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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
23 July 2020 (23.07.2020)

WIPO | PCT

(10) International Publication Number
WO 2020/147747 A1

(51) International Patent Classification:
H04N 19/105 (2014.01) *H04N 19/573* (2014.01)

(21) International Application Number:
PCT/CN2020/072241

(22) International Filing Date:
15 January 2020 (15.01.2020)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
PCT/CN2019/071759
15 January 2019 (15.01.2019) CN

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

(72) Inventors: **LIU, Hongbin**; Jinritoutiao Post Office, China Satellite Communications Tower, No. 63, Zhichun Road, Haidian District, Beijing 100080 (CN). **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). **WANG, Yue**; Jinritoutiao Post Office, China Satellite Communications Tower, No. 63, Zhichun Road, Haidian District, Beijing 100080 (CN).

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

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

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH,

(54) Title: WEIGHTED PREDICTION IN VIDEO CODING

(57) Abstract: A video processing method includes generating, for a conversion between a current block of a video and a bitstream representation of the video, a motion vector for the current block; invoking a weighted prediction processing tool or a second processing tool in a mutually exclusive manner; and performing the conversion according to the invoking. The invoking of the second processing tool comprises refining the motion vector.

WO 2020/147747 A1

[Continued on next page]

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).

Declarations under Rule 4.17:

— *of inventorship (Rule 4.17(iv))*

Published:

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

WEIGHTED PREDICTION IN VIDEO CODING

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] Under the applicable patent law and/or rules pursuant to the Paris Convention, this application is made to timely claim the priority to and benefits of International Patent Application No. PCT/CN2019/071759, filed on January 15, 2019. For all purposes under the U.S. law, the entire disclosure of the aforementioned application is incorporated by reference as part of the disclosure of this application.

TECHNICAL FIELD

[0002] The present patent document relates to the field of video coding.

BACKGROUND

[0003] Currently, efforts are underway to improve the performance of current video codec technologies to provide better compression ratios or provide video coding and decoding schemes that allow for lower complexity or parallelized implementations. Industry experts have recently proposed several new video coding tools and tests are currently underway for determining their effectivity.

SUMMARY

[0004] The present document provides techniques for incorporating local illumination compensation in embodiments of video encoders or decoders.

[0005] In one example aspect, a method of video processing is disclosed. The method includes generating, for a conversion between a current block of a video and a bitstream representation of the video, a motion vector for the current block; invoking a weighted prediction processing tool or a second processing tool in a mutually exclusive manner; and performing the conversion according to the invoking. The invoking of the second processing tool comprises refining the motion vector.

[0006] In another example aspect, a method of video processing is disclosed. The method includes determining, in a conversion between a video block and a bitstream representation of the video block, that a video block is a boundary block of a coding tree unit (CTU) in which the video block is positioned and therefore local illumination compensation (LIC) coding tool is enabled for the video block, deriving, based on the determining that LIC coding tool is enabled for the video block, parameters for local illumination compensation (LIC) of the video block, and performing the conversion by adjusting pixel values of the video block using LIC.

[0007] In another example aspect, a method of video processing is disclosed. The method includes determining, in a conversion between a video block and a bitstream representation of the video block, that a video block is an inner block of a coding tree unit (CTU) in which the video block is positioned and therefore local illumination compensation (LIC) coding tool is disabled for the video block, inheriting parameters for LIC of the video block, and performing the conversion by adjusting pixel values of the video block using LIC.

[0008] In yet another example aspect, another method of video processing is disclosed. The method includes determining, in a conversion between a video block and a bitstream representation of the video block, that both local illumination compensation and intra block copy coding tools are enabled for use by the current block, and performing the conversion by performing local illumination compensation (LIC) and intra block copy operations on the video block.

[0009] In yet another example aspect, another method of video processing is disclosed. The method includes performing the conversion by performing local illumination compensation (LIC) and intra block copy operations on the video block, and performing the conversion between the current block and a corresponding bitstream representation of the current block.

[0010] In yet another example aspect, another method of video processing is disclosed. The method includes determining, during a conversion between a video block of a video and a bitstream representation of the video, local illumination compensation (LIC) parameters for the video block using at least some samples of neighboring blocks

of the video block, and performing the conversion between the video block and the bitstream representation by performing LIC using the determined parameters.

[0011] In yet another example aspect, another method of video processing is disclosed. The method includes performing a conversion between a video and a bitstream representation of the video, wherein the video is represented as video frames comprising video blocks, and local illumination compensation (LIC) is enabled only for video blocks that use a geometric prediction structure including a triangular prediction mode.

[0012] In yet another example aspect, another method of video processing is disclosed. The method includes performing a conversion between a video and a bitstream representation of the video, wherein the video is represented as video frames comprising video blocks, and local illumination compensation (LIC) is implemented for less than all pixels of a current block in the conversion to its corresponding bitstream representation.

[0013] In yet another example aspect, another method of video processing is disclosed. The method includes determining, in a conversion between a video block and a bitstream representation of the video block, that both local illumination compensation (LIC) and either generalized bi-prediction (GBi) or multi-hypothesis inter prediction coding tools are enabled for use by the current block, and performing the conversion by performing LIC and either GBi or multi-hypothesis inter prediction operations on the video block.

