] **2.3.4 Decoder-side motion vector refinement**

[00194] 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.

[00195] 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.

[00196] 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.

[00197] **2.3.5 Local illumination compensation**

[00198] Local Illumination Compensation (IC) is based on a linear model for illumination changes, using a scaling factor *a* and an offset *b*. And it is enabled or disabled adaptively for each inter-mode coded coding unit (CU).

[00199] When IC applies for a CU, a least square error method is employed to derive the parameters a and b by using the neighbouring samples of the current CU and their corresponding reference samples. More specifically, as illustrated in FIG. 25, the subsampled (2:1 subsampling) neighbouring samples of the CU and the corresponding samples (identified by motion information of the current CU or sub-CU) in the reference picture are used. The IC parameters are derived and applied for each prediction direction separately.

[00200] When a CU is coded with merge mode, the IC flag is copied from neighbouring blocks, in a way similar to motion information copy in merge mode; otherwise, an IC flag is signalled for the CU to indicate whether LIC applies or not.

[00201] When IC is enabled for a pciture, addtional CU level RD check is needed to determine whether LIC is applied or not for a CU. When IC is enabled for a CU, mean-removed sum of absolute diffefference (MR-SAD) and mean-removed sum of absolute Hadamard-transformed difference (MR-SATD) are used, instead of SAD and SATD, for integer pel motion search and fractional pel motion search, respectively.

[00202] To reduce the encoding complexity, the following encoding scheme is applied in the JEM.

[00203] IC is disabled for the entire picture when there is no obvious illumination change between a current picture and its reference pictures. To identify this situation, histograms of a current picture and every reference picture of the current picture are calculated at the encoder. If the histogram difference between the current picture and every reference picture of the current picture is smaller than a given threshold, IC is disabled for the current picture; otherwise, IC is enabled for the current picture.

[00204] **2.3.6 Merge/Skip mode with Bilateral matching refinement**

[00205] 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. 26, 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:

1. Spatial candidates for blocks 1-4.
2. Extrapolated affine candidates for blocks 1-4.
3. ATMVP.
4. STMVP.
5. Virtual affine candidate.
6. Spatial candidate (block 5) (used only when the number of the available candidates is smaller than 6).
7. Extrapolated affine candidate (block 5).
8. Temporal candidate (derived as in HEVC).
9. **Non-adjacent spatial candidate followed by extrapolated affine candidate (blocks 6 to 49).**
10. Combined candidates.
11. Zero candidates.

[00206] 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.

[00207] **2.3.7 JVET-K0161**

[00208] In this proposal, no subblock STMVP is proposed as a spatial-temporal merge mode. This proposed method uses a collocated block, which is the same as HEVC / JEM (only 1 picture, no temporal vector here). The proposed method also checks upper and left spatial position, which position is adjusted in this proposal. Specifically to check neighbouring inter-prediction information, at most two positions is checked for each above and left. The exact position is shown in FIG. 27.

Afar: (nPbW * 5 / 2, -1), Amid (nPbW / 2, -1) (note: offsets of above spatial blocks above the current block)

Lfar: (-1, nPbH * 5 / 2), Lmid (-1, nPbH/2) (note: offsets of left spatial blocks above the current block)

[00209] An average of motion vectors of above block, left block and a temporal block is

calculated as the same as BMS software implementation. If the 3 reference inter-prediction block is available.

mvLX[0] = ((mvLX_A[0] + mvLX_L[0] + mvLX_C[0]) * 43) / 128

mvLX[1] = ((mvLX_A[1] + mvLX_L[1] + mvLX_C[1]) * 43) / 128

[00210] If only two or one inter-prediction block is available, average of two or just use one mv is used.

[00211] **2.3.8 JVT-K0135**

[00212] To generate a smooth fine granularity motion field, FIG. 28 gives a brief description of the planar motion vector prediction process.

[00213] Planar motion vector prediction is achieved by averaging a horizontal and vertical linear interpolation on 4x4 block basis as follows.

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

[00214] W and H denote the width and the height of the block. *(x,y)* is the coordinates of current sub-block relative to the above left corner sub-block. All the distances are denoted by the pixel distances divided by 4. $P(x,y)$ is the motion vector of current sub-block.

The horizontal prediction $P_h(x,y)$ and the vertical prediction $P_v(x,y)$ for location *(x,y)* are calculated as follows:

$$P_h(x,y) = (W-1-x) \times L(-1,y) + (x+1) \times R(W,y)$$

$$P_v(x,y) = (H-1-y) \times A(x,-1) + (y+1) \times B(x,H)$$

where $L(-1,y)$ and $R(W,y)$ are the motion vectors of the 4x4 blocks to the left and right of the current block. $A(x,-1)$ and $B(x,H)$ are the motion vectors of the 4x4 blocks to the above and bottom of the current block.

[00215] The reference motion information of the left column and above row neighbour blocks are derived from the spatial neighbour blocks of current block.

[00216] The reference motion information of the right column and bottom row neighbour blocks are derived as follows.

[00217] Derive the motion information of the bottom right temporal neighbour *4x4* block

[00218] Compute the motion vectors of the right column neighbour *4x4* blocks, using the derived motion information of the bottom right neighbour *4x4* block along with the motion

information of the above right neighbour *4x4* block, as described in Equation K1.

[00219] Compute the motion vectors of the bottom row neighbour 4x4 blocks, using the derived motion information of the bottom right neighbour 4x4 block along with the motion information of the bottom left neighbour 4x4 block, as described in Equation K2.

$$R(W,y) = ((H-y-1) \times AR + (y+1) \times BR)/H \quad \textbf{Equation K1}$$

$$B(x,H) = ((W-x-1) \times BL + (x+1) \times BR)/W \quad \textbf{Equation K2}$$

[00220] where *AR* is the motion vector of the above right spatial neighbour *4x4* block, BR is the motion vector of the bottom right temporal neighbour *4x4* block, and BL is the motion vector of the bottom left spatial neighbour *4x4* block.

[00221] The motion information obtained from the neighbouring blocks for each list is scaled to the first reference picture for a given list.

[00222] **3. Examples of Problems Addressed by Embodiments disclosed herein**

[00223] Inventors have previously proposed look-up table based motion vector prediction techniques using one or more look up tables with at least one motion candidate stored to predict motion information of a block can be implemented in various embodiments to provide video coding with higher coding efficiencies. Each LUT can include one or more motion candidates, each associated with corresponding motion information. Motion information of a motion candidate can include partial or all of the prediction direction, reference indices/pictures, motion vectors, LIC flags, affine flags, Motion Vector Difference (MVD) precisions, and/or MVD values. Motion information may further include the block position information to indicate wherein the motion information is coming from.

[00224] The LUT-based motion vector prediction based on the disclosed technology, which may enhance both existing and future video coding standards, is elucidated in the following examples described for various implementations. Because the LUTs allow the encoding/decoding process to be performed based on historical data (e.g., the blocks that have been processed), the LUT-based motion vector prediction can also be referred to as History-based Motion Vector Prediction (HMVP) method. In the LUT-based motion vector prediction method, one or multiple tables with motion information from previously coded blocks are maintained during the encoding/decoding process. These motion candidates stored in the LUTs

are named HMVP candidates. During the encoding/decoding of one block, the associated motion information in LUTs may be added to the motion candidate lists (e.g., merge/AMVP candidate lists), and after encoding/decoding one block, LUTs may be updated. The updated LUTs are then used to code the subsequent blocks. That is, the updating of motion candidates in the LUTs are based on the encoding/decoding order of blocks. The examples below should be considered as examples to explain general concepts. These examples should not be interpreted in a narrow way. Furthermore, these examples can be combined in any manner.

[00225] 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.

[00226] Although current LUT-based motion vector prediction techniques overcome the drawback of HEVC by using historical data, however, only information from spatial neighbouring blocks are considered.

[00227] When a motion candidate from the LUT is used for either AMVP or merge list construction process, it is directly inherited without any changes.

[00228] The design of JVET-K0161 is beneficial for coding performance. However, it requires additional derivation of TMVP which increases the computation complexity and memory bandwidth.

[00229] **4. Some Examples**

[00230] 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.

[00231] Some embodiments that use the presently disclosed technology may jointly use motion candidates from LUTs and motion information from temporal neighboring blocks. In addition, complexity reduction of the JVET-K0161 is also proposed.

[00232] **Utilization of motion candidates from LUTs**

1. It is proposed to construct a new AMVP/merge candidate by utilizing a motion candidate from LUTs.

a. In one example, a new candidate may be derived by adding/subtracting offset(s) of motion vectors of a motion candidate from LUTs.
b. In one example, a new candidate may be derived by averaging of motion vectors of selected motion candidates from LUTs.
    i. In one embodiment, the averaging can be approximately implemented without the division operation. For example, MVa, MVb and MVc can be averaged as (MVa+MVb+MVc)*$\lfloor 2^N/3 \rfloor/2^N$ or (MVa+MVb+MVc)*$\lceil 2^N/3 \rceil/2^N$. For example when N=7, the average is (MVa+MVb+MVc)*42/128 or (MVa+MVb+MVc)*43/128. Notice that $\lfloor 2^N/3 \rfloor$ or $\lceil 2^N/3 \rceil$ are pre-calculated and stored in a look-up table.
    ii. In one example, only motion vectors with identical reference pictures (in both prediction directions) are selected.
    iii. In one example, the reference picture in each prediction direction is pre-determined, and motion vector are scaled to the pre-determined reference picture if necessary.
        1. In one example, the first entry in the reference picture list X (X = 0 or 1) is selected as the reference picture.
        2. Alternatively, for each prediction direction, the most frequently used reference picture in the LUT is selected as the reference picture.
c. In one example, for each prediction direction, motion vectors with identical reference picture to the pre-determined reference picture is firstly selected, and then other motion vectors are selected.

2. It is proposed to construct a new AMVP/merge candidate by a function of one or multiple motion candidates from LUTs and motion information from temporal neighboring blocks.
    a. In one example, similar to STMVP or JVET-K0161, the new candidate may be derived by an average of motion candidates from LUTs and TMVP.
    b. In one example, the above blocks (e.g., Amid and Afar in) may be replaced by candidates from LUTs. Alternatively, furthermore, the other process may be kept unchanged like what has been implemented in JVET-K0161.

3. It is proposed to construct a new AMVP/merge candidate by a function of one or multiple motion candidates from LUTs, AMVP/merge candidates from spatial neighboring blocks and/or spatial non-adjacent neighboring blocks and motion information from temporal blocks.
    a. In one example, one or more of above blocks (e.g., Amid and Afar in) may be replaced by candidates from LUTs. Alternatively, furthermore, the other process may be kept unchanged like what has been implemented in JVET-K0161.
    b. In one example, one or more of left blocks (e.g., Amid and Afar in) may be replaced by candidates from LUTs. Alternatively, furthermore, the other process may be kept unchanged like what has been implemented in JVET-K0161.
4. It is proposed that when inserting motion information of a block into the LUT, whether pruning with existing entries in LUT or not may depend on the encoding mode of the block.
    a. In one example, if the block is coded in merge mode, the pruning is not performed.
    b. In one example, if the block is coded in AMVP mode, the pruning is not performed.
    c. In one example, if the block is coded in AMVP/merge mode, it is only pruned with the latest M entries of the LUT.
    d. In one example, when the block is coded with sub-block mode (e.g., affine or ATMVP), the pruning is always disabled.
5. It is proposed to add motion information from temporal blocks to LUTs.
    a. In one example, motion information could be from a co-located block.
    b. In one example, motion information could be from one or multiple blocks from different reference pictures.

[00233] Related to STMVP

1. It is proposed to always use the spatial merge candidates to derive a new merge candidate without taking the TMVP candidate into consideration.
    a. In one example, the average of two motion merge candidates may be utilized.

b. In one example, spatial merge candidates and motion candidates from LTUs may be jointly used to derive a new candidate.

2. It is proposed that non-adjacent blocks (which is not the right or left neighboring block) may be utilized to derive a STMVP candidate.
    a. In one example, the above blocks used for STMVP candidate derivation are kept unchanged, while the used left blocks are changed from neighboring blocks to non-adjacent blocks.
    b. In one example, the left blocks used for STMVP candidate derivation are kept unchanged, while the used above blocks are changed from neighboring blocks to non-adjacent blocks.
    c. In one example, candidates of non-adjacent blocks and motion candidates from LTUs may be jointly used to derive a new candidate.

3. It is proposed to always use the spatial merge candidates to derive a new merge candidate without taking the TMVP candidate into consideration.
    a. In one example, the average of two motion merge candidates may be utilized.
    b. Alternatively, the average of two, three or more MVs from different locations, which may be neighbouring to the current block or not.
        i. In one embodiment, the MVs can only be fetched from locations which are in the current LCU (a.k.a CTU).
        ii. In one embodiment, the MVs can only be fetched from locations which are in the current LCU line.
        iii. In one embodiment, the MVs can only be fetched from locations which are in the current LCU line or next to the current LCU line. An example is shown in Fig. 29. Block A, B, C, E, E and F are next to the current LCU line.
        iv. In one embodiment, the MVs can only be fetched from locations which are in the current LCU line or next to the current LCU line but not at left of the top-left neighbouring block. An example is shown in Fig. 29. Block

T is the top-left neighbouring block. Block B, C, E, E and F are next to the current LCU line but not at left of the top-left neighbouring block.

c. In one example, spatial merge candidates and motion candidates from LTUs may be jointly used to derive a new candidate.

4. [11]It is proposed that the MV for the BR block in Fig. 28 for the planar motion prediction is not fetched from temporal MV prediction but from one entry of the LUT.

5. It is proposed that motion candidates from LUTs could be jointly used with other kinds of merge/AMVP candidates (e.g., spatial merge/AMVP candidates, temporal merge/AMVP candidates, default motion candidates) to derive new candidates.

[00234] In various implementations of this example and other examples disclosed in this patent document, the pruning may include a) comparing the motion information with existing entries for uniqueness, and b) if unique, then adding the motion information to the list, or c) if not unique, then either c1) not adding or c2) adding the motion information and deleting existing entry that matched. In some implementations, the pruning operation is not invoked when adding a motion candidate from a table to a candidate list.

[00235] FIG. 30 is a block diagram illustrating an example of the architecture for a computer system or other control device 3000 that can be utilized to implement various portions of the presently disclosed technology. In FIG. 30, the computer system 3000 includes one or more processors 3005 and memory 3010 connected via an interconnect 3025. The interconnect 3025 may represent any one or more separate physical buses, point to point connections, or both, connected by appropriate bridges, adapters, or controllers. The interconnect 3025, therefore, may include, for example, a system bus, a Peripheral Component Interconnect (PCI) bus, a HyperTransport or industry standard architecture (ISA) bus, a small computer system interface (SCSI) bus, a universal serial bus (USB), IIC (I2C) bus, or an Institute of Electrical and Electronics Engineers (IEEE) standard 674 bus, sometimes referred to as "Firewire."

[00236] The processor(s) 3005 may include central processing units (CPUs) to control the overall operation of, for example, the host computer. In certain embodiments, the processor(s) 3005 accomplish this by executing software or firmware stored in memory 3010. The processor(s) 3005 may be, or may include, one or more programmable general-purpose or special-purpose

microprocessors, digital signal processors (DSPs), programmable controllers, application specific integrated circuits (ASICs), programmable logic devices (PLDs), or the like, or a combination of such devices.

[00237] The memory 3010 can be or include the main memory of the computer system. The memory 3010 represents any suitable form of random access memory (RAM), read-only memory (ROM), flash memory, or the like, or a combination of such devices. In use, the memory 3010 may contain, among other things, a set of machine instructions which, when executed by processor 3005, causes the processor 3005 to perform operations to implement embodiments of the presently disclosed technology.

[00238] Also connected to the processor(s) 3005 through the interconnect 3025 is a (optional) network adapter 3015. The network adapter 3015 provides the computer system 3000 with the ability to communicate with remote devices, such as the storage clients, and/or other storage servers, and may be, for example, an Ethernet adapter or Fiber Channel adapter.

[00239] FIG. 31 shows a block diagram of an example embodiment of a mobile device 3100 that can be utilized to implement various portions of the presently disclosed technology. The mobile device 3100 can be a laptop, a smartphone, a tablet, a camcorder, or other types of devices that are capable of processing videos. The mobile device 3100 includes a processor or controller 3101 to process data, and memory 3102 in communication with the processor 3101 to store and/or buffer data. For example, the processor 3101 can include a central processing unit (CPU) or a microcontroller unit (MCU). In some implementations, the processor 3101 can include a field-programmable gate-array (FPGA). In some implementations, the mobile device 3100 includes or is in communication with a graphics processing unit (GPU), video processing unit (VPU) and/or wireless communications unit for various visual and/or communications data processing functions of the smartphone device. For example, the memory 3102 can include and store processor-executable code, which when executed by the processor 3101, configures the mobile device 3100 to perform various operations, e.g., such as receiving information, commands, and/or data, processing information and data, and transmitting or providing processed information/data to another device, such as an actuator or external display.

[00240] To support various functions of the mobile device 3100, the memory 3102 can store information and data, such as instructions, software, values, images, and other data processed or referenced by the processor 3101. For example, various types of Random Access Memory (RAM)

devices, Read Only Memory (ROM) devices, Flash Memory devices, and other suitable storage media can be used to implement storage functions of the memory 3102. In some implementations, the mobile device 3100 includes an input/output (I/O) unit 3103 to interface the processor 3101 and/or memory 3102 to other modules, units or devices. For example, the I/O unit 3103 can interface the processor 3101 and memory 3102 with to utilize various types of wireless interfaces compatible with typical data communication standards, e.g., such as between the one or more computers in the cloud and the user device. In some implementations, the mobile device 3100 can interface with other devices using a wired connection via the I/O unit 3103. The mobile device 3100 can also interface with other external interfaces, such as data storage, and/or visual or audio display devices 3104, to retrieve and transfer data and information that can be processed by the processor, stored in the memory, or exhibited on an output unit of a display device 3104 or an external device. For example, the display device 3104 can display a video frame that includes a block (a CU, PU or TU) that applies the intra-block copy based on whether the block is encoded using a motion compensation algorithm, and in accordance with the disclosed technology.

[00241] In some embodiments, a video decoder apparatus may implement a method of sub-block based prediction as described herein is used for video decoding.

[00242] In some embodiments, the video decoding methods may be implemented using a decoding apparatus that is implemented on a hardware platform as described with respect to FIG. 30 and FIG. 31.

[00243] Various embodiments and techniques disclosed in the present document can be described in the following listing of examples.

[00244] FIG. 32 is a flowchart of an example method 3200 for video processing in accordance with the presently disclosed technology. The method 3200 includes, at operation 3202, determining a new candidate for video processing by averaging motion vectors of two or more selected motion candidates. The method 3200 includes, at operation 3204, adding the new candidate to a candidate list. The method 3200 includes, at operation 3206, performing a conversion between a first video block of a video and a bitstream representation of the video by using the determined new candidate in the candidate list.

[00245] In some embodiments, the candidate list is a merge candidate list and the determined new candidate is a merge candidate.

[00246] In some embodiments, the merge candidate list is an inter prediction merge candidate

list or an intra block copy prediction merge candidate list.

[00247] In some embodiments, the one or more tables include motion candidates derived from previously processed video blocks that are processed prior to the first video block in video data.

[00248] In some embodiments, there is no available spatial and temporal candidates in the candidate list.

[00249] In some embodiments, the selected motion candidates are from one or more tables.

[00250] In some embodiments, the averaging is implemented without a division operation.

[00251] In some embodiments, the averaging is implemented by a multiplication operation of a sum of the motion vectors of the selected motion candidates and a scaling factor.

[00252] In some embodiments, the horizontal components of motion vectors of the selected motion candidates are averaged to derive the horizontal component of a new candidate.

[00253] In some embodiments, the vertical components of motion vectors of the selected motion candidates are averaged to derive the vertical component of a new candidate.

[00254] In some embodiments, the scaling factor is pre-calculated and stored in a look-up table.

[00255] In some embodiments, only motion vectors with identical reference pictures are selected.

[00256] In some embodiments, only motion vectors with identical reference pictures in both prediction directions are selected in both prediction directions.

[00257] In some embodiments, a target reference picture in each prediction direction is pre-determined, and the motion vectors are scaled to the pre-determined reference picture.

[00258] In some embodiments, a first entry in a reference picture list X is selected as the target reference picture for the reference picture list, X is 0 or 1.

[00259] In some embodiments, for each prediction direction, the most frequently used reference picture in the table is selected as the target reference picture.

[00260] In some embodiments, for each prediction direction, the motion vectors with identical reference picture to the pre-determined target reference picture are firstly selected, and then other motion vectors are selected.

[00261] In some embodiments, the motion candidate from a table is associated with motion information which includes at least one of: a prediction direction, a reference picture index, motion vector values, intensity compensation flag, affine flag, motion vector difference precision, or motion vector difference value.

[00262] In some embodiments, the method 3200 further comprises updating, based on the conversion, one or more tables.

[00263] In some embodiments, the updating of one or more tables includes updating one or more tables based on the motion information of the first video block of the video after performing the conversion.

[00264] In some embodiments, the method 3200 further comprises performing a conversion between a subsequent video block of the video and the bitstream representation of the video based on the updated tables.

[00265] In some embodiments, the conversion includes an encoding process and/or decoding process.

[00266] In some embodiments, a video encoding apparatus may perform the method 2900 and other methods described herein during reconstruction of video for subsequent video.

[00267] In some embodiments, an apparatus in a video system may include a processor configured to perform the methods described herein.

[00268] In some embodiments, the described methods may be embodied as computer-executable code that is stored on a computer-readable program medium.

[00269] FIG. 33 is a flowchart of an example method 3300 for video processing in accordance with the presently disclosed technology. The method 3300 includes, at operation 3302, determining a new motion candidate for video processing by using one or multiple motion candidate from one or multiple tables, wherein a table includes one or multiple of motion candidates and each motion candidate is associated motion information. The method 3300 includes, at operation 3304, performing a conversion between a video block and a coded representation of the video block based on the new candidate.

[00270] In some embodiments, the new motion candidate is derived by adding or subtracting offsets of motion vectors associated with a motion candidate from the one or multiple tables.

[00271] In some embodiments, the determining the new motion candidate includes: determining the new motion candidate as a function of one or multiple of the motion candidates and motion information from temporal neighboring blocks.

[00272] In some embodiments, the determining the new motion candidate includes: averaging motion candidates from the one or multiple look-up tables and temporal motion vector predictors.

[00273] In some embodiments, the averaging selected motion candidates comprises weighted

average or average of motion vectors associated with selected motion candidates.

[00274] In some embodiments, the averaging is implemented without a division operation.

[00275] In some embodiments, the averaging is implemented by a multiplication operation of a sum of the motion vectors of motion candidates from the one or multiple tables and temporal motion vector predictors and a scaling factor.

[00276] In some embodiments, the horizontal components of the motion vectors of motion candidates from the one or multiple tables and temporal motion vector predictors are averaged to derive the horizontal component of a new motion candidate.

[00277] In some embodiments, the averaging selected horizontal components comprises weighted average or average of horizontal components associated with selected motion candidates.

[00278] In some embodiments, the vertical components of the motion vectors of motion candidates from the one or multiple tables and temporal motion vector predictors are averaged to derive the vertical component of a new motion candidate.

[00279] In some embodiments, the averaging selected vertical components comprises weighted average or average of vertical components associated with selected motion candidates.

[00280] In some embodiments, the one or multiple motion candidate from one or multiple tables are derived from previously processed video blocks that are processed prior to the video block in video data.

[00281] In some embodiments, the determining the new candidate includes: determining the new motion candidate as a function of one or more motion candidates from the one or multiple tables, merge candidates from spatial neighboring blocks and/or spatial non-adjacent neighboring blocks, and motion information from temporal neighboring blocks.

[00282] In some embodiments, the determining the new candidate includes: determining the new motion candidate as a function of one or more motion candidates from the one or multiple tables, advanced motion vector prediction (AMVP) candidate from spatial neighboring blocks and/or spatial non-adjacent neighboring blocks, and motion information from temporal neighboring blocks.

[00283] In some embodiments, the determining the new candidate includes: determining the new motion candidate as a function of one or more motion candidates from the one or multiple tables, and advanced motion vector prediction (AMVP) candidate in the AMVP candidate list or merge candidates in the merge candidate list.

[00284] In some embodiments, the new motion candidate is added to a merge candidate list.

[00285] In some embodiments, the new motion candidate is added to a AMVP candidate list.

[00286] In some embodiments, each of one or multiple table includes a set of motion candidates, wherein each motion candidate is associated with corresponding motion information.

[00287] In some embodiments, the motion candidate is associated with motion information which includes at least one of: a prediction direction, a reference picture index, motion vector values, intensity compensation flag, affine flag, motion vector difference precision, or motion vector difference value.

[00288] In some embodiments, the method further comprises updating, based on the conversion, one or more tables.

[00289] In some embodiments, the updating of one or more tables includes updating one or more tables based on the motion information of a first video block after performing the conversion.

[00290] In some embodiments, the method further comprises performing a conversion between a subsequent video block of the video and the bitstream representation of the video based on the updated tables.

[00291] FIG. 34 is a flowchart of an example method 3400 for video processing in accordance with the presently disclosed technology. The method 3400 includes, at operation 3402, determining, a new candidate for video processing by always using motion information from more than one spatial neighboring block of a first video block in current picture and without using motion information from temporal blocks in a picture different from the current picture. The method 3400 includes, at operation 3404, performing a conversion between the first video block in the current picture of a video and a bitstream representation of the video by using the determined new candidate.

[00292] In some embodiments, the determined new candidate is added to a candidate list, comprising a merge or Advanced Motion Vector Prediction (AMVP) candidate list.

[00293] In some embodiments, the motion information from more than one spatial neighboring blocks are candidates derived from pre-defined spatial neighboring blocks relative to the first video block in the current picture or motion candidates from one or more tables.

[00294] In some embodiments, the candidates derived from pre-defined spatial neighboring blocks relative to the first video block in the current picture are spatial merge candidates.

[00295] In some embodiments, the new candidate is derived by averaging at least two spatial

merge candidates.

[00296] In some embodiments, the new candidate is derived by jointly using the spatial merge candidates and motion candidates from one or more tables.

[00297] In some embodiments, the new candidate is derived by averaging at least two motion vectors associated with candidates derived from different locations.

[00298] In some embodiments, the different locations are neighbouring to the first video block.

[00299] In some embodiments, the motion vectors only are fetched from locations which are in a current largest coding unit to which the first video block belongs.

[00300] In some embodiments, the motion vectors only are fetched from locations which are in a current largest coding unit line.

[00301] In some embodiments, the motion vectors only are fetched from locations which are in a current largest coding unit line or next to the current largest coding unit line.

[00302] In some embodiments, the motion vectors only are fetched from locations which are in the current largest coding unit line or next to the current largest coding unit line but not at left of the top-left neighbouring block.

[00303] In some embodiments, the motion vector for the bottom-right block for the planar motion prediction is not fetched from temporal motion vector prediction candidates but from one entry of the table.

[00304] In some embodiments, the new candidate is derived by jointly using motion candidates from one or more tables and other kinds of merge/AMVP candidates.

[00305] In some embodiments, the motion candidate in the one or more tables is associated with motion information which includes at least one of: a prediction direction, a reference picture index, motion vector values, intensity compensation flag, affine flag, motion vector difference precision, or motion vector difference value.

[00306] In some embodiments, the method 3400 further comprises updating, based on the conversion, one or more tables.

[00307] In some embodiments, the updating of one or more tables includes updating one or more tables based on the motion information of the first video block after performing the conversion.

[00308] In some embodiments, the method 3400 further comprises: performing a conversion between a subsequent video block of the video and the bitstream representation of the video based

on the updated tables.

[00309] In some embodiments, the conversion includes an encoding process and/or decoding process.

[00310] FIG. 35 is a flowchart of an example method 3500 for video processing in accordance with the presently disclosed technology. The method 3500 includes, at operation 3502, determining, a new candidate for video processing by using motion information from at least one spatial non-adjacent block of a first video block in current picture and other candidates derived from spatial non-adjacent or not from spatial non-adjacent block of the first video block. The method 3500 includes, at operation 3504, performing a conversion between the first video block of a video and a bitstream representation of the video by using the determined new candidate.

[00311] In some embodiments, the determined new candidate is added to a candidate list, comprising a merge or Advanced Motion Vector Prediction (AMVP) candidate list.

[00312] In some embodiments, the motion information from more than one spatial non-adjacent blocks are candidates derived from pre-defined spatial non-adjacent blocks relative to the first video block in the current picture.

[00313] In some embodiments, the candidates derived from pre-defined spatial non-adjacent blocks relative to the first video block in the current picture are spatial-temporal motion vector prediction (STMVP) candidate.

[00314] In some embodiments, the non-adjacent blocks of the video block are not right or left neighboring block of the first video block.

[00315] In some embodiments, above blocks of the first video block used for STMVP candidate derivation are kept unchanged, while the used left blocks are changed from neighboring blocks to non-adjacent blocks.

[00316] In some embodiments, left blocks of the first video block used for STMVP candidate derivation are kept unchanged, while the used above blocks are changed from neighboring blocks to non-adjacent blocks.

[00317] FIG. 36 is a flowchart of an example method 3600 for video processing in accordance with the presently disclosed technology. The method 3600 includes, at operation 3602, determining, a new candidate for video processing by using motion information from one or more tables of a first video block in current picture and motion information from temporal blocks in a picture different from the current picture. The method 3600 includes, at operation 3604,

performing a conversion between the first video block in the current picture of a video and a bitstream representation of the video by using the determined new candidate.

[00318] In some embodiments, the determined new candidate is added to a candidate list, comprising a merge or AMVP candidate list.

[00319] In some embodiments, the motion information from one or more tables in current picture are associated with one or more History Motion Vector Prediction (HMVP) candidates selected from one or more tables, and the motion information from temporal blocks in a picture different from the current picture are temporal motion candidates.

[00320] In some embodiments, the new candidate is derived by averaging one or more HMVP candidates and one or more temporal motion candidates.

[00321] In some embodiments, the one or more tables include motion candidates derived from previously processed video blocks that are processed prior to the first video block in video data.

[00322] FIG. 37 is a flowchart of an example method 3700 for video processing in accordance with the presently disclosed technology. The method 3700 includes, at operation 3702, determining, a new candidate for video processing by using motion information from one or more tables of a first video block and motion information from one or more spatial neighboring block of the first video block. The method 3700 includes, at operation 3704, performing a conversion between the first video block in the current picture of a video and a bitstream representation of the video by using the determined new candidate.

[00323] In some embodiments, the determined new candidate is added to a candidate list, comprising a merge or AMVP candidate list.

[00324] In some embodiments, the motion information from one or more tables of a first video block are associated with one or more History Motion Vector Prediction (HMVP) candidates selected from one or more tables, and the motion information from one or more spatial neighboring block of the first video block are candidates derived from pre-defined spatial blocks relative to the first video block.

[00325] In some embodiments, the candidates derived from pre-defined spatial blocks relative to the first video block are spatial merge candidates.

[00326] In some embodiments, the new candidate is derived by averaging one or more HMVP candidates and one or more spatial merge candidates.

[00327] In some embodiments, the one or more tables include motion candidates derived from

previously processed video blocks that are processed prior to the first video block in video data.

[00328] In some embodiments, a motion candidate from a table is associated with motion information including at least one of: a prediction direction, a reference picture index, motion vector values, an intensity compensation flag, an affine flag, a motion vector difference precision, or motion vector difference value.

[00329] In some embodiments, the method further comprises updating, based on the conversion, one or more tables.

[00330] In some embodiments, the updating of one or more tables includes updating one or more tables based on the motion information of the current video block after performing the conversion.

[00331] In some embodiments, the method further comprises: performing a conversion between a subsequent video block of the video data and the bitstream representation of the video data based on the updated tables.

[00332] In some embodiments, an apparatus in a video system comprising a processor configured to implement the methods described herein.

[00333] In some embodiments, a computer-readable program medium having code stored thereupon, the code comprising instructions that, when executed by a processor, causing the processor to implement the methods described herein.

[00334] FIG. 38 is a flowchart of an example method 3800 for video processing in accordance with the presently disclosed technology. The method 3800 includes, at operation 3802, maintaining a set of tables, wherein each table includes motion candidates and each motion candidate is associated with corresponding motion information; at operation 3804, performing a conversion between a first video block and a bitstream representation of a video including the first video block; and at operation 3806 updating one or multiple tables by selectively performing pruning with existing motion candidates in the one or multiple tables based on an encoding/decoding mode of the first video block.

[00335] In some embodiments, the conversion between the first video block and the bitstream representation of a video including the first video block is performed based on one or multiple tables in the set of tables.

[00336] In some embodiments, pruning is omitted in case that the first video block is coded in merge mode.

[00337] In some embodiments, pruning is omitted in case that the first video block is coded in advanced motion vector prediction mode

[00338] In some embodiments, pruning is performed with the latest M entries of the table in case that the first video block is coded in merge mode or advanced motion vector prediction mode, where M is a pre-specified integer.

[00339] In some embodiments, pruning is disabled in case that the first video block is coded in sub-block mode.

[00340] In some embodiments, the sub-block mode includes affine mode, alternative temporal motion vector prediction mode.

[00341] In some embodiments, the pruning includes checking whether there is a redundant existing motion candidate in the table.

[00342] In some embodiments, the pruning further includes inserting motion information associated with the first video block into the table and deleting the redundant existing motion candidate in the table if there is one.

[00343] In some embodiments, if there is one redundant existing motion candidate in the table, motion information associated with the first video block is not used to update the table.

[00344] In some embodiments, the method further comprises performing a conversion between a subsequent video block of the video and the bitstream representation of the video based on the updated tables.

[00345] FIG. 39 is a flowchart of an example method 3900 for video processing in accordance with the presently disclosed technology. The method 3900 includes, at operation 3902, maintaining a set of tables, wherein each table includes motion candidates and each motion candidate is associated with corresponding motion information; at operation 3904, performing a conversion between a first video block and a bitstream representation of a video including the first video block; and at operation 3906, updating one or multiple tables to include motion information from temporal neighbor block(s) of the first video block as a new motion candidate.

[00346] In some embodiments, the conversion between the first video block and the bitstream representation of a video including the first video block is performed based on one or multiple tables in the set of tables.

[00347] In some embodiments, the temporal neighbor block(s) is a co-located block.

[00348] In some embodiments, the temporal neighbors block(s) include one or multiple blocks

from different reference pictures.

[00349] In some embodiments, the method further comprises performing a conversion between a subsequent video block of the video and the bitstream representation of the video based on the updated tables.

[00350] FIG. 40 is a flowchart of an example method 4000 for updating a table of motion candidates in accordance with the presently disclosed technology. The method 4000 includes, at operation 4002, selectively performing pruning with existing motion candidates in the table based on an encoding/decoding mode of a video block being processed, each motion candidate being associated with corresponding motion information; at operation 4004, updating the table to include motion information of the video block as a new motion candidate.