[0014] In yet another example aspect, another method of video processing is disclosed. The method includes determining, in a conversion between a video block and a bitstream representation of the video block, that both local illumination compensation (LIC) and combined inter-intra prediction (CIIP) coding tools are enabled for use by the current block, and performing the conversion by performing LIC and CIIP operations on the video block.

[0015] 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.

[0016] In yet another representative aspect, a video decoder apparatus may implement a method as described herein.

[0017] 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

[0018] FIG. 1 shows an example of a derivation process for merge candidates list construction.

[0019] FIG. 2 shows example positions of spatial merge candidates.

[0020] FIG. 3 shows examples of candidate pairs considered for redundancy check of spatial merge candidates.

[0021] FIG. 4 shows example Positions for the second PU of N×2N and 2N×N partitions.

[0022] FIG. 5 is an Illustration of motion vector scaling for temporal merge candidate.

[0023] FIG. 6 shows examples of candidate positions for temporal merge candidate, C0 and C1.

[0024] FIG. 7 shows an example of combined bi-predictive merge candidate

[0025] FIG. 8 shows an example of a derivation process for motion vector prediction candidates.

[0026] FIG. 9 is an example illustration of motion vector scaling for spatial motion vector candidate.

[0027] FIG. 10 illustrates an example of advanced temporal motion vector predictor (ATMVP) for a Coding Unit (CU).

[0028] FIG. 11 shows an Example of one CU with four sub-blocks (A-D) and its neighboring blocks (a–d).

[0029] FIG. 12 shows an example of a planar motion vector prediction process.

[0030] FIG. 13 is a flowchart of an example of encoding with different motion vector (MV) precision.

[0031] FIG. 14 is an example Illustration of sub-blocks where OBMC applies.

[0032] FIG. 15 shows an example of neighboring samples used for deriving IC parameters.

[0033] FIG. 16 is an illustration of splitting a coding unit (CU) into two triangular prediction units.

[0034] FIG. 17 shows an example of positions of neighboring blocks.

[0035] FIG. 18 shows an example in which a CU applies the 1st weighting factor group.

[0036] FIG. 19 shows an example of motion vector storage implementation.

[0037] FIG. 20 shows an example of a simplified affine motion model.

[0038] FIG. 21 shows an example of affine MVF per sub-block.

[0039] FIG. 22 shows examples of (a) 4-paramenter affine model (b) and 6-parameter affine model.

[0040] FIG. 23 shows an example of a Motion Vector Predictor (MV) for AF_INTER mode.

[0041] FIG. 24A-24B shows examples of candidates for AF_MERGE mode.

[0042] FIG. 25 shows candidate positions for affine merge mode.

[0043] FIG. 26 shows example process for bilateral matching.

[0044] FIG. 27 shows example process of template matching.

[0045] FIG. 28 illustrates an implementation of unilateral motion estimation (ME) in frame rate upconversion (FRUC).

[0046] FIG. 29 illustrates an embodiment of an Ultimate Motion Vector Expression (UMVE) search process.

[0047] FIG. 30 shows examples of UMVE search points.

[0048] FIG. 31 shows an example of distance index and distance offset mapping.

[0049] FIG. 32 shows an example of an optical flow trajectory.

[0050] FIG. 33A-33B show examples of Bi-directional Optical flow (BIO) w/o block extension: a) access positions outside of the block; b) padding used in order to avoid extra memory access and calculation.

[0051] FIG. 34 illustrates an example of using Decoder-side motion vector

refinement (DMVR) based on bilateral template matching.

[0052] FIG. 35 shows an example of neighboring samples used in a bilateral filter.

[0053] FIG. 36 shows an example of windows covering two samples used in weight calculation.

[0054] FIG. 37 shows an example of a decoding flow with the proposed history based motion vector prediction (HMVP) method.

[0055] FIG. 38 shows an example of updating the table in the proposed HMVP method.

[0056] FIG. 39 is a block diagram of a hardware platform for implementing the video coding or decoding techniques described in the present document.

[0057] FIG. 40 shows an example of a hardware platform for implementing methods and techniques described in the present document.

[0058] FIG. 41 is a flowchart of an example method of video processing.

[0059] FIG. 42 is a flowchart of an example method of video processing in accordance with the disclosure.

DETAILED DESCRIPTION

[0060] The present document provides several techniques that can be embodied into digital video encoders and decoders. Section headings are used in the present document for clarity of understanding and do not limit scope of the techniques and embodiments disclosed in each section only to that section.

[0061] **1. Summary**

[0062] This patent document is related to video coding technologies. Specifically, it is related to local illumination compensation (LIC) 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.

[0063] **2. Examples of video coding/decoding technologies**

[0064] 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. 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). In April 2018, the Joint Video Expert Team (JVET) between VCEG (Q6/16) and ISO/IEC JTC1 SC29/WG11 (MPEG) was created to work on the VVC standard targeting at 50% bitrate reduction compared to HEVC.

[0065] 2.1. Inter prediction in HEVC/H.265

[0066] 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.

[0067] When a CU is coded with skip mode, one PU is associated with the CU, and there are no significant residual coefficients, no coded motion vector delta or reference picture index. A merge mode is specified whereby the motion parameters for the current PU are obtained from neighboring PUs, including spatial and temporal candidates. The merge mode can be applied to any inter-predicted PU, not only for skip mode. The alternative to merge mode is the explicit transmission of motion parameters, where motion vector (to be more precise, motion vector difference compared to a motion vector predictor), corresponding reference picture index for each reference picture list and reference picture list usage are signalled explicitly per each PU. Such a mode is named Advanced motion vector prediction (AMVP) in this disclosure.

[0068] 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.

[0069] 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.

[0070] The following text provides the details on the inter prediction modes specified in HEVC. The description will start with the merge mode.

[0071] **2.1.1. Merge Mode**

[0072] **2.1.1.1. Derivation of candidates for merge mode**

[0073] 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

[0074] These steps are also schematically depicted in FIG. 1. 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 obtained from step 1 does not reach the 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.

[0075] In the following, the operations associated with the aforementioned steps are

detailed.

[0076] 2.1.1.2. Spatial candidates derivation

[0077] In the derivation of spatial merge candidates, a maximum of four merge candidates are selected among candidates located in the positions depicted in FIG. 2. 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. FIG. 3 shows examples of candidate pairs considered for redundancy check of spatial merge candidates. 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. 3 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. 4 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.

[0078] 2.1.1.3. Temporal candidates derivation

[0079] 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. FIG. 5 is an Illustration of motion vector scaling for temporal merge candidate. The scaled motion vector for temporal merge candidate is obtained as illustrated by the dotted line in FIG. 5, 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.

[0080] 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. 6. If PU at position C_0 is not available, is intra coded, or is outside of the current CTU row, position C_1 is used. Otherwise, position C_0 is used in the derivation of the temporal merge candidate.

[0081] **2.1.1.4. Additional candidates insertion**

[0082] Besides spatial and 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 spatial and 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. 7 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.

[0083] 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.

[0084] 2.1.1.5. Motion estimation regions for parallel processing

[0085] 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.

[0086] 2.1.2. AMVP

[0087] AMVP exploits spatio-temporal correlation of motion vector with neighboring PUs, which is used for explicit transmission of motion parameters. For each reference picture list, a motion vector candidate list is constructed by firstly checking availability of left, above temporally neighboring PU positions, removing redundant candidates and adding zero vector to make the candidate list to be constant length. Then, the encoder can select the best predictor from the candidate list and transmit the corresponding index indicating the chosen candidate. Similarly with merge index 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 (see FIG. 8). In the following sections, details about derivation process of motion vector prediction candidate are provided.

[0088] 2.1.2.1. Derivation of AMVP candidates

[0089] 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. 8.

[0090] 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.

[0091] **2.1.2.2. Spatial motion vector candidates**

[0092] 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. 2, 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)

[0093] 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 neighboring 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.

[0094] In a spatial scaling process, the motion vector of the neighboring PU is scaled in a similar manner as for temporal scaling, as depicted as FIG. 9. 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.

[0095] **2.1.2.3. Temporal motion vector candidates**

[0096] 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 FIG. 6). The reference picture index is signalled to the decoder.

[0097] **2.2. New inter prediction methods in JEM**

[0098] **2.2.1. Sub-CU based motion vector prediction**

[0099] 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 neighboring motion vector.

[00100] To preserve more accurate motion field for sub-CU motion prediction, the motion compression for the reference frames is currently disabled.

[00101] **2.2.1.1. Alternative temporal motion vector prediction**

[00102] 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. 10, the sub-CUs are square N×N blocks (N is set to 4 by default).

[00103] 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. 10.

[00104] In the first step, a reference picture and the corresponding block is determined by the motion information of the spatial neighboring blocks of the current CU. To avoid the repetitive scanning process of neighboring 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.

[00105] 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.

[00106] 2.2.1.2. Spatial-temporal motion vector prediction

[00107] In this method, the motion vectors of the sub-CUs are derived recursively, following raster scan order. FIG. 11 illustrates this concept. Let us consider an 8×8 CU which contains four 4×4 sub-CUs A, B, C, and D. The neighboring 4×4 blocks in the current frame are labelled as a, b, c, and d.

[00108] 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 neighboring blocks for each list is scaled to the first reference frame for a given list. Next, temporal motion vector predictor (TMVP) of sub-block A 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.

[00109] 2.2.1.3. Sub-CU motion prediction mode signalling

[00110] 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.

[00111] 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.

[00112] **2.2.2. Pairwise average candidates**

[00113] Pairwise average candidates are generated by averaging predefined pairs of candidates in the current merge candidate list, and the predefined pairs are defined as {(0, 1), (0, 2), (1, 2), (0, 3), (1, 3), (2, 3)}, where the numbers denote the merge indices to the merge candidate list. The averaged motion vectors are calculated separately for each reference list. If both motion vectors are available in one list, these two motion vectors are averaged even when they point to different reference pictures; if only one motion vector is available, use the one directly; if no motion vector is available, keep this list invalid. The pairwise average candidates replace the combined candidates in HEVC standard.

[00114] The complexity analysis of pairwise average candidates is summarized in the Table 1. For the worst case of additional calculations for averaging (the last column in Table 1), 4 additions and 4 shifts are needed for each pair (MVx and MVy in L0 and L1), and 4 reference index comparisons are needed for each pair (refIdx0 is valid and refIdx1 is valid in L0 and L1). There are 6 pairs, leading to 24 additions, 24 shifts, and 24 reference index comparisons in total. The combined candidates in HEVC standard use 2 reference index comparisons for each pair (refIdx0 is valid in L0 and refIdx1 is valid in L1), and there are 12 pairs, leading to 24 reference index comparisons in total.