[00351] In some embodiments, pruning is performed with the latest M entries of the table in case that the video block is coded in merge mode or advanced motion vector prediction mode, where M is a pre-specified integer.

[00352] In some embodiments, pruning is disabled in case that the video block is coded in sub-block mode.

[00353] In some embodiments, the sub-block mode includes affine mode, alternative temporal motion vector prediction mode.

[00354] In some embodiments, the pruning includes checking whether there is a redundant motion candidate in the table.

[00355] In some embodiments, the pruning further includes inserting motion information associated with the video block being processed into the table and deleting the redundant motion candidate in the table if there is one.

[00356] In some embodiments, if there is one redundant existing motion candidate in the table, motion information associated with the first video block is not used to update the table.

[00357] FIG. 41 is a flowchart of an example method 4100 for updating a table of motion candidates in accordance with the presently disclosed technology. The method 4100 includes, at operation 4102, maintaining the table of motion candidates, each motion candidate being associated with corresponding motion information; and at operation 4104, updating the table to include motion information from temporal neighbor block(s) of a video block being processed as a new motion candidate.

[00358] In some embodiments, the temporal neighbor block(s) is a co-located block.

[00359] In some embodiments, the temporal neighbor block(s) include one or multiple blocks from different reference pictures.

[00360] In some embodiments, a motion candidate is associated with motion information including at least one of: a prediction direction, a reference picture index, motion vector values, an intensity compensation flag, an affine flag, a motion vector difference precision, or motion vector difference value.

[00361] In some embodiments, the motion candidates correspond to motion candidates for intra prediction modes for intra mode coding.

[00362] In some embodiments, the motion candidates correspond to motion candidates that include illumination compensation parameters for IC parameter coding.

[00363] FIG. 42 is a flowchart of an example method 4200 for video processing in accordance with the presently disclosed technology. The method 4200 includes, at operation 4202, determining a new motion candidate for video processing by using one or multiple motion candidate from one or multiple tables, wherein a table includes one or multiple of motion candidates and each motion candidate is associated motion information; at operation 4204, performing a conversion between a video block and a coded representation of the video block based on the new candidate.

[00364] In some embodiments, the method further comprises: adding the determined new candidate to a candidate list, comprising a merge or Advanced Motion Vector Prediction (AMVP) candidate list.

[00365] In some embodiments, determining the new candidate further includes: determining the new motion candidate as a function of one or more motion candidates from the one or multiple tables, and advanced motion vector prediction (AMVP) candidate in the AMVP candidate list or merge candidates in the merge candidate list.

[00366] 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.

[00367] 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.

[00368] 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.

[00369] 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).

[00370] 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.

[00371] It is intended that the specification, together with the drawings, be considered exemplary only, where exemplary means an example. As used herein, the singular forms "a", "an" and "the" are intended to include the plural forms as well, unless the context clearly indicates otherwise. Additionally, the use of "or" is intended to include "and/or", unless the context clearly indicates otherwise.

[00372] 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.

[00373] 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.

[00374] 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, a new candidate for video processing by always using motion information from more than one spatial neighboring block of a first video block in current picture and without using motion information from temporal blocks in a picture different from the current picture; and

performing a conversion between the first video block in the current picture of a video and a bitstream representation of the video by using the determined new candidate.

2. The method of claim 1, wherein the determined new candidate is added to a candidate list, comprising a merge candidate list or Advanced Motion Vector Prediction (AMVP) candidate list.

3. The method of any one of claim 1 to 2, wherein the motion information from more than one spatial neighboring blocks are candidates derived from pre-defined spatial neighboring blocks relative to the first video block in the current picture or motion candidates from one or more tables.

4. The method of claim 3, wherein the one or more tables include motion candidates derived from previously processed video blocks that are processed prior to the first video block in video data.

5. The method of claim 3, wherein the candidates derived from pre-defined spatial neighboring blocks relative to the first video block in the current picture are spatial merge candidates.

6. The method of any one of claim 1 to 5, wherein the new candidate is derived by averaging at least two spatial merge candidates.

7. The method of any one of claim 1 to 5, wherein the new candidate is derived by jointly using the spatial merge candidates and motion candidates from one or more tables.

8. The method of any one of claim 1 to 5, wherein the new candidate is derived by averaging at least two motion vectors associated with candidates derived from different locations.

9. The method of claim 8, wherein the different locations are neighbouring to the first video block.

10. The method of claim 9, wherein the motion vectors only are fetched from locations which are in a current largest coding unit to which the first video block belongs.

11. The method of claim 9, wherein the motion vectors only are fetched from locations which are in a current largest coding unit line.

12. The method of claim 9, wherein the motion vectors only are fetched from locations which are in a current largest coding unit line or next to the current largest coding unit line.

13. The method of claim 9, wherein the motion vectors only are fetched from locations which are in the current largest coding unit line or next to the current largest coding unit line but not at left of the top-left neighbouring block.

14. The method of claim 8, wherein the motion vector for the bottom-right block for the planar motion prediction is not fetched from temporal motion vector prediction candidates but from one entry of the table.

15. The method of any one of claim 1 to 5, wherein the new candidate is derived by jointly using motion candidates from one or more tables and other kinds of merge/AMVP candidates.

16. The method of any one of claims 3-15, wherein the motion candidate in the one or more tables is associated with motion information which includes at least one of: a prediction direction, a reference picture index, motion vector values, intensity compensation flag, affine flag, motion vector difference precision, or motion vector difference value.

17. The method of any one of claims 3-16, further comprising updating, based on the conversion, one or more tables.

18. The method of claim 17, wherein the updating of one or more tables includes updating one or more tables based on the motion information of the first video block after performing the conversion.

19. The method of claim 18, further comprising:

performing a conversion between a subsequent video block of the video and the bitstream representation of the video based on the updated tables.

20. The method of any one of claims 1-19, wherein the conversion includes an encoding process and/or decoding process.

21. A method for video processing, comprising:

determining, a new candidate for video processing by using motion information from at least one spatial non-adjacent block of a first video block in current picture and other candidates derived from spatial non-adjacent or not from spatial non-adjacent block of the first video block; and

performing a conversion between the first video block of a video and a bitstream representation of the video by using the determined new candidate.

22. The method of claim 21, wherein the determined new candidate is added to a candidate list, comprising a merge candidate list or Advanced Motion Vector Prediction (AMVP) candidate list.

23. The method of any one of claim 21 to 22, wherein the motion information from more than one spatial non-adjacent blocks are candidates derived from pre-defined spatial non-adjacent blocks relative to the first video block in the current picture.

24. The method of claim 23, wherein the candidates derived from pre-defined spatial non-adjacent blocks relative to the first video block in the current picture are spatial-temporal motion vector prediction (STMVP) candidate.

25. The method of any one of claims 21-24, wherein the non-adjacent blocks of the video block are not right or left neighboring block of the first video block.

26. The method of claim 25, wherein above blocks of the first video block used for STMVP candidate derivation are kept unchanged, while the used left blocks are changed from neighboring blocks to non-adjacent blocks.

27. The method of claim 25, wherein left blocks of the first video block used for STMVP candidate derivation are kept unchanged, while the used above blocks are changed from neighboring blocks to non-adjacent blocks.

28. A method for video processing, comprising:

determining, a new candidate for video processing by using motion information from one or more tables of a first video block in current picture and motion information from temporal blocks in a picture different from the current picture; and

performing a conversion between the first video block in the current picture of a video and a bitstream representation of the video by using the determined new candidate.

29. The method of claim 28, wherein the determined new candidate is added to a candidate list, comprising a merge candidate list or Advanced Motion Vector Prediction (AMVP) candidate list.

30. The method of any one of claims 28 to 29, wherein the motion information from one or more tables in current picture are associated with one or more History Motion Vector Prediction (HMVP) candidates selected from one or more tables, and the motion information from temporal blocks in a picture different from the current picture are temporal motion candidates.

31. The method of claim 30, wherein the new candidate is derived by averaging one or more HMVP candidates and one or more temporal motion candidates.

32. The method of any one of claims 28-31, wherein the one or more tables include motion candidates derived from previously processed video blocks that are processed prior to the first video block in video data.

33. A method for video processing, comprising:

determining, a new candidate for video processing by using motion information from one or more tables of a first video block and motion information from one or more spatial neighboring block of the first video block; and

performing a conversion between the first video block in the current picture of a video and a bitstream representation of the video by using the determined new candidate.

34. The method of claim 33, wherein the determined new candidate is added to a candidate list, comprising a merge candidate list or Advanced Motion Vector Prediction (AMVP) candidate list.

35. The method of any one of claims 33 to 34, wherein the motion information from one or more tables of a first video block are associated with one or more History Motion Vector Prediction (HMVP) candidates selected from one or more tables, and the motion information from one or more spatial neighboring block of the first video block are candidates derived from pre-defined spatial blocks relative to the first video block.

36. The method of claim 35, wherein the candidates derived from pre-defined spatial blocks relative to the first video block are spatial merge candidates.

37. The method of any one of claims 33 to 36, wherein the new candidate is derived by averaging one or more HMVP candidates and one or more spatial merge candidates.

38. The method of any one of claims 33-37, wherein the one or more tables include motion candidates derived from previously processed video blocks that are processed prior to the first video block in video data.

39. The method of any one of claims 1 to 38, wherein a motion candidate from a table is associated with motion information including at least one of: a prediction direction, a reference picture index, motion vector values, an intensity compensation flag, an affine flag, a motion vector difference precision, or motion vector difference value.

40. The method of any one of claims 1-39, further comprising updating, based on the conversion, one or more tables.

41. The method of any one of claims 1-39, wherein the updating of one or more tables includes updating one or more tables based on the motion information of the current video block after performing the conversion.

42. The method of claim 41, further comprising: performing a conversion between a subsequent video block of the video data and the bitstream representation of the video data based on the updated tables.

43. A method of video processing, comprising:

determining a new motion candidate for video processing by using one or multiple motion candidate from one or multiple tables, wherein a table includes one or multiple of motion candidates and each motion candidate is associated motion information; and

performing a conversion between a video block and a coded representation of the video block based on the new candidate.

44. The method of claim 43, wherein the determined new candidate is added to a candidate list, comprising a merge candidate list or Advanced Motion Vector Prediction (AMVP) candidate list.

45. The method of claim 44, wherein the determining the new candidate includes:

determining the new motion candidate as a function of one or more motion candidates from the one or multiple tables, and advanced motion vector prediction (AMVP) candidate in the AMVP candidate list or merge candidates in the merge candidate list.

46. An apparatus in a video system comprising a processor configured to implement a method recited in one or more of claims 1 to 45.

47. A computer-readable program medium having code stored thereupon, the code comprising instructions that, when executed by a processor, causing the processor to implement a method recited in one or more of claims 1 to 45.

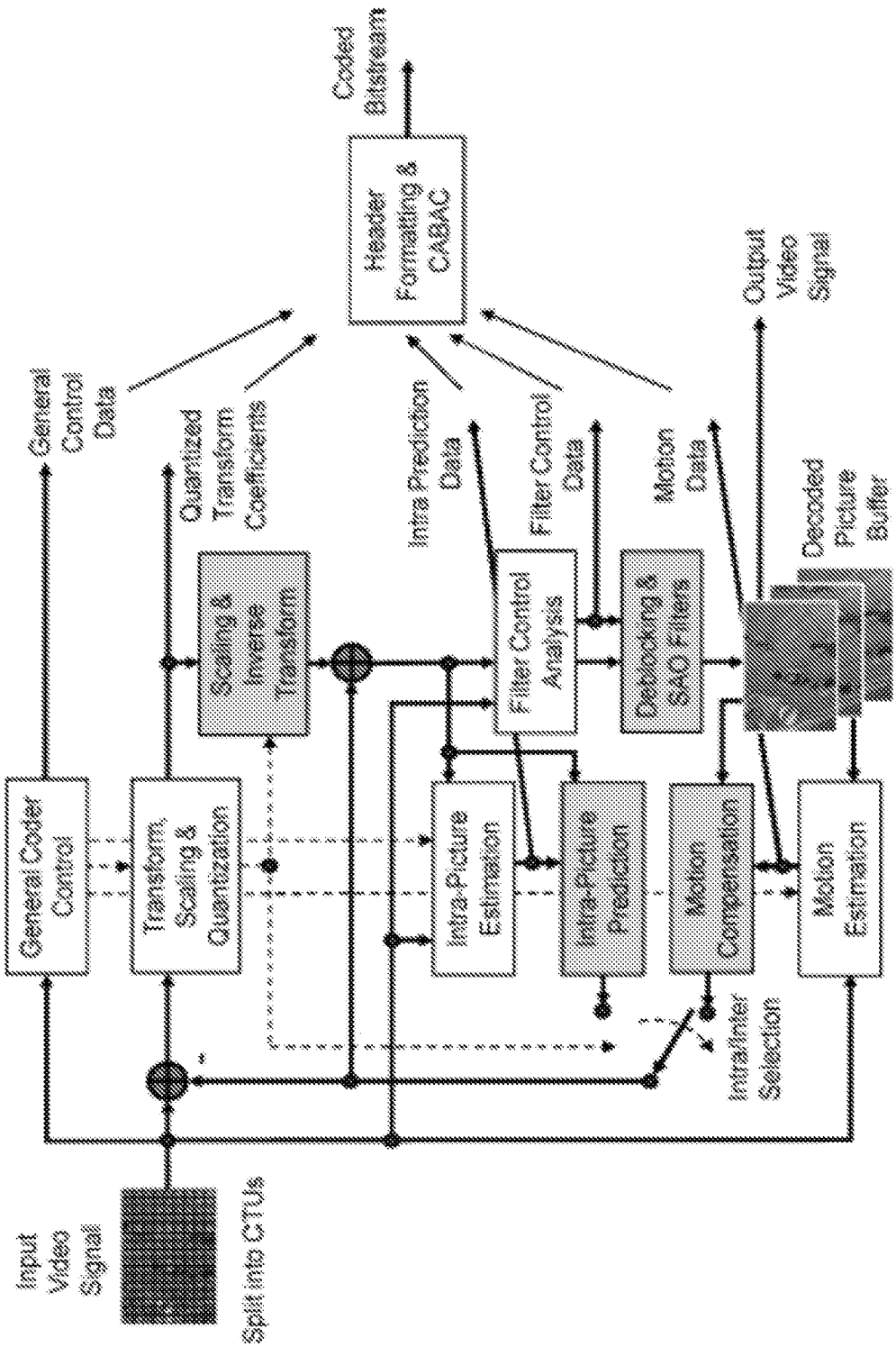

Input Video Signal
Split into CTUs
General Coder Control
Transform, Scaling & Quantization
Scaling & Inverse Transform
Intra-Picture Estimation
Intra-Picture Prediction
Filter Control Analysis
Deblocking & SAO Filters
Motion Compensation
Motion Estimation
Header Formatting & CABAC
Coded Bitstream
General Control Data
Quantized Transform Coefficients
Intra Prediction Data
Filter Control Data
Motion Data
Intra/Inter Selection
Decoded Picture Buffer
Output Video Signal

FIG. 1

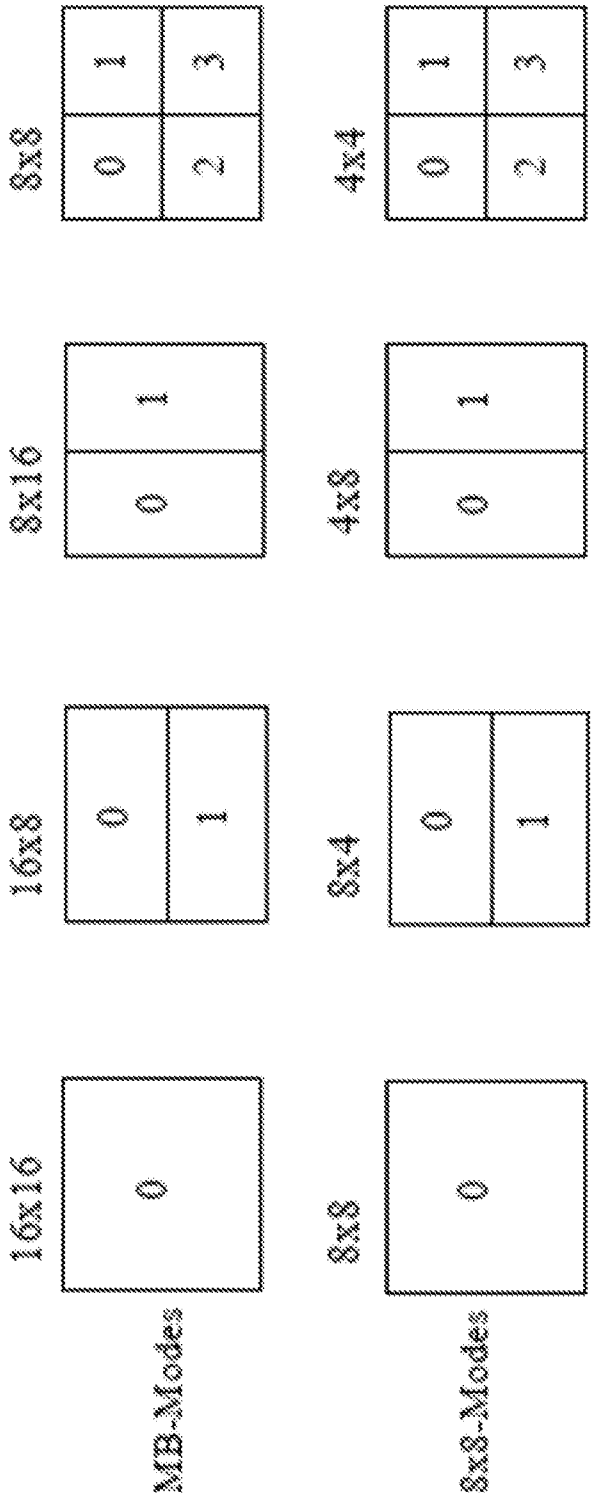

MB-Modes
16x16
0
16x8
0
1
8x16
0
1
8x8
0
1
2
3
8x8-Modes
8x8
0
8x4
0
1
4x8
0
1
4x4
0
1
2
3

FIG. 2

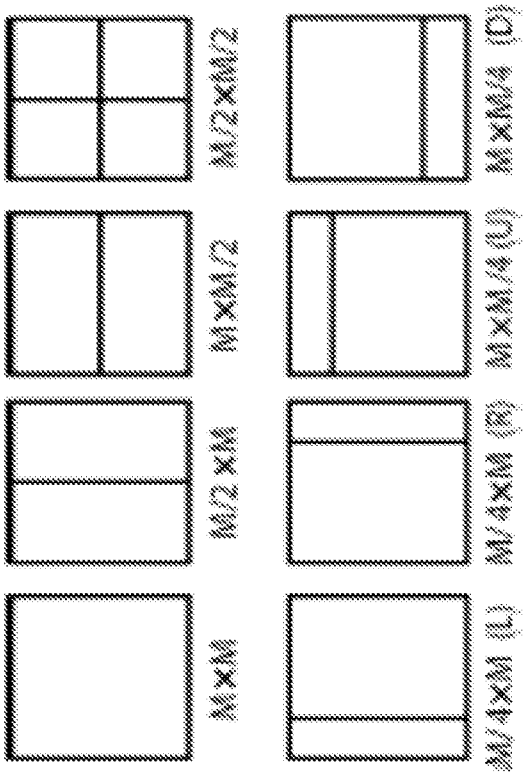
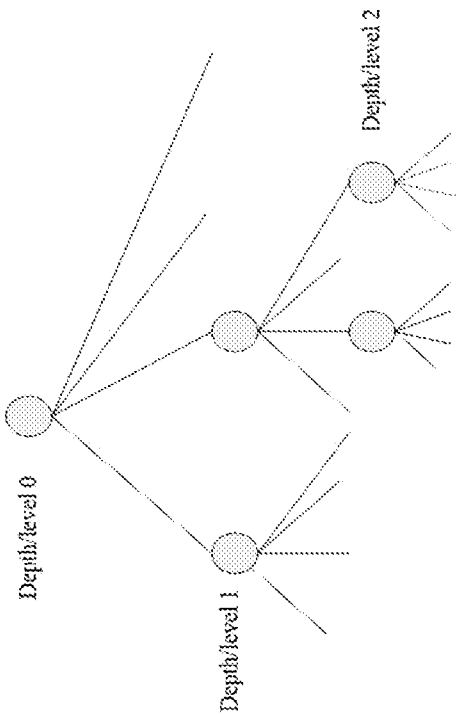


FIG. 3



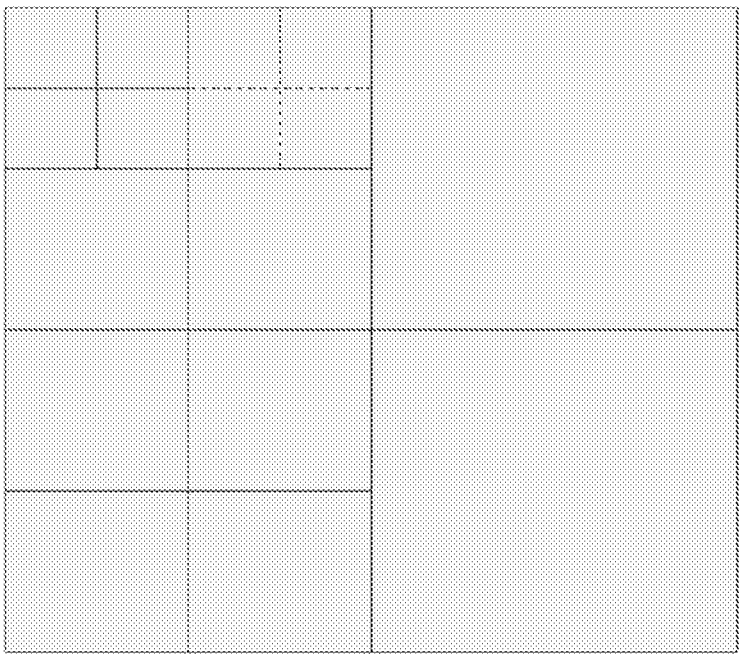


FIG. 4

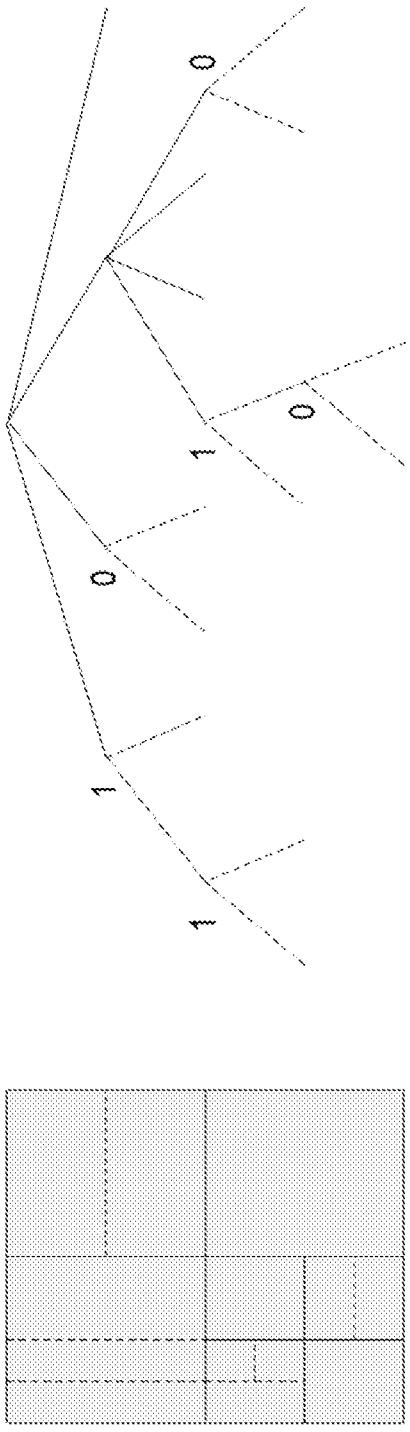


FIG. 5

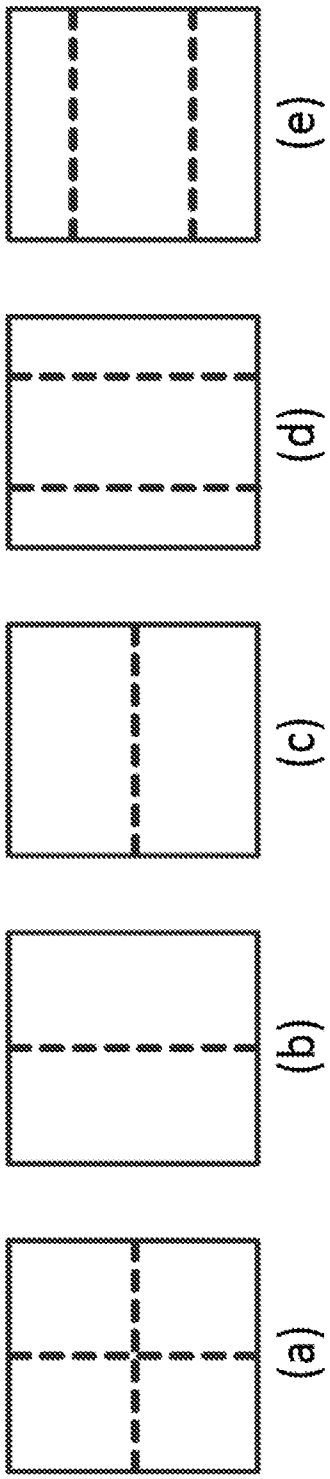


FIG. 6

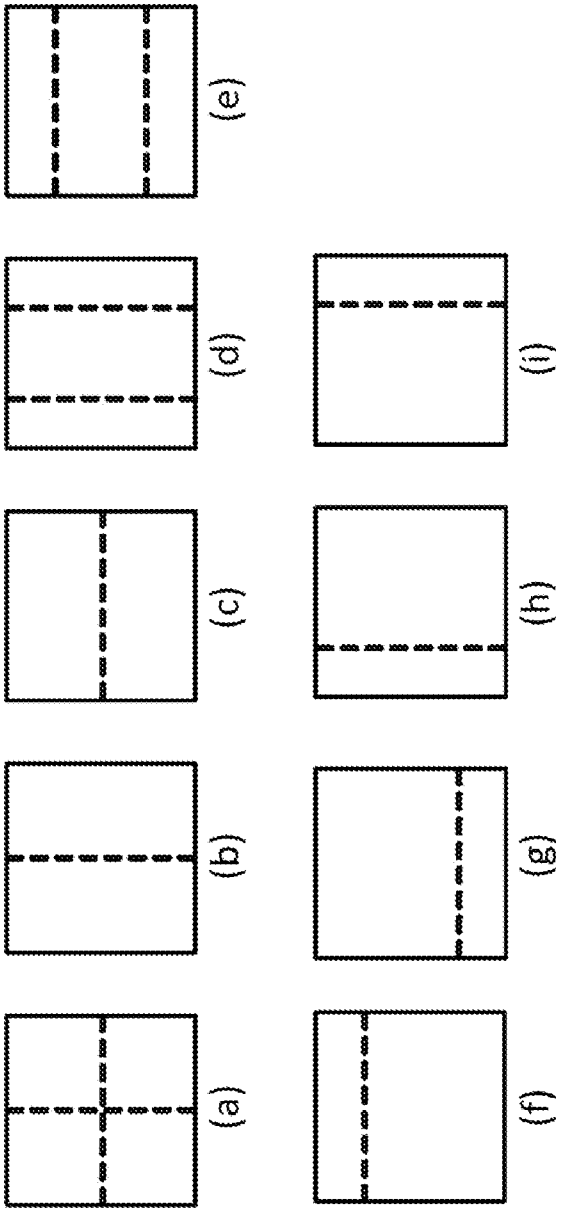


FIG. 7

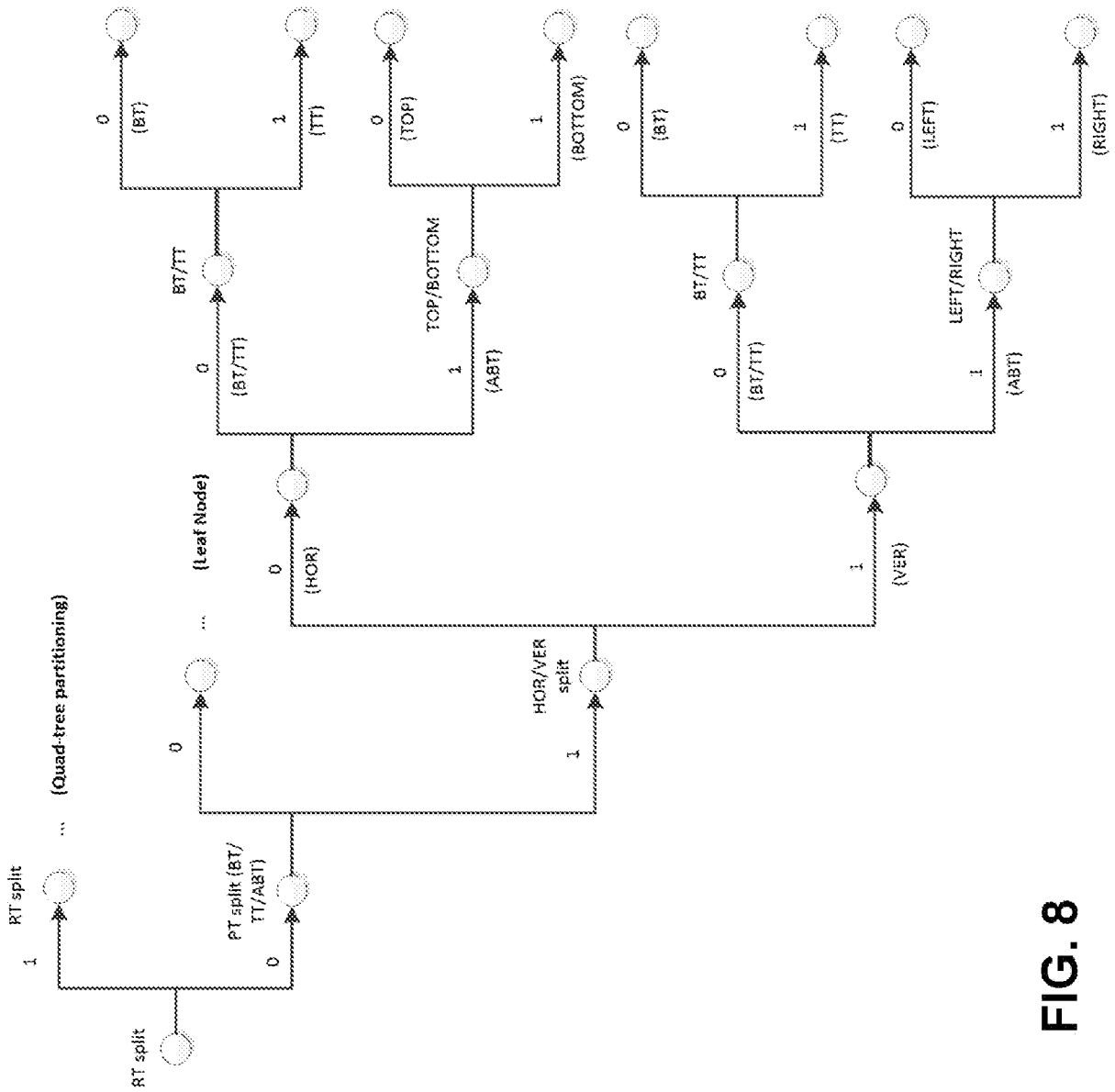

RT split
RT split
1
0
PT split (BT/TT/ABT)
0
1
... (Quad-tree partitioning)
0
... (Leaf Node)
HOR/VER split
1
0 (HOR)
1 (VER)
0 (BT/TT)
1 (ABT)
BT/TT
TOP/BOTTOM
0 (BT)
1 (TT)
0 (TOP)
1 (BOTTOM)
0 (BT/TT)
1 (ABT)
BT/TT
LEFT/RIGHT
0 (BT)
1 (TT)
0 (LEFT)
1 (RIGHT)

FIG. 8

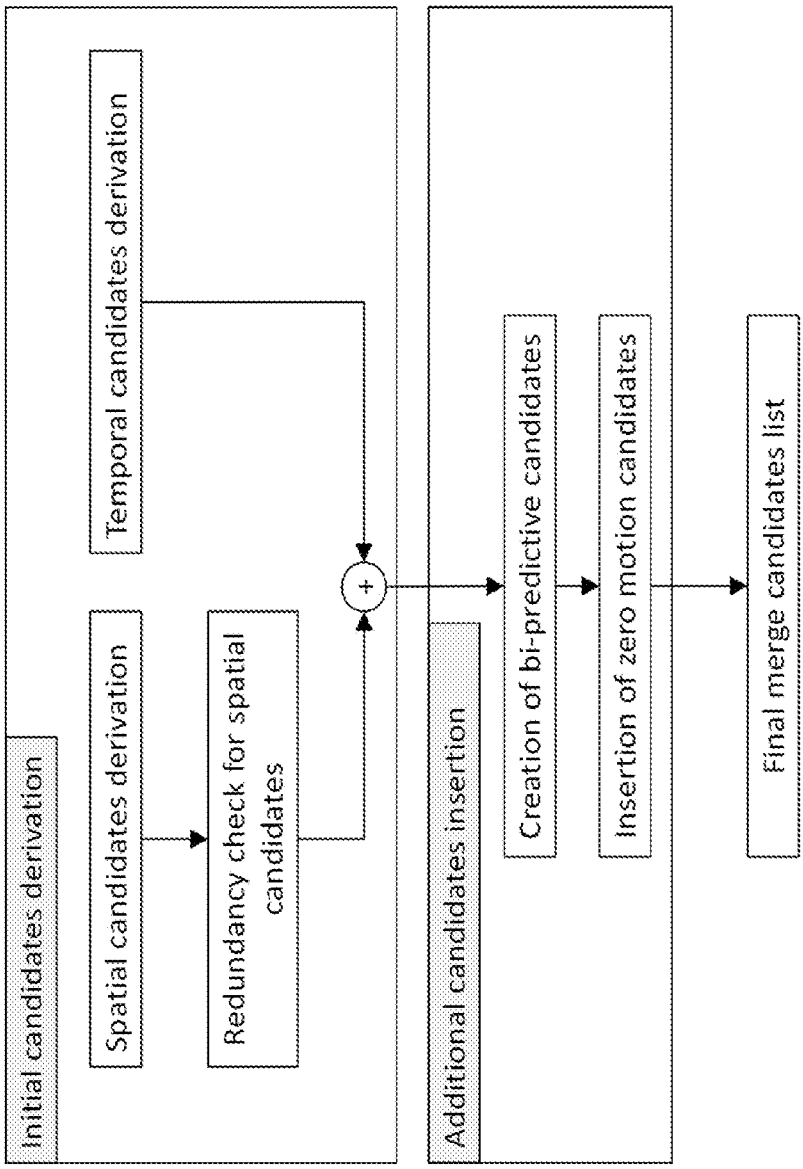

Initial candidates derivation
Spatial candidates derivation
Redundancy check for spatial candidates
Temporal candidates derivation
+
Additional candidates insertion
Creation of bi-predictive candidates
Insertion of zero motion candidates
Final merge candidates list