Table 1: Operation analysis for the pairwise average candidates

Merge list size	Max number of potential candidates	Max number of candidate comparisons	Max number of MV scalings	Max number of temporal candidates	Additional local buffer	Max number of memory access	Others
6, 8, 10	6	0	0	0	0	0	Replace HEVC combined candidates, need

							additional calculations for averaging

[00115] 2.2.3. Planar Motion Vector Prediction

[00116] In JVET-K0135, planar motion vector prediction is proposed.

[00117] To generate a smooth fine granularity motion field, FIG. 12 gives a brief description of the planar motion vector prediction process.

[00118] Planar motion vector prediction is achieved by averaging a horizontal and vertical linear interpolation on 4x4 block basis as follows.

[00119] $\mathrm{P}(\mathrm{x},\mathrm{y}) = (H \times P_h(x,y) + W \times P_v(x,y) + H \times W)/(2 \times H \times W)$ Eq. (1)

[00120] *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.

[00121] The horizontal prediction $P_h(x,y)$ and the vertical prediction $P_v(x,y)$ for location *(x,y)* are calculated as follows:

[00122] $P_h(\mathrm{x},\mathrm{y}) = (W-1-x) \times L(-1,y) + (x+1) \times R(W,y)$ Eq. (2)

[00123] $P_v(\mathrm{x},\mathrm{y}) = (H-1-y) \times A(x,-1) + (y+1) \times B(x,H)$ Eq. (3)

[00124] 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.

[00125] The reference motion information of the left column and above row neighbour blocks are derived from the spatial neighbour blocks of current block.

[00126] The reference motion information of the right column and bottom row neighbour blocks are derived as follows.

[00127] (1) Derive the motion information of the bottom right temporal neighbour *4x4* block

[00128] (2) 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 Eq. (4) below.

[00129] (3) 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 Eq. (5)**Error! Reference source not found.**.

[00130] $$\mathrm{R(W,y)} = ((\mathrm{H} - \mathrm{y} - 1) \times \mathrm{AR} + (\mathrm{y} + 1) \times \mathrm{BR})/\mathrm{H} \quad \text{Eq. (4)}$$

[00131] $$\mathrm{B(x,H)} = ((W - x - 1) \times BL + (x + 1) \times BR)/W \quad \text{Eq. (5)}$$

[00132] 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.

[00133] The motion information obtained from the neighboring blocks for each list is scaled to the first reference picture for a given list.

[00134] **2.2.4. Adaptive motion vector different resolution**

[00135] 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.

[00136] 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.

[00137] 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.

[00138] 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.

- 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.
- 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.

[00139] The encoding process is shown in FIG. 13. First, 1/4 pel MV is tested and the RD cost is calculated and denoted as *RDCost0*, then integer MV is tested and the RD cost is denoted as *RDCost1*. If $RDCost1 < th * RDCost0$ (wherein *th* is a positive value), then 4-pel MV is tested; otherwise, 4-pel MV is skipped. Basically, motion information and RD cost etc. are already known for 1/4 pel MV when checking integer or 4-pel MV, which can be reused to speed up the encoding process of integer or 4-pel MV.

[00140] **2.2.5. Higher motion vector storage accuracy**

[00141] In HEVC, motion vector accuracy is one-quarter pel (one-quarter luma sample and one-eighth chroma sample for 4:2:0 video). In the JEM, the accuracy for the internal motion vector storage and the merge candidate increases to 1/16 pel. The higher motion vector accuracy (1/16 pel) is used in motion compensation inter prediction for the CU coded with skip/merge mode. For the CU coded with normal AMVP mode, either the integer-pel or quarter-pel motion is used, as described in section 2.2.2.

[00142] SHVC upsampling interpolation filters, which have same filter length and normalization factor as HEVC motion compensation interpolation filters, are used as motion compensation interpolation filters for the additional fractional pel positions. The

chroma component motion vector accuracy is 1/32 sample in the JEM, the additional interpolation filters of 1/32 pel fractional positions are derived by using the average of the filters of the two neighboring 1/16 pel fractional positions.

[00143] 2.2.6. Overlapped block motion compensation

[00144] Overlapped Block Motion Compensation (OBMC) has previously been used in H.263. In the JEM, unlike in H.263, OBMC can be switched on and off using syntax at the CU level. When OBMC is used in the JEM, the OBMC is performed for all motion compensation (MC) block boundaries except the right and bottom boundaries of a CU. Moreover, it is applied for both the luma and chroma components. In the JEM, a MC block is corresponding to a coding block. When a CU is coded with sub-CU mode (includes sub-CU merge, affine and FRUC mode), each sub-block of the CU is a MC block. To process CU boundaries in a uniform fashion, OBMC is performed at sub-block level for all MC block boundaries, where sub-block size is set equal to 4×4, as illustrated in FIG. 14.

[00145] When OBMC applies to the current sub-block, besides current motion vectors, motion vectors of four connected neighboring sub-blocks, if available and are not identical to the current motion vector, are also used to derive prediction block for the current sub-block. These multiple prediction blocks based on multiple motion vectors are combined to generate the final prediction signal of the current sub-block.

[00146] Prediction block based on motion vectors of a neighboring sub-block is denoted as P_N, with N indicating an index for the neighboring *above*, *below*, *left* and *right* sub-blocks and prediction block based on motion vectors of the current sub-block is denoted as P_C. When P_N is based on the motion information of a neighboring sub-block that contains the same motion information to the current sub-block, the OBMC is not performed from P_N. Otherwise, every sample of P_N is added to the same sample in P_C, i.e., four rows/columns of P_N are added to P_C. The weighting factors {1/4, 1/8, 1/16, 1/32} are used for P_N and the weighting factors {3/4, 7/8, 15/16, 31/32} are used for P_C. The exception are small MC blocks, (i.e., when height or width of the coding block is equal to 4 or a CU is coded with sub-CU mode), for which only two rows/columns of P_N are added to P_C. In this case weighting factors {1/4, 1/8} are used for P_N and weighting

factors {3/4, 7/8} are used for P_C. For P_N generated based on motion vectors of vertically (horizontally) neighboring sub-block, samples in the same row (column) of P_N are added to P_C with a same weighting factor.

[00147] In the JEM, for a CU with size less than or equal to 256 luma samples, a CU level flag is signalled to indicate whether OBMC is applied or not for the current CU. For the CUs with size larger than 256 luma samples or not coded with AMVP mode, OBMC is applied by default. At the encoder, when OBMC is applied for a CU, its impact is taken into account during the motion estimation stage. The prediction signal formed by OBMC using motion information of the top neighboring block and the left neighboring block is used to compensate the top and left boundaries of the original signal of the current CU, and then the normal motion estimation process is applied.

[00148] **2.2.7. Local illumination compensation**

[00149] Local Illumination Compensation (LIC) 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).

[00150] When LIC applies for a CU, a least square error method is employed to derive the parameters *a* and *b* by using the neighboring samples of the current CU and their corresponding reference samples. More specifically, as illustrated in FIG. 15, the subsampled (2:1 subsampling) neighboring 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.

[00151] When a CU is coded with merge mode, the LIC flag is copied from neighboring blocks, in a way similar to motion information copy in merge mode; otherwise, an LIC flag is signalled for the CU to indicate whether LIC applies or not.

[00152] When LIC is enabled for a picture, addtional CU level RD check is needed to determine whether LIC is applied or not for a CU. When LIC 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.

[00153] To reduce the encoding complexity, the following encoding scheme is applied in the JEM: LIC 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, LIC is disabled for the current picture; otherwise, LIC is enabled for the current picture.

[00154] **2.2.8. Hybrid intra and inter prediction**

[00155] In JVET-L0100, multi-hypothesis prediction is proposed, wherein hybrid intra and inter prediction is one way to generate multiple hypotheses.

[00156] When the multi-hypothesis prediction is applied to improve intra mode, multi-hypothesis prediction combines one intra prediction and one merge indexed prediction. In a merge CU, one flag is signaled for merge mode to select an intra mode from an intra candidate list when the flag is true. For luma component, the intra candidate list is derived from 4 intra prediction modes including DC, planar, horizontal, and vertical modes, and the size of the intra candidate list can be 3 or 4 depending on the block shape. When the CU width is larger than the double of CU height, horizontal mode is exclusive of the intra mode list and when the CU height is larger than the double of CU width, vertical mode is removed from the intra mode list. One intra prediction mode selected by the intra mode index and one merge indexed prediction selected by the merge index are combined using weighted average. For chroma component, DM is always applied without extra signaling. The weights for combining predictions are described as follow. When DC or planar mode is selected, or the CB width or height is smaller than 4, equal weights are applied. For those CBs with CB width and height larger than or equal to 4, when horizontal/vertical mode is selected, one CB is first vertically/horizontally split into four equal-area regions. Each weight set, denoted as (w_intra_i, w_inter_i), where i is from 1 to 4 and (w_intra_1, w_inter_1) = (6, 2), (w_intra_2, w_inter_2) = (5, 3), (w_intra_3, w_inter_3) = (3, 5), and (w_intra_4, w_inter_4) = (2, 6), will be applied to a corresponding region. (w_intra_1, w_inter_1) is for the region closest to the reference samples and (w_intra_4, w_inter_4) is for the region farthest away from the

reference samples. Then, the combined prediction can be calculated by summing up the two weighted predictions and right-shifting 3 bits. Moreover, the intra prediction mode for the intra hypothesis of predictors can be saved for reference of the following neighboring CUs.

[00157] 2.2.9. Triangular prediction unit mode

[00158] The concept of the triangular prediction unit mode is to introduce a new triangular partition for motion compensated prediction. As shown in FIG. 16, it splits a CU into two triangular prediction units (PUs), in either diagonal or inverse diagonal direction. Each triangular prediction unit in the CU is inter-predicted using its own uni-prediction motion vector and reference frame index which are derived from a uni-prediction candidate list. An adaptive weighting process is performed to the diagonal edge after predicting the triangular prediction units. Then, the transform and quantization process are applied to the whole CU. It is noted that this mode is only applied to skip and merge modes.

[00159] Uni-prediction candidate list

[00160] The uni-prediction candidate list consists of five uni-prediction motion vector candidates. It is derived from seven neighboring blocks including five spatial neighboring blocks (1 to 5) and two temporal co-located blocks (6 to 7), as shown in FIG. 17. The motion vectors of the seven neighboring blocks are collected and put into the uni-prediction candidate list according in the order of uni-prediction motion vectors, L0 motion vector of bi-prediction motion vectors, L1 motion vector of bi-prediction motion vectors, and averaged motion vector of the L0 and L1 motion vectors of bi-prediction motion vectors. If the number of candidates is less than five, zero motion vector is added to the list.