FIG. 9

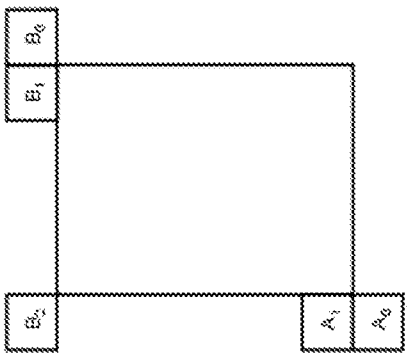


FIG. 10

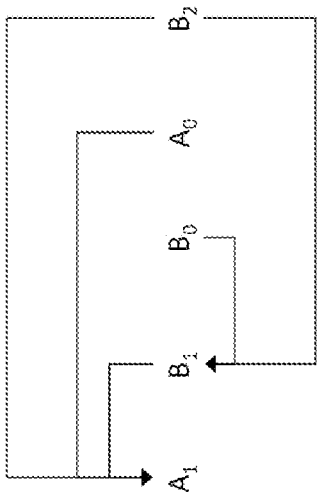


FIG. 11

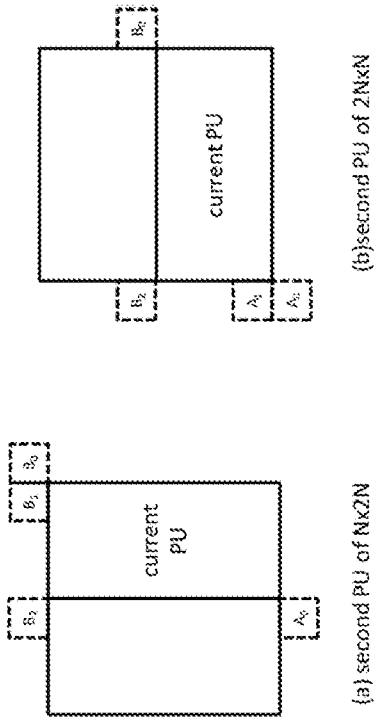


FIG. 12

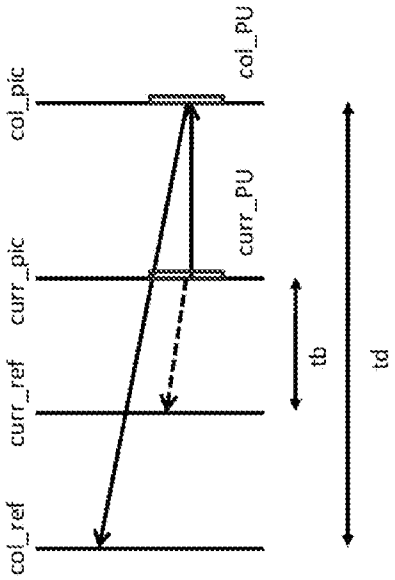


FIG. 13

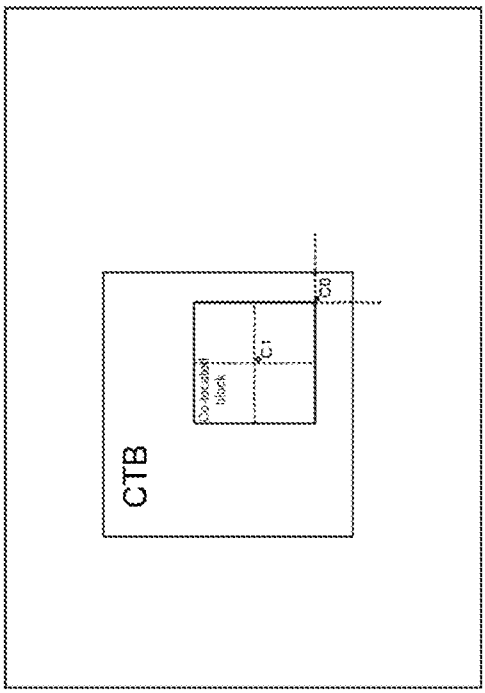


FIG. 14

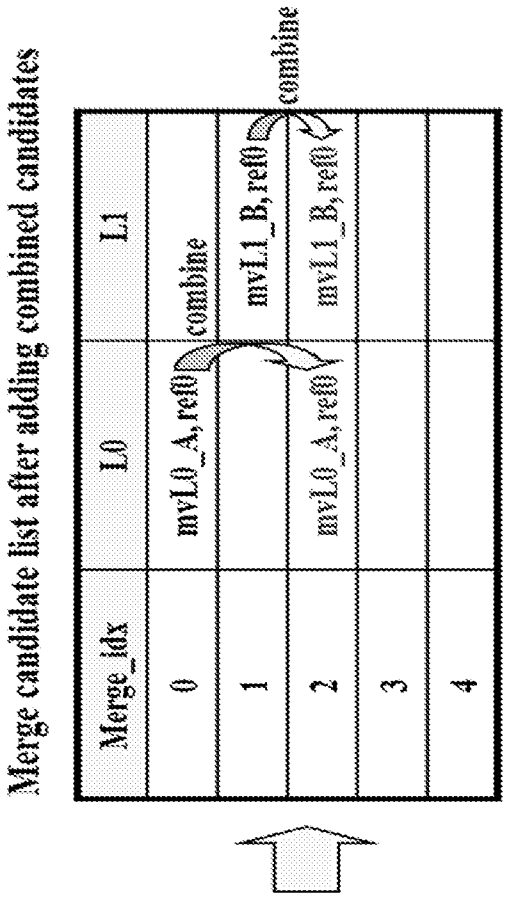

Original Merge candidate list
Merge_idx
L0
L1
0
mvL0_A,ref0
-
1
-
mvL1_B,ref0
2
3
4
Merge candidate list after adding combined candidates
Merge_idx
L0
L1
0
mvL0_A,ref0
combine
1
mvL1_B,ref0
combine
2
mvL0_A,ref0
mvL1_B,ref0
3
4

FIG. 15

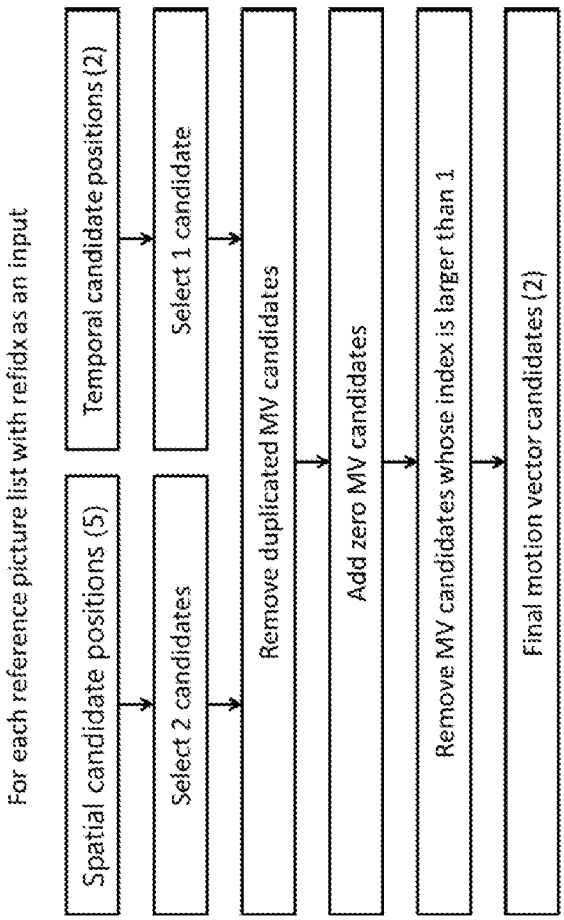

For each reference picture list with refidx as an input
Spatial candidate positions (5)
Temporal candidate positions (2)
Select 2 candidates
Select 1 candidate
Remove duplicated MV candidates
Add zero MV candidates
Remove MV candidates whose index is larger than 1
Final motion vector candidates (2)

FIG. 16

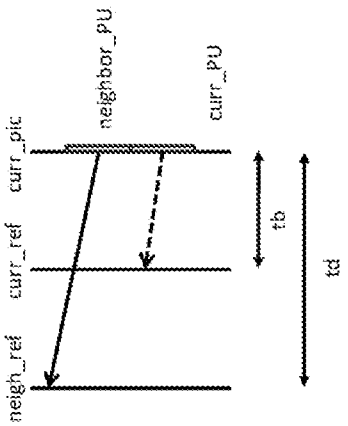


FIG. 17

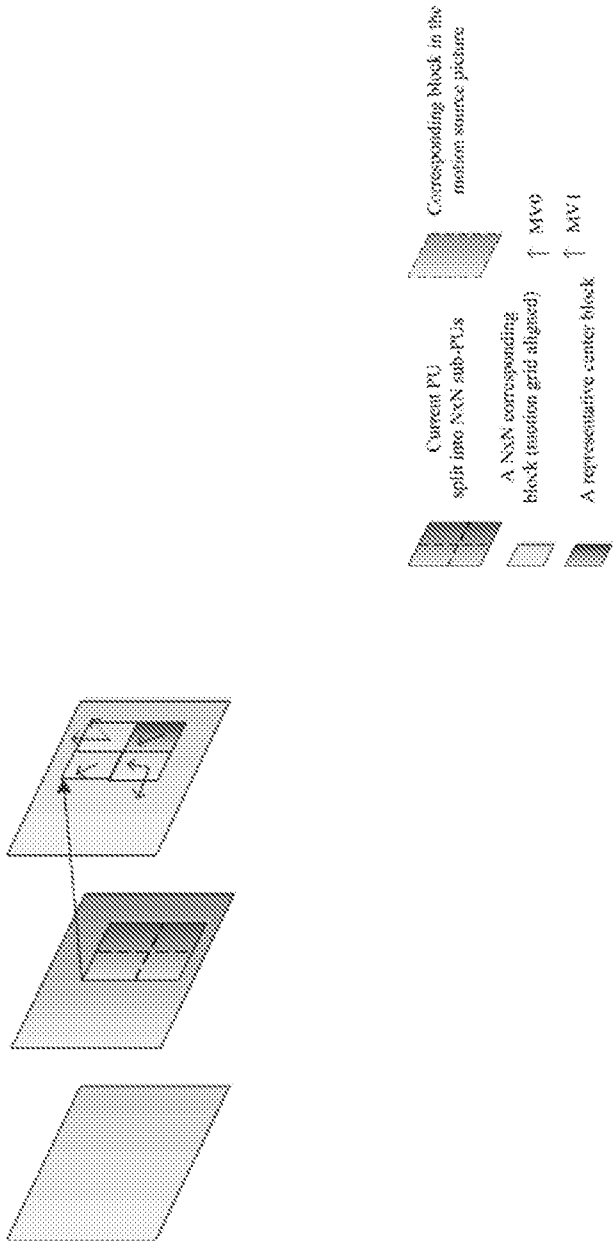

Current PU
split into NxN sub-PUs
A NxN corresponding
block (motion grid aligned)
A representative center block
MV0
MV1
Corresponding block in the
motion source picture

FIG. 18

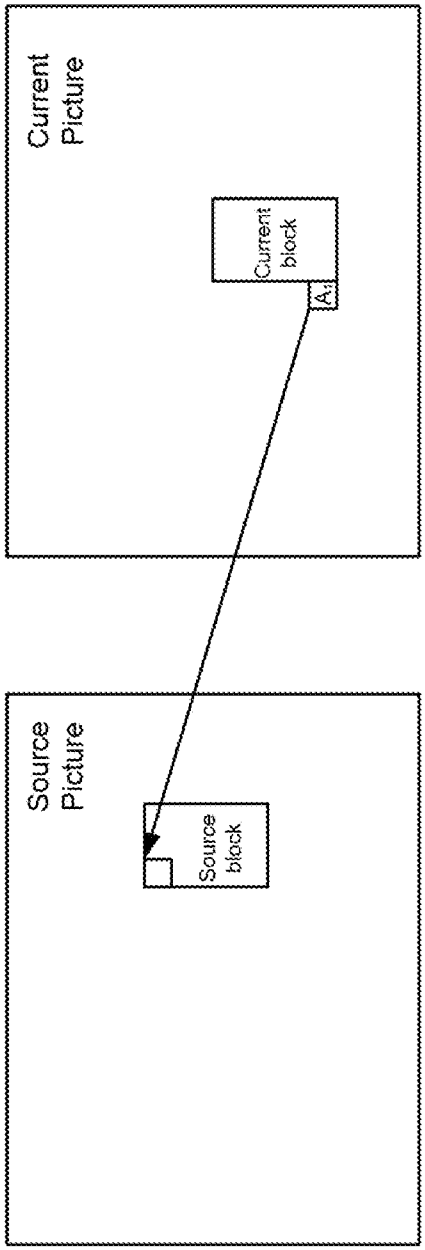

Current Picture
Current block
A1
Source Picture
Source block

FIG. 19

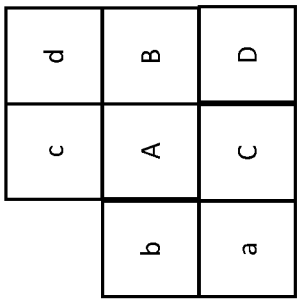


FIG. 20

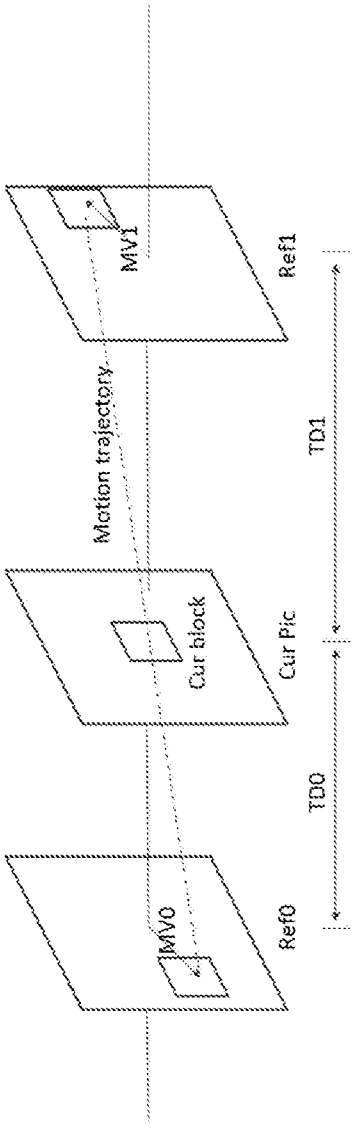

MV1
Ref1
Motion trajectory
TD1
Cur block
Cur Pic
TD0
MV0
Ref0

FIG. 21

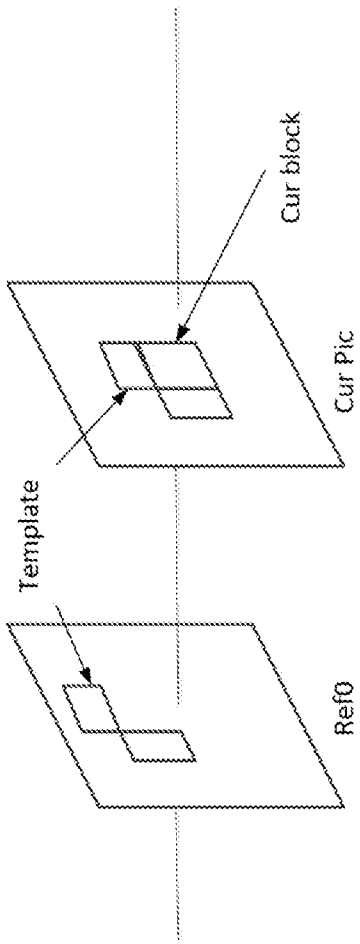


FIG. 22

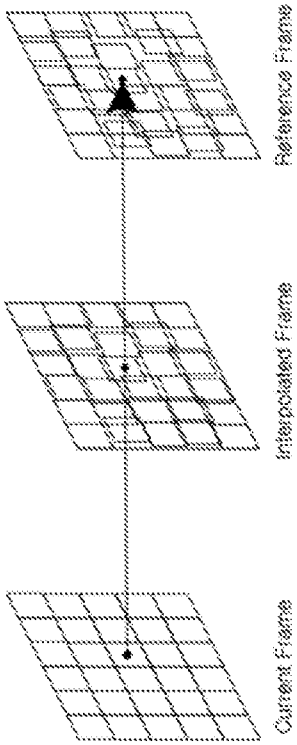


FIG. 23

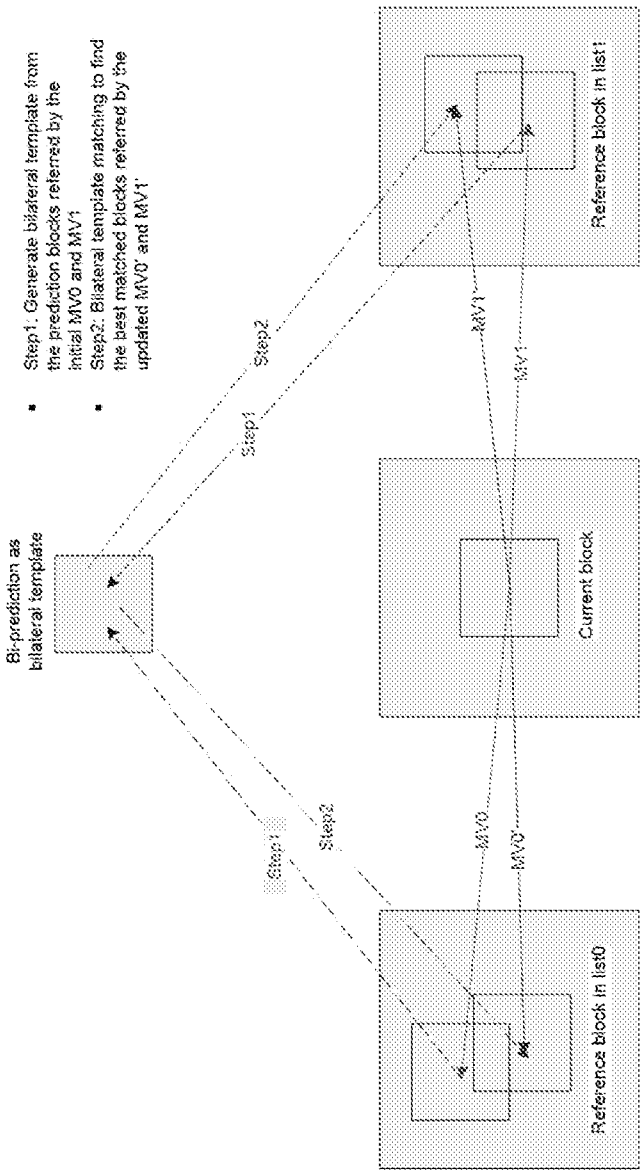

Step1: Generate bilateral template from the prediction blocks referred by the initial MV0 and MV1
Step2: Bilateral template matching to find the best matched blocks referred by the updated MV0' and MV1'
Bi-prediction as bilateral template
Step1
Step2
Step1
Step2
MV0
MV0'
MV1
MV1'
Reference block in list0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