[00161] Adaptive weighting process

[00162] After predicting each triangular prediction unit, an adaptive weighting process is applied to the diagonal edge between the two triangular prediction units to derive the final prediction for the whole CU. Two weighting factor groups are listed as follows:

- 1st weighting factor group: {7/8, 6/8, 4/8, 2/8, 1/8} and {7/8, 4/8, 1/8} are used for the luminance and the chrominance samples, respectively;

- 2nd weighting factor group: {7/8, 6/8, 5/8, 4/8, 3/8, 2/8, 1/8} and {6/8, 4/8, 2/8} are used for the luminance and the chrominance samples, respectively.

[00163] One weighting factor group is selected based on the comparison of the motion vectors of two triangular prediction units. The 2nd weighting factor group is used when the reference pictures of the two triangular prediction units are different from each other or their motion vector difference is larger than 16 pixels. Otherwise, the 1st weighting factor group is used. An example is shown in FIG. 18.

[00164] **Motion vector storage**

[00165] The motion vectors (Mv1 and Mv2 in FIG. 19) of the triangular prediction units are stored in 4×4 grids. For each 4×4 grid, either uni-prediction or bi-prediction motion vector is stored depending on the position of the 4×4 grid in the CU. As shown in FIG. 19, uni-prediction motion vector, either Mv1 or Mv2, is stored for the 4×4 grid located in the non-weighted area. On the other hand, a bi-prediction motion vector is stored for the 4×4 grid located in the weighted area. The bi-prediction motion vector is derived from Mv1 and Mv2 according to the following rules:

[00166] 1. In the case that Mv1 and Mv2 have motion vector from different directions (L0 or L1), Mv1 and Mv2 are simply combined to form the bi-prediction motion vector.

[00167] 2. In the case that both Mv1 and Mv2 are from the same L0 (or L1) direction,

[00168] 2.a. If the reference picture of Mv2 is the same as a picture in the L1 (or L0) reference picture list, Mv2 is scaled to the picture. Mv1 and the scaled Mv2 are combined to form the bi-prediction motion vector.

[00169] 2.b. If the reference picture of Mv1 is the same as a picture in the L1 (or L0) reference picture list, Mv1 is scaled to the picture. The scaled Mv1 and Mv2 are combined to form the bi-prediction motion vector.

[00170] 2.c. Otherwise, only Mv1 is stored for the weighted area.

[00171] **2.2.10. Affine motion compensation prediction**

[00172] In HEVC, only translation motion model is applied for motion compensation prediction (MCP). While in the real world, there are many kinds of motion, e.g. zoom in/out, rotation, perspective motions and he other irregular motions. In the JEM, a simplified affine transform motion compensation prediction is applied. As shown in

FIG. 20, the affine motion field of the block is described by two control point motion vectors.

[00173] The motion vector field (MVF) of a block is described by the following equation:

[00174]
$$\begin{cases} v_x = \frac{(v_{1x}-v_{0x})}{w}x - \frac{(v_{1y}-v_{0y})}{w}y + v_{0x} \\ v_y = \frac{(v_{1y}-v_{0y})}{w}x + \frac{(v_{1x}-v_{0x})}{w}y + v_{0y} \end{cases} \quad \text{Eq. (6)}$$

[00175] Where (v_{0x}, v_{0y}) is motion vector of the top-left corner control point, and (v_{1x}, v_{1y}) is motion vector of the top-right corner control point.

[00176] In order to further simplify the motion compensation prediction, sub-block based affine transform prediction is applied. The sub-block size $M \times N$ is derived as in Eq. (7), where *MvPre* is the motion vector fraction accuracy (1/16 in JEM), (v_{2x}, v_{2y}) is motion vector of the bottom-left control point, calculated according to Eq. (6).

[00177]
$$\begin{cases} M = clip3\left(4, w, \frac{w \times MvPre}{\max\left(abs(v_{1x}-v_{0x}), abs(v_{1y}-v_{0y})\right)}\right) \\ N = clip3\left(4, h, \frac{h \times MvPre}{\max\left(abs(v_{2x}-v_{0x}), abs(v_{2y}-v_{0y})\right)}\right) \end{cases} \quad \text{Eq. (7)}$$

[00178] After derived by Eq. (7), M and N should be adjusted downward if necessary to make it a divisor of w and h, respectively.

[00179] To derive motion vector of each M×N sub-block, the motion vector of the center sample of each sub-block, as shown in FIG. 21, is calculated according to Equation **Error! Reference source not found.**, and rounded to 1/16 fraction accuracy. Then the motion compensation interpolation filters mentioned in section [00140] are applied to generate the prediction of each sub-block with derived motion vector.

[00180] After MCP, the high accuracy motion vector of each sub-block is rounded and saved as the same accuracy as the normal motion vector.

[00181] **2.2.10.1. AF_INTER mode**

[00182] In the JEM, there are two affine motion modes: AF_INTER mode and AF_MERGE mode. For CUs with both width and height larger than 8, AF_INTER mode can be applied. An affine flag in CU level is signalled in the bitstream to indicate whether AF_INTER mode is used. In this mode, a candidate list with motion vector pair $\{(v_0, v_1)|v_0 = \{v_A, v_B, v_c\}, v_1 = \{v_D, v_E\}\}$ is constructed using the neighbour blocks. As

shown in FIG. 23, v_0 is selected from the motion vectors of the block A, B or C. The motion vector from the neighbour block is scaled according to the reference list and the relationship among the POC of the reference for the neighbour block, the POC of the reference for the current CU and the POC of the current CU. And the approach to select v_1 from the neighbour block D and E is similar. If the number of candidate list is smaller than 2, the list is padded by the motion vector pair composed by duplicating each of the AMVP candidates. When the candidate list is larger than 2, the candidates are firstly sorted according to the consistency of the neighboring motion vectors (similarity of the two motion vectors in a pair candidate) and only the first two candidates are kept. An RD cost check is used to determine which motion vector pair candidate is selected as the control point motion vector prediction (CPMVP) of the current CU. And an index indicating the position of the CPMVP in the candidate list is signalled in the bitstream. After the CPMVP of the current affine CU is determined, affine motion estimation is applied and the control point motion vector (CPMV) is found. Then the difference of the CPMV and the CPMVP is signalled in the bitstream.

[00183] In AF_INTER mode, when 4/6 parameter affine mode is used, 2/3 control points are required, and therefore 2/3 MVD needs to be coded for these control points, as shown in FIG. 22. In JVET-K0337, it is proposed to derive the MV as follows, i.e., mvd_1 and mvd_2 are predicted from mvd_0.

[00184] $$\boldsymbol{mv}_0 = \overline{\boldsymbol{mv}}_0 + \boldsymbol{mvd}_0 \quad \text{Eq. (8)}$$

[00185] $$\boldsymbol{mv}_1 = \overline{\boldsymbol{mv}}_1 + \boldsymbol{mvd}_1 + \boldsymbol{mvd}_0 \quad \text{Eq. (9)}$$

[00186] $$\boldsymbol{mv}_2 = \overline{\boldsymbol{mv}}_2 + \boldsymbol{mvd}_2 + \boldsymbol{mvd}_0 \quad \text{Eq. (10)}$$

[00187] Wherein $\overline{\boldsymbol{mv}}_i$, mvd_i and mv_1 are the predicted motion vector, motion vector difference and motion vector of the top-left pixel (i = 0), top-right pixel (i = 1) or left-bottom pixel (i = 2) respectively, as shown in FIG. 22. Please note that the addition of two motion vectors (e.g., mvA(xA, yA) and mvB(xB, yB)) is equal to summation of two components separately, that is, newMV = mvA + mvB and the two components of newMV is set to (xA + xB) and (yA + yB), respectively.

[00188] **2.2.10.2. Fast affine ME algorithm in AF_INTER mode**

[00189] In affine mode, MV of 2 or 3 control points needs to be determined jointly.

Directly searching the multiple MVs jointly is computationally complex. A fast affine ME algorithm is proposed and is adopted into VTM/BMS.

[00190] The fast affine ME algorithm is described for the 4-parameter affine model, and the idea can be extended to 6-parameter affine model.

[00191] $$\begin{cases} x' = ax + by + c \\ y' = -bx + ay + d \end{cases} \quad \text{Eq. (11)}$$

[00192] $$\begin{cases} mv^h_{(x,y)} = x' - x = (a-1)x + by + c \\ mv^v_{(x,y)} = y' - y = -bx + (a-1)y + d \end{cases} \quad \text{Eq. (12)}$$

[00193] Replace (a – 1) with a', then the motion vector can be rewritten as:

[00194] $$\begin{cases} mv^h_{(x,y)} = x' - x = a'x + by + c \\ mv^v_{(x,y)} = y' - y = -bx + a'y + d \end{cases} \quad \text{Eq. (13)}$$

[00195] Suppose motion vectors of the two controls points (0, 0) and (0, w) are known, from Eq. (13) we can derive affine parameters:

[00196] $$\begin{cases} c = mv^h_{(0,0)} \\ d = mv^v_{(0,0)} \end{cases} \quad \text{Eq. (14)}$$

[00197] The motion vectors can be rewritten in vector form as:

[00198] $$MV(p) = A(P) * MV_C^T \quad \text{Eq. (15)}$$

[00199] Wherein

[00200] $$A(P) = \begin{bmatrix} 1 & x & 0 & y \\ 0 & y & 1 & -x \end{bmatrix} \quad \text{Eq. (16)}$$

[00201] $$MV_C = \begin{bmatrix} mv^h_{(0,0)} & a & mv^v_{(0,0)} & b \end{bmatrix} \quad \text{Eq. (17)}$$

[00202] P = (x, y) is the pixel position.

[00203] At encoder, MVD of AF_INTER are derived iteratively. Denote $MV^i(P)$ as the MV derived in the ith iteration for position P and denote dMV_C^i as the delta updated for MV_C in the ith iteration. Then in the (i+1)th iteration:

[00204] $$\begin{aligned} MV^{i+1}(P) &= A(P) * \left(\left(MV_C^i\right)^T + \left(dMV_C^i\right)^T \right) \\ &= A(P) * \left(MV_C^i\right)^T + A(P) * \left(dMV_C^i\right)^T \\ &= MV^i(P) + A(P) * \left(dMV_C^i\right)^T \end{aligned} \quad \text{Eq. (18)}$$

[00205] Denote Pic_{ref} as the reference picture and denote Pic_{cur} as the current picture and denote $Q = P + MV^i(P)$. Suppose we use MSE as the matching criterion, then we

need to minimize:

[00206] $min \sum_P \left(Pic_{cur}(P) - Pic_{ref}\left(P + MV^{i+1}(P)\right)\right)^2$

[00207] $= min \sum_P \left(Pic_{cur}(P) - Pic_{ref}\left(Q + A(P) * \left(dMV_C^i\right)^T\right)\right)^2$ Eq. (19)

[00208] Suppose $\left(dMV_C^i\right)^T$ is small enough, we can rewrite $Pic_{ref}\left(Q + A(P) * \left(dMV_C^i\right)^T\right)$ approximately as follows with 1th order Taylor expansion.