FIG. 24

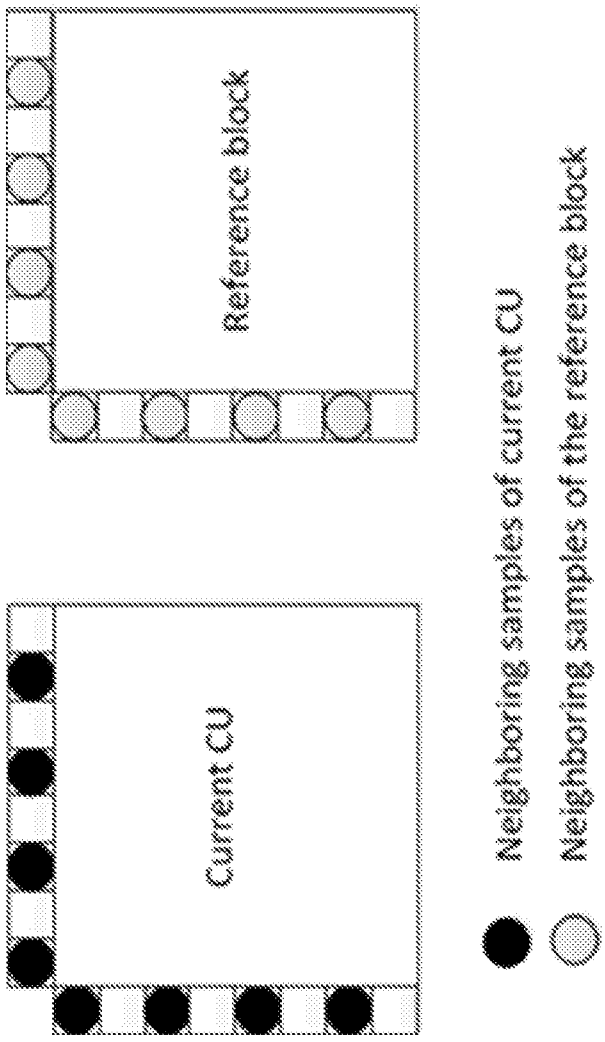

Reference block
Current CU
Neighboring samples of current CU
Neighboring samples of the reference block

FIG. 25

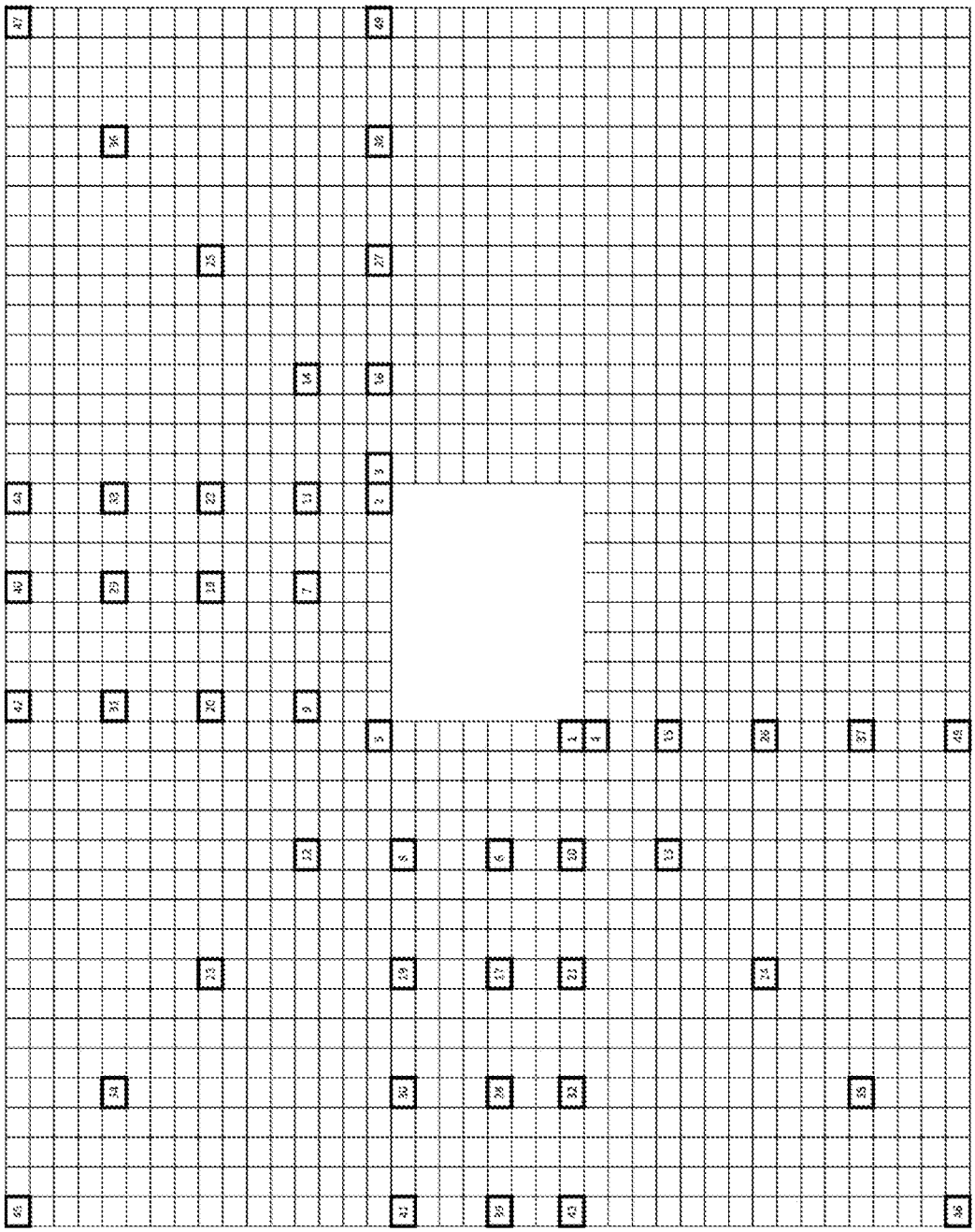


FIG. 26

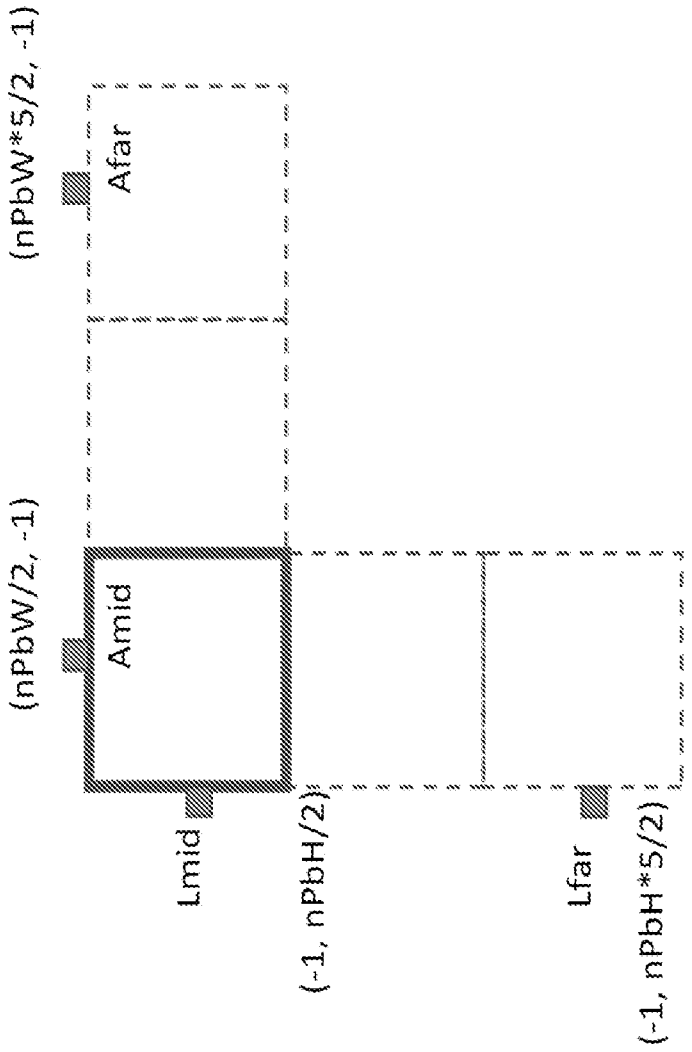


FIG. 27

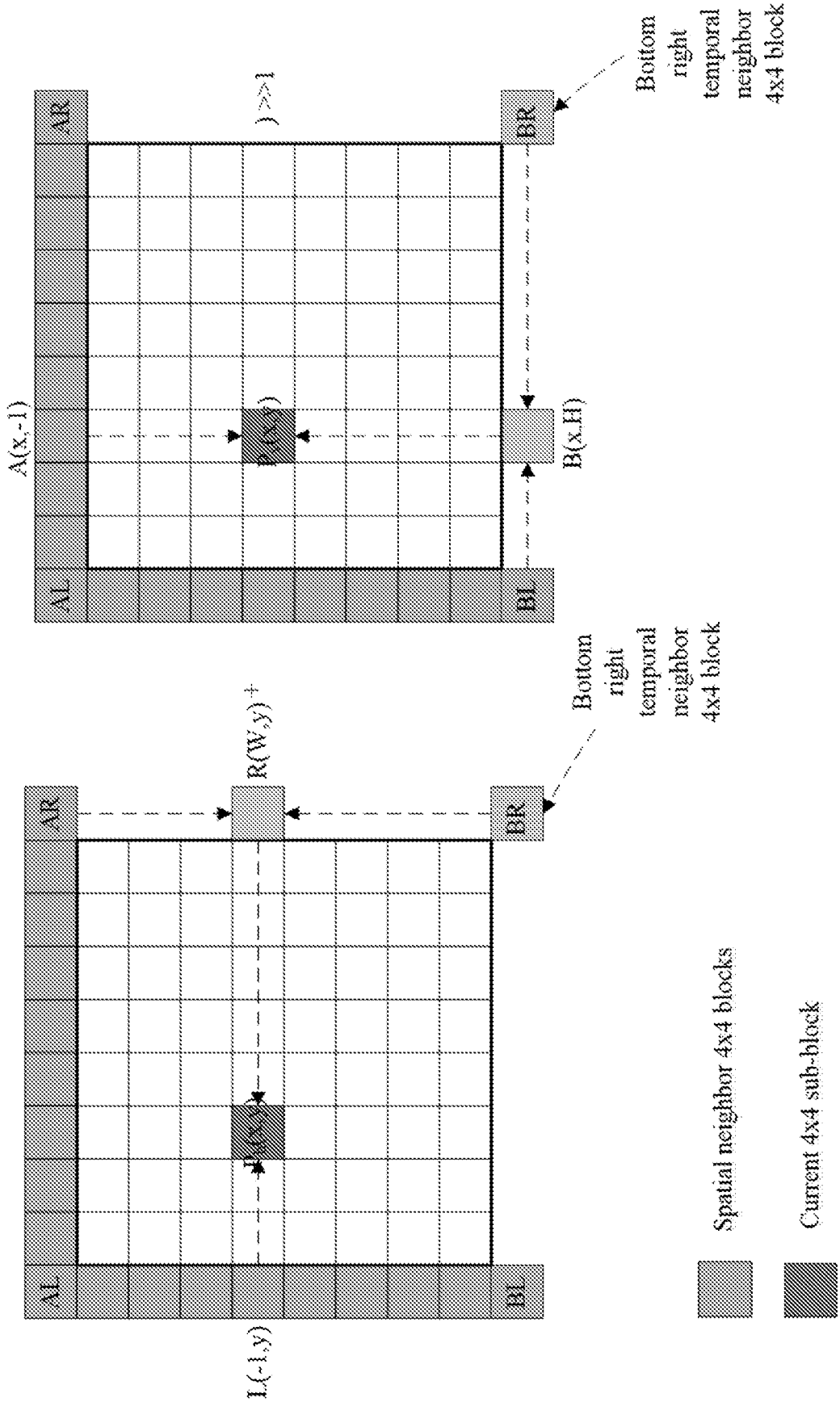

AR
AL
BR
BL
A(x,-1)
B(x,H)
R(W,y)
L(-1,y)
Bottom right temporal neighbor 4x4 block
Spatial neighbor 4x4 blocks
Current 4x4 sub-block

FIG. 28

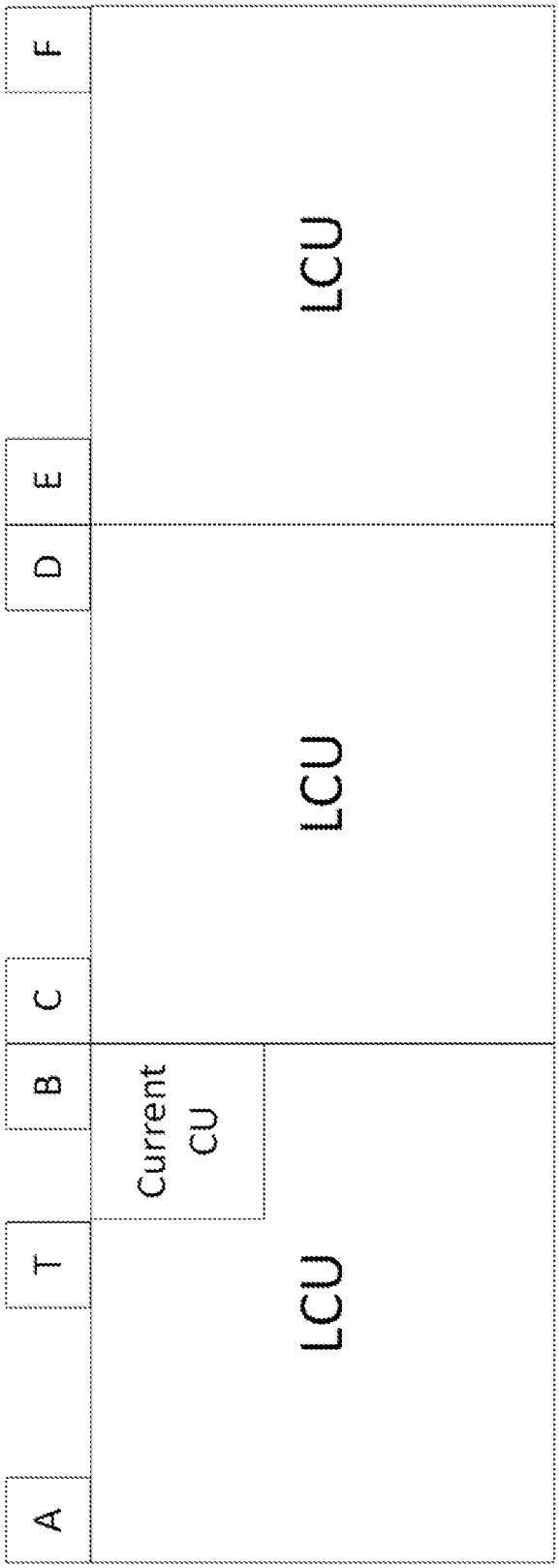


FIG. 29

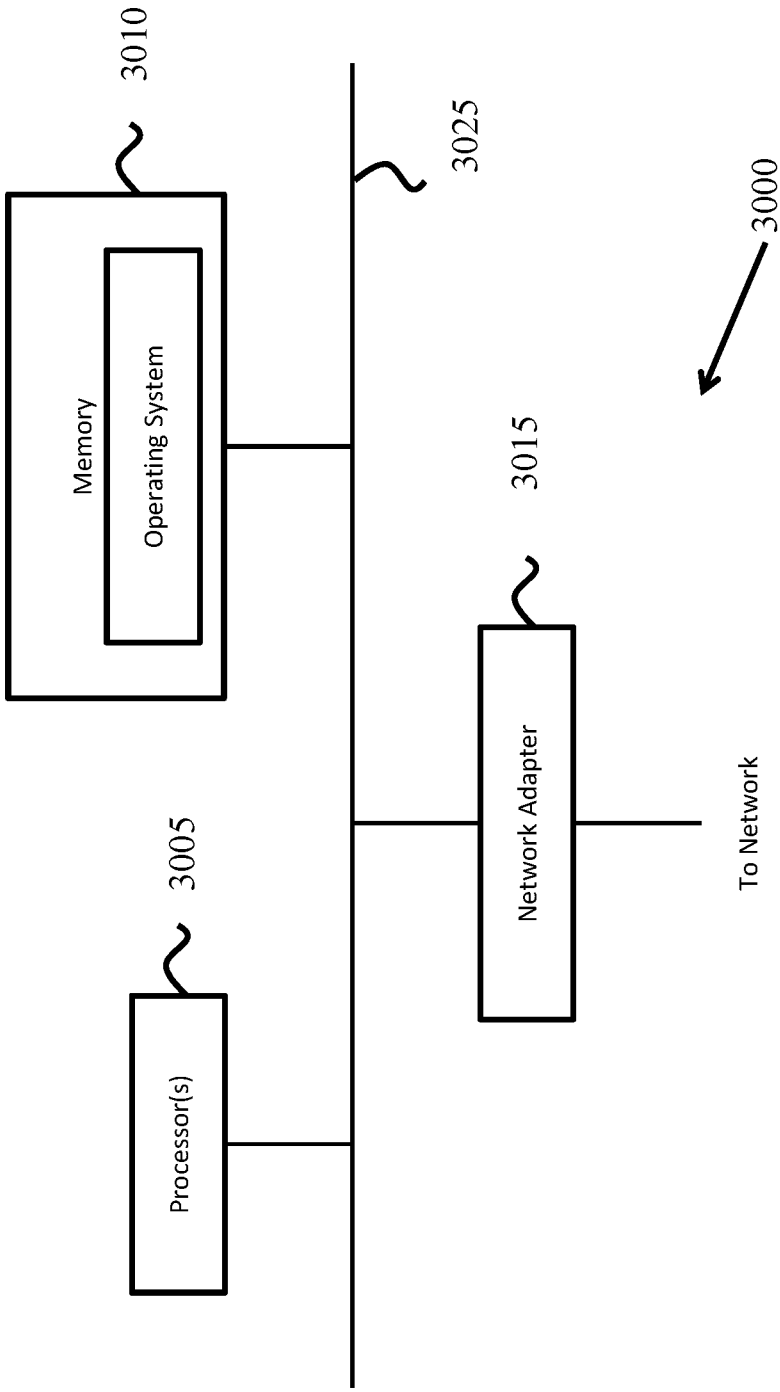

3010
3025
3000
Memory
Operating System
3015
Network Adapter
To Network
3005
Processor(s)

FIG. 30

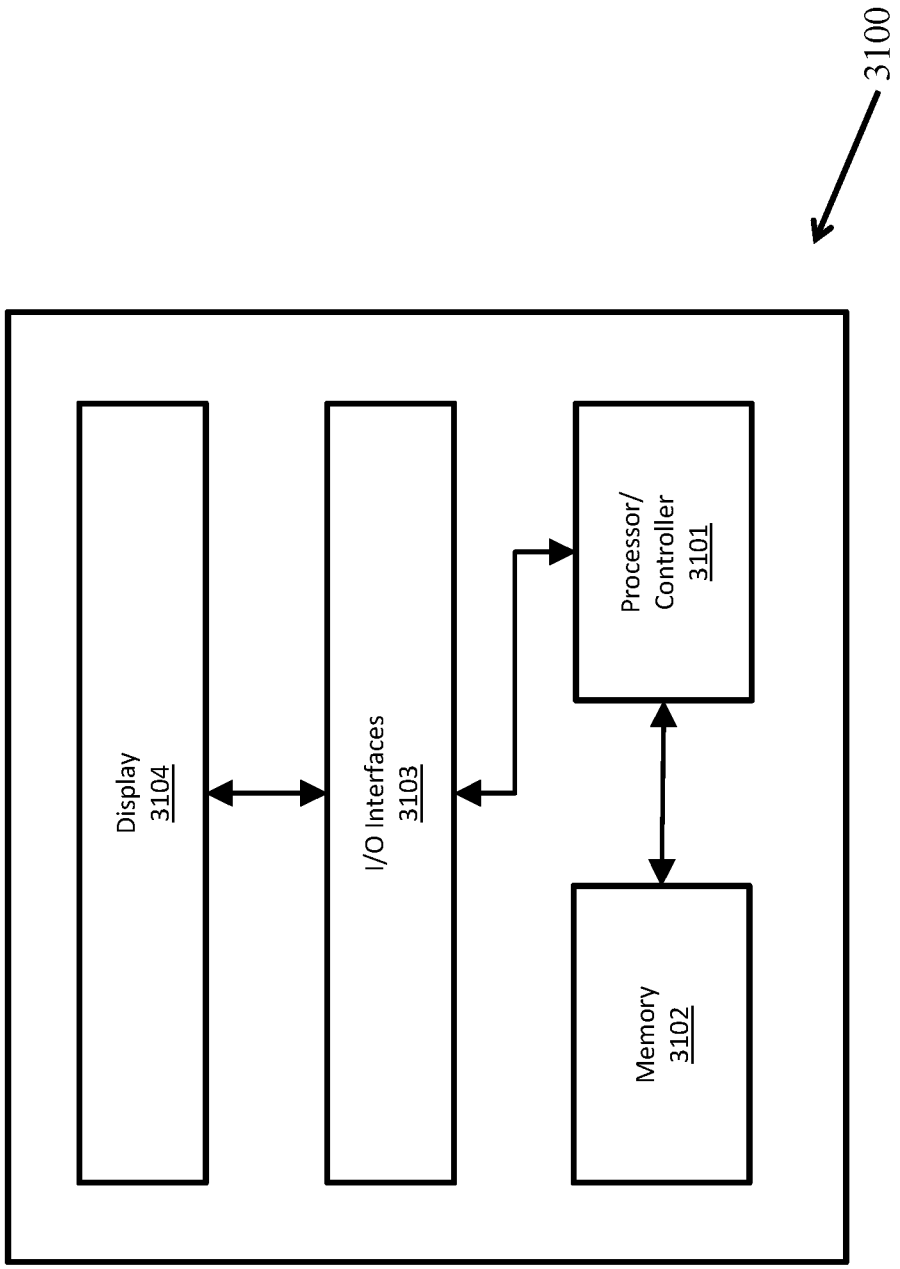

3100
Display
3104
I/O Interfaces
3103
Processor/
Controller
3101
Memory
3102

FIG. 31

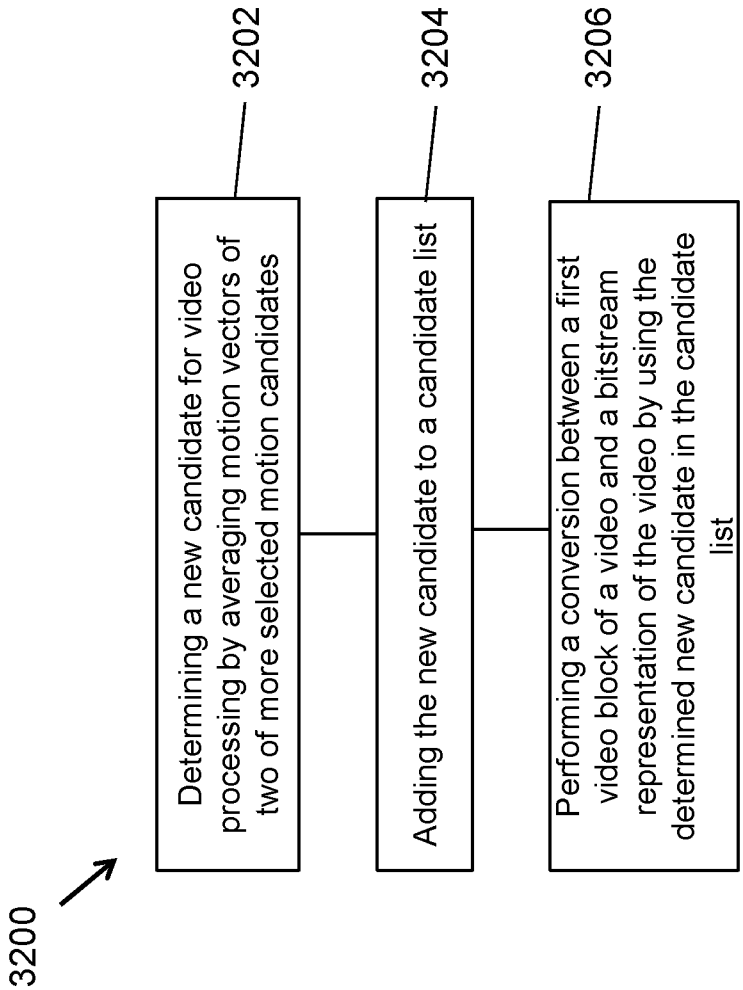

3200
Determining a new candidate for video processing by averaging motion vectors of two of more selected motion candidates
3202
Adding the new candidate to a candidate list
3204
Performing a conversion between a first video block of a video and a bitstream representation of the video by using the determined new candidate in the candidate list
3206

FIG. 32

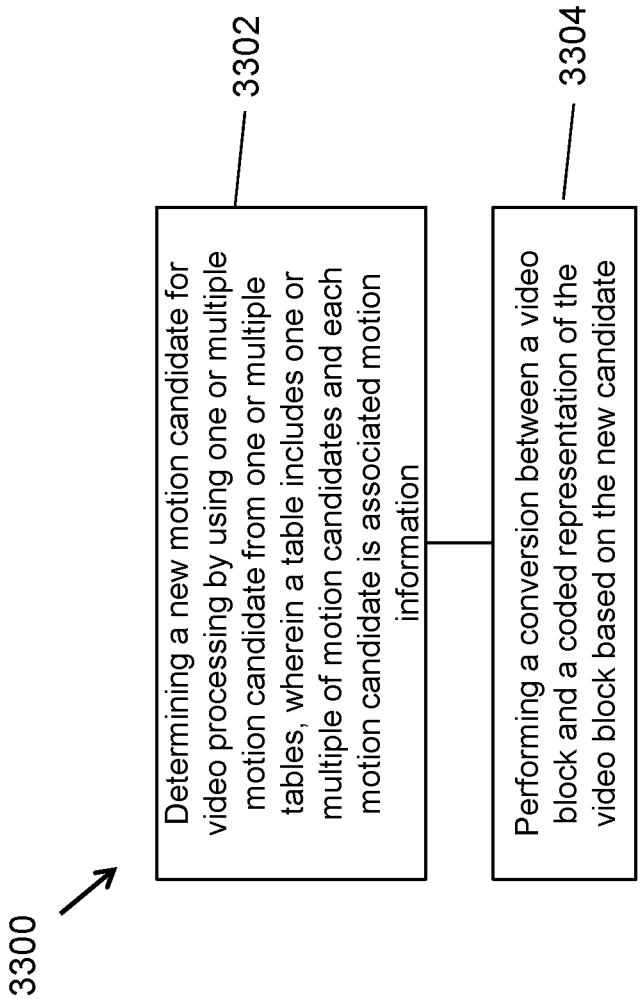

3300
3302
Determining a new motion candidate for video processing by using one or multiple motion candidate from one or multiple tables, wherein a table includes one or multiple of motion candidates and each motion candidate is associated motion information
3304
Performing a conversion between a video block and a coded representation of the video block based on the new candidate

FIG. 33

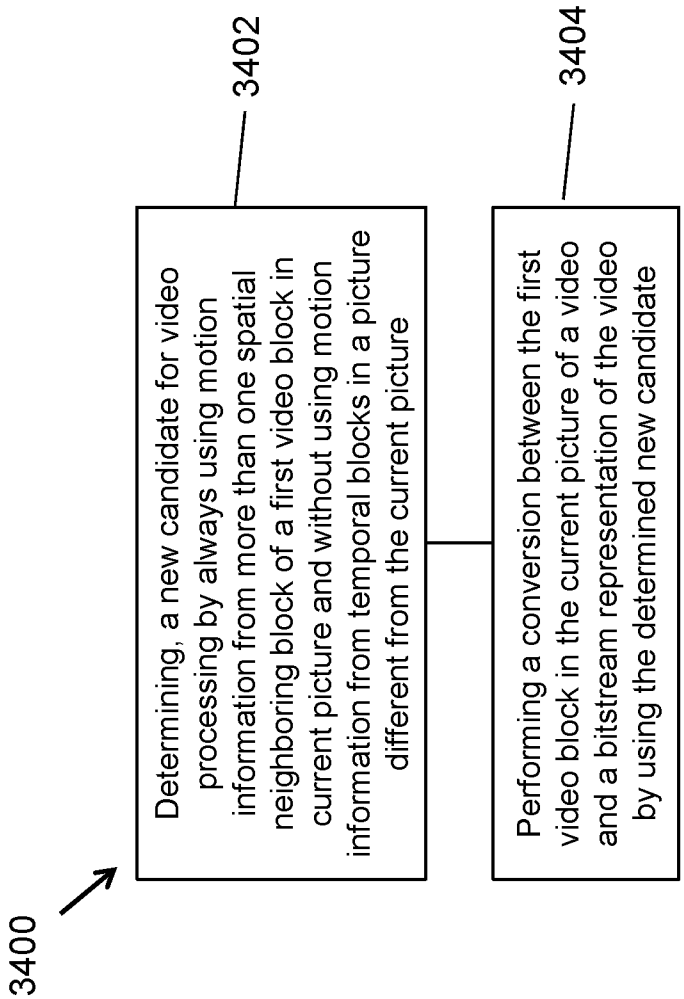

3400
Determining, a new candidate for video processing by always using motion information from more than one spatial neighboring block of a first video block in current picture and without using motion information from temporal blocks in a picture different from the current picture
3402
Performing a conversion between the first video block in the current picture of a video and a bitstream representation of the video by using the determined new candidate
3404

FIG. 34

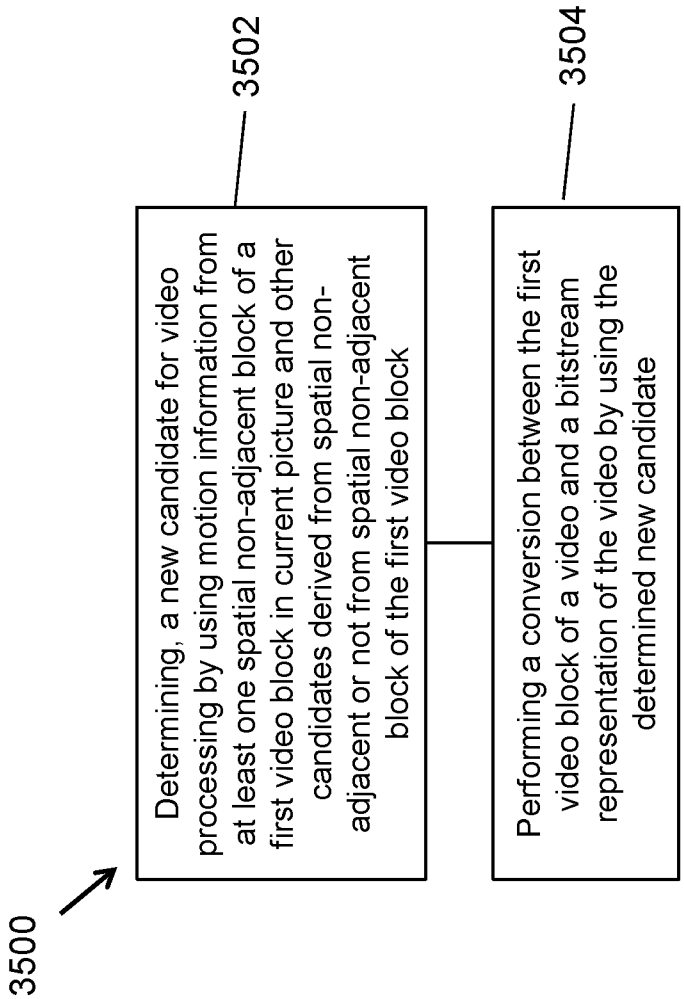

3500
Determining, a new candidate for video processing by using motion information from at least one spatial non-adjacent block of a first video block in current picture and other candidates derived from spatial non-adjacent or not from spatial non-adjacent block of the first video block
3502
Performing a conversion between the first video block of a video and a bitstream representation of the video by using the determined new candidate
3504

FIG. 35

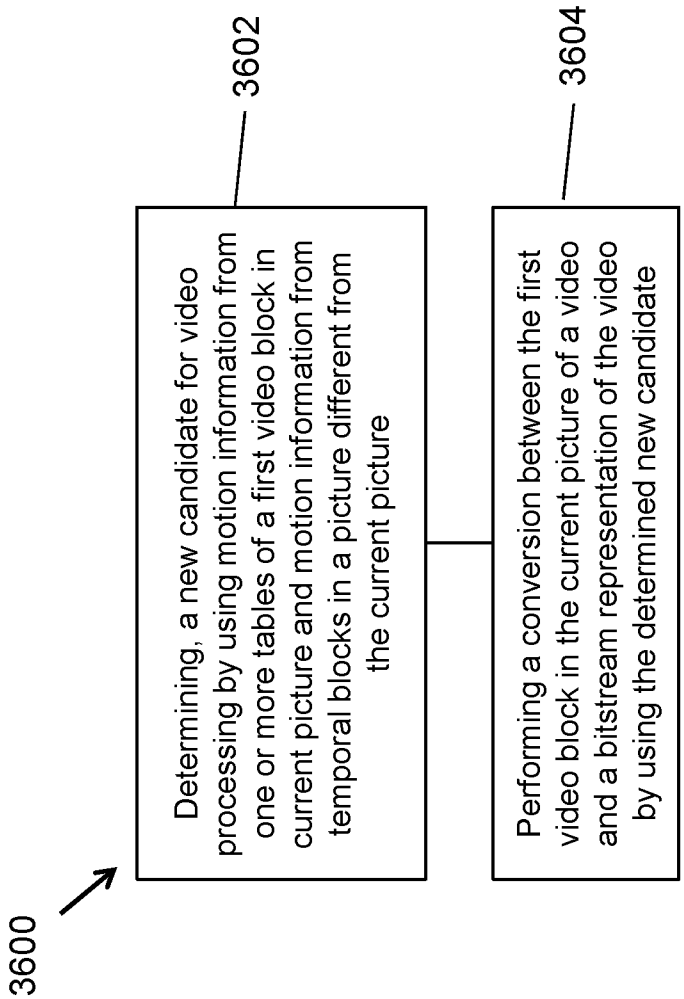

3600
Determining, a new candidate for video processing by using motion information from one or more tables of a first video block in current picture and motion information from temporal blocks in a picture different from the current picture
3602
Performing a conversion between the first video block in the current picture of a video and a bitstream representation of the video by using the determined new candidate
3604

FIG. 36

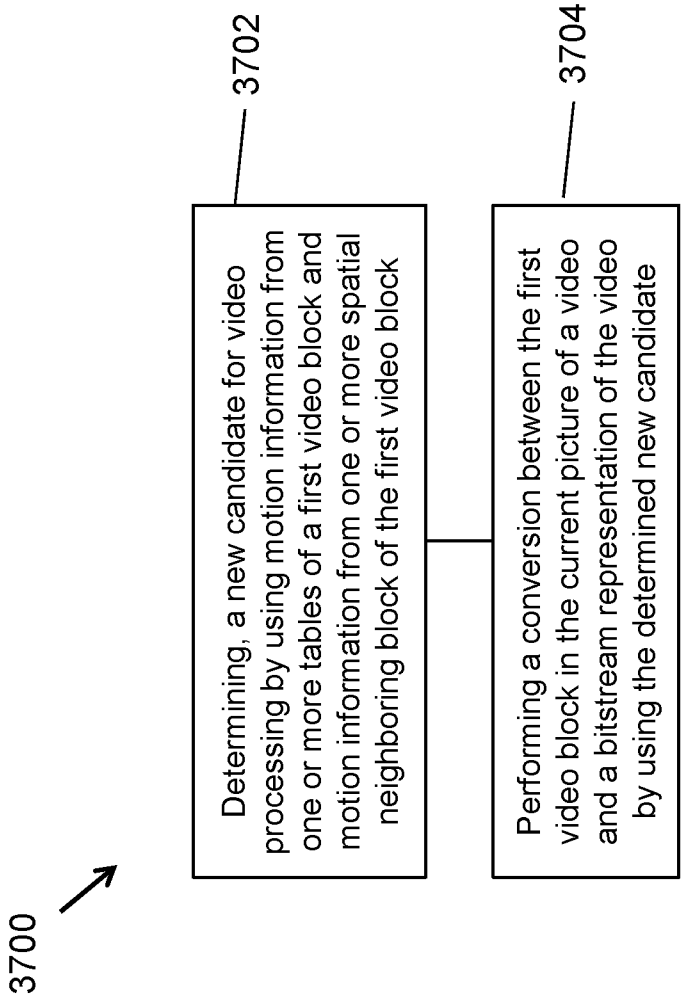

3700
3702
Determining, a new candidate for video processing by using motion information from one or more tables of a first video block and motion information from one or more spatial neighboring block of the first video block
3704
Performing a conversion between the first video block in the current picture of a video and a bitstream representation of the video by using the determined new candidate

FIG. 37

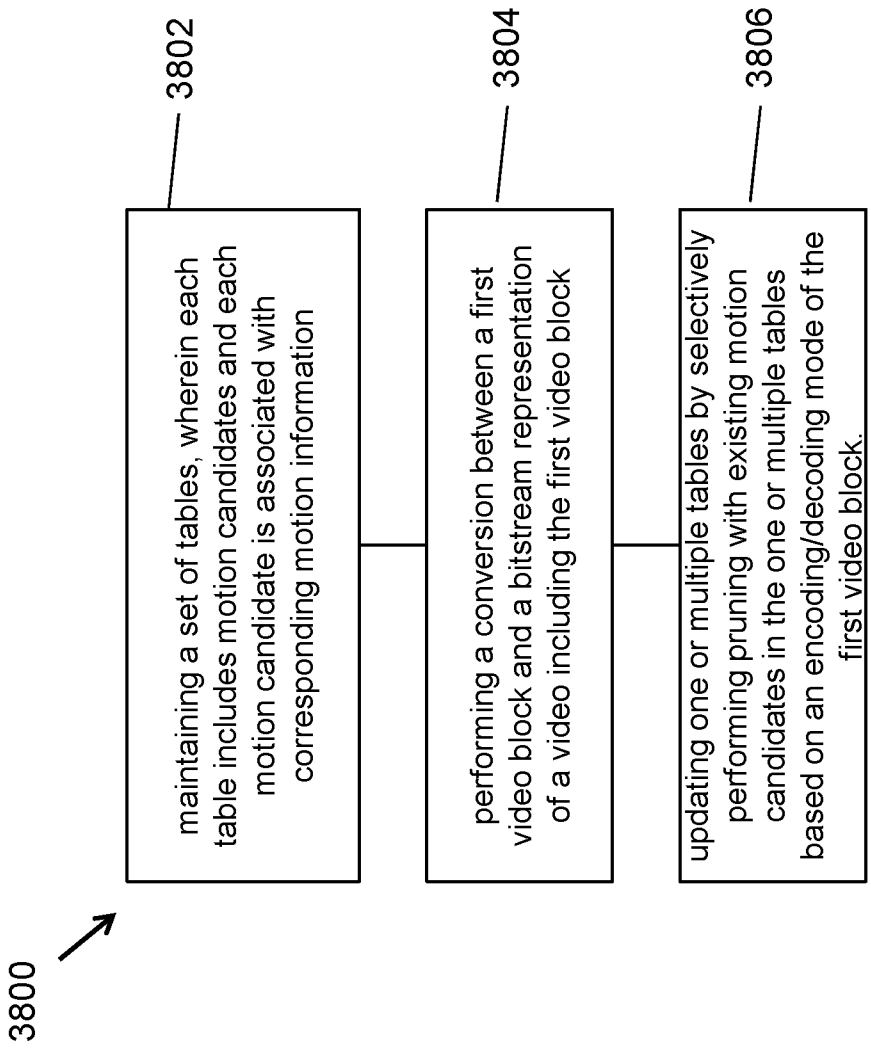

3800
3802
maintaining a set of tables, wherein each table includes motion candidates and each motion candidate is associated with corresponding motion information
3804
performing a conversion between a first video block and a bitstream representation of a video including the first video block
3806
updating one or multiple tables by selectively performing pruning with existing motion candidates in the one or multiple tables based on an encoding/decoding mode of the first video block.

FIG. 38

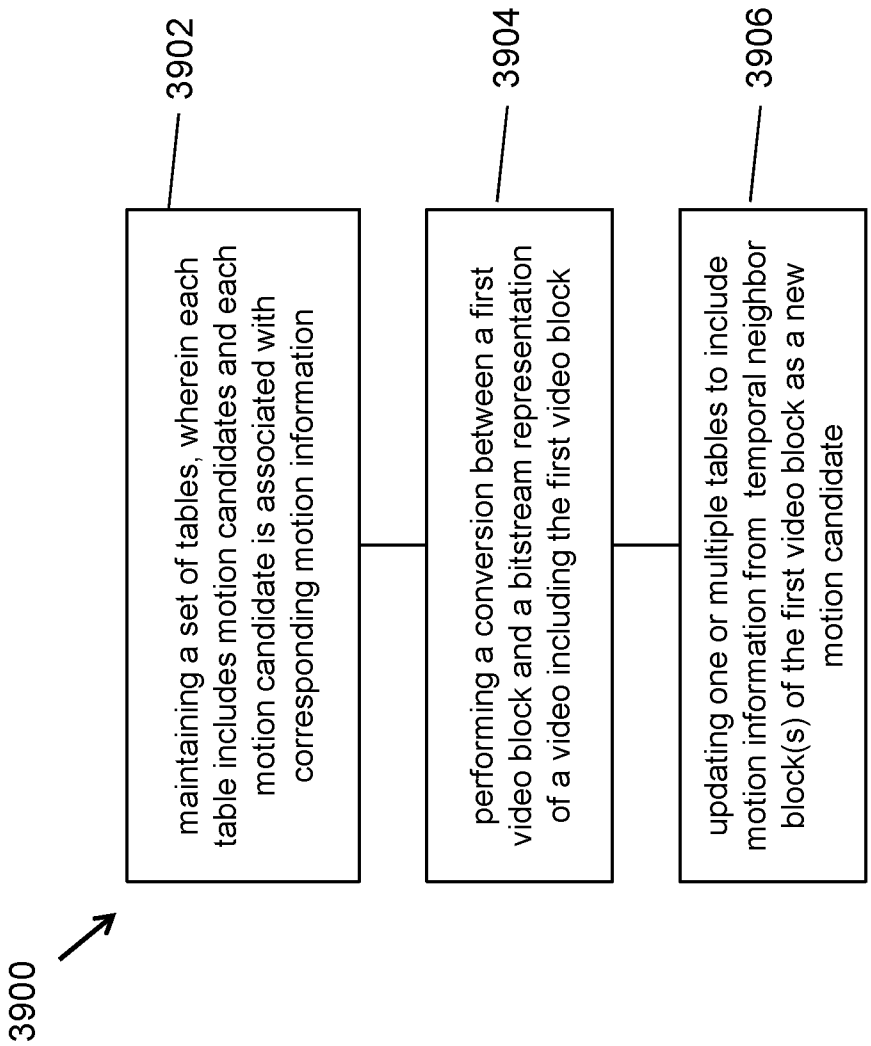

3900
3902
maintaining a set of tables, wherein each table includes motion candidates and each motion candidate is associated with corresponding motion information
3904
performing a conversion between a first video block and a bitstream representation of a video including the first video block
3906
updating one or multiple tables to include motion information from temporal neighbor block(s) of the first video block as a new motion candidate

FIG. 39

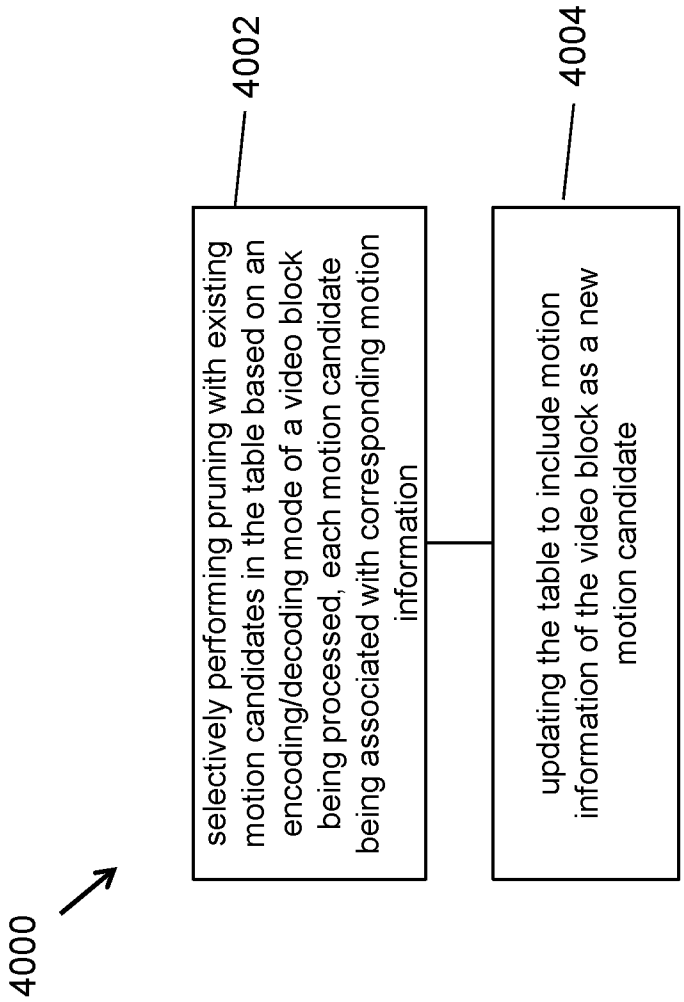

4000
4002
selectively performing pruning with existing motion candidates in the table based on an encoding/decoding mode of a video block being processed, each motion candidate being associated with corresponding motion information
4004
updating the table to include motion information of the video block as a new motion candidate

FIG. 40

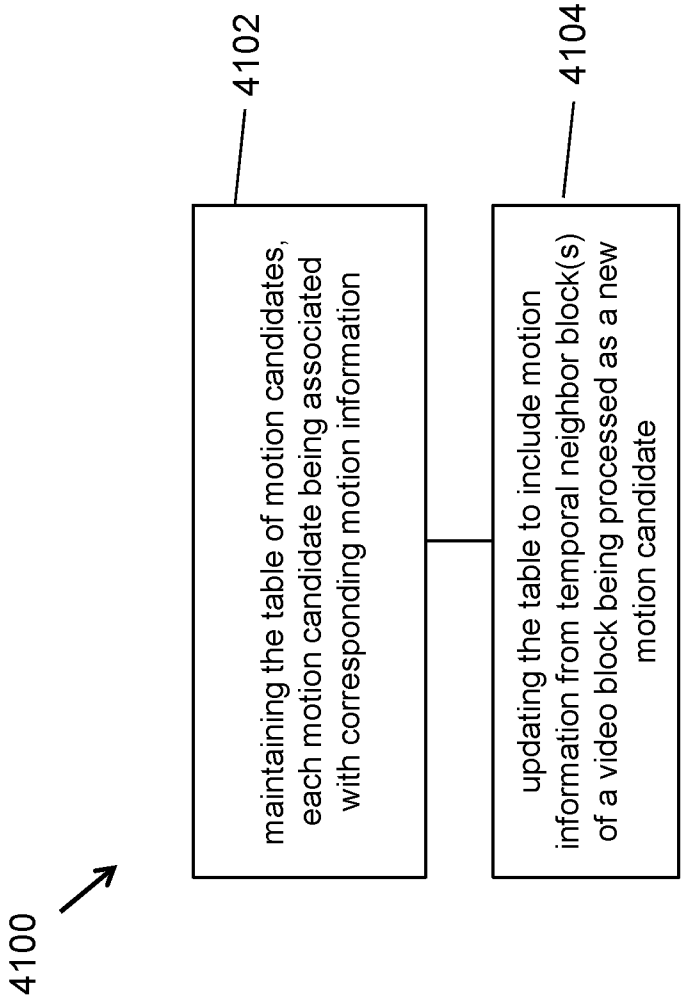

4100
4102
maintaining the table of motion candidates, each motion candidate being associated with corresponding motion information
4104
updating the table to include motion information from temporal neighbor block(s) of a video block being processed as a new motion candidate

FIG. 41

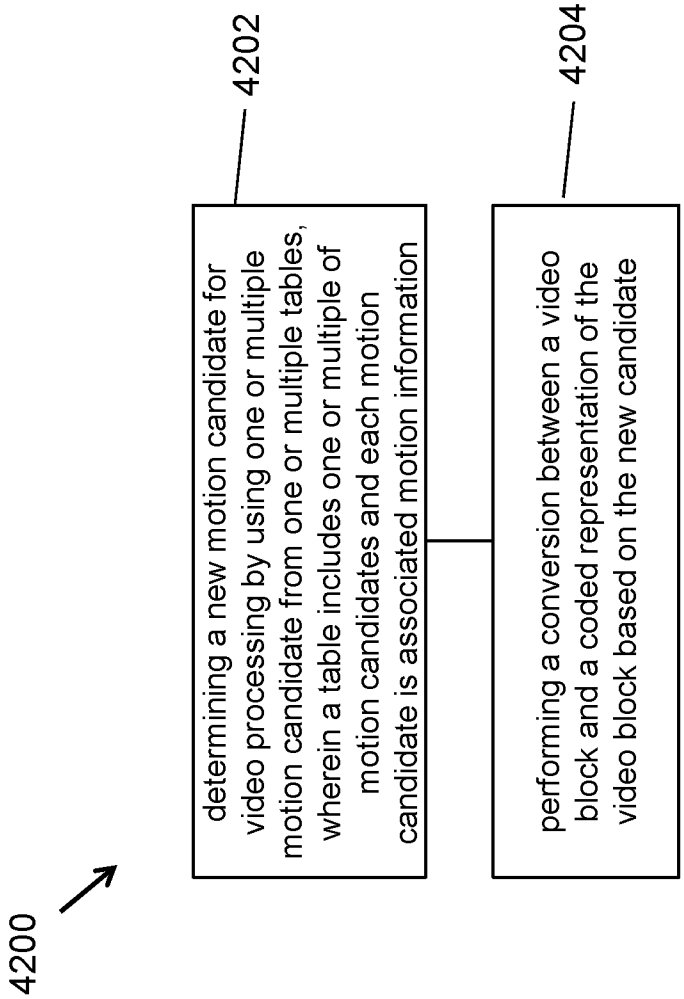

4200
4202
determining a new motion candidate for video processing by using one or multiple motion candidate from one or multiple tables, wherein a table includes one or multiple of motion candidates and each motion candidate is associated motion information
4204
performing a conversion between a video block and a coded representation of the video block based on the new candidate

FIG. 42