[00209] $Pic_{ref}\left(Q + A(P) * \left(dMV_C^i\right)^T\right) \approx$

[00210] $Pic_{ref}(Q) + Pic_{ref}{}'(Q) * A(P) * \left(dMV_C^i\right)^T$ Eq. (20)

[00211] Wherein $Pic_{ref}{}'(Q) = \left[\frac{dPic_{ref}(Q)}{dx} \ \frac{dPic_{ref}(Q)}{dy}\right]$. Denote $E^{i+1}(P) = Pic_{cur}(P) - Pic_{ref}(Q)$,

[00212] $min \sum_P \left(Pic_{cur}(P) - Pic_{ref}(Q) - Pic_{ref}{}'(Q) * A(P) * \left(dMV_C^i\right)^T\right)^2 =$

[00213] $min \sum_P \left(E^{i+1}(P) - Pic_{ref}{}'(Q) * A(P) * \left(dMV_C^i\right)^T\right)^2$ Eq. (21)

[00214] We can derive dMV_C^i by setting the derivative of the error function to zero. Then we can then calculate delta MV of the control points (0, 0) and (0, w) according to $A(P) * \left(dMV_C^i\right)^T$,

[00215] $dMV_{(0,0)}^h = dMV_C^i[0]$ Eq. (22)

[00216] $dMV_{(0,w)}^h = dMV_C^i[1] * w + dMV_C^i[2]$ Eq. (23)

[00217] $dMV_{(0,0)}^v = dMV_C^i[2]$ Eq. (24)

[00218] $dMV_{(0,w)}^v = -dMV_C^i[3] * w + dMV_C^i[2]$ Eq. (25)

[00219] Suppose such MVD derivation process is iterated by n times, then the final MVD is calculated as follows,

[00220] $fdMV_{(0,0)}^h = \sum_{i=0}^{n-1} dMV_C^i[0]$ Eq. (26)

[00221] $fdMV_{(0,w)}^h = \sum_{i=0}^{n-1} dMV_C^i[1] * w + \sum_{i=0}^{n-1} dMV_C^i[0]$ Eq. (27)

[00222] $fdMV_{(0,0)}^v = \sum_{i=0}^{n-1} dMV_C^i[2]$ Eq. (28)

[00223] $fdMV_{(0,w)}^v = \sum_{i=0}^{n-1} -dMV_C^i[3] * w + \sum_{i=0}^{n-1} dMV_C^i[2]$ Eq. (29)

[00224] With JVET-K0337, i.e., predicting delta MV of control point (0, w), denoted by mvd_1 from delta MV of control point (0, 0), denoted by mvd_0, now actually only $\left(\sum_{i=0}^{n-1} dMV_C^i[1] * w, -\sum_{i=0}^{n-1} -dMV_C^i[3] * w\right)$ is encoded formvd_1.

[00225] **2.2.10.3. AF_MERGE mode**

[00226] When a CU is applied in AF_MERGE mode, it gets the first block coded with affine mode from the valid neighbour reconstructed blocks. And the selection order for the candidate block is from left, above, above right, left bottom to above left as shown in FIG. 24A, the motion vectors v_2 , v_3 and v_4 of the top left corner, above right corner and left bottom corner of the CU which contains the block A are derived. And the motion vector v_0 of the top left corner on the current CU is calculated according to v_2 , v_3 and v_4. Secondly, the motion vector v_1 of the above right of the current CU is calculated.

[00227] After the CPMV of the current CU v_0 and v_1 are derived, according to the simplified affine motion model Equation **Error! Reference source not found.**, the MVF of the current CU is generated. In order to identify whether the current CU is coded with AF_MERGE mode, an affine flag is signalled in the bitstream when there is at least one neighbour block is coded in affine mode.

[00228] FIG. 24A and 24B show examples of candidates for AF_MERGE

[00229] In JVET-L0366, which was planned to be adopted into VTM 3.0, an affine merge candidate list is constructed with following steps:

[00230] (1) Insert inherited affine candidates

[00231] Inherited affine candidate means that the candidate is derived from the affine motion model of its valid neighbor affine coded block. In the common base, as shown in FIG. 25, the scan order for the candidate positions is: A1, B1, B0, A0 and B2.

[00232] After a candidate is derived, full pruning process is performed to check whether same candidate has been inserted into the list. If a same candidate exists, the derived candidate is discarded.

[00233] (2) Insert constructed affine candidates

[00234] If the number of candidates in affine merge candidate list is less than MaxNumAffineCand (set to 5 in this contribution), constructed affine candidates are inserted into the candidate list. Constructed affine candidate means the candidate is

constructed by combining the neighbor motion information of each control point.

[00235] The motion information for the control points is derived firstly from the specified spatial neighbors and temporal neighbor shown in FIG. 25. CPk (k=1, 2, 3, 4) represents the k-th control point. A0, A1, A2, B0, B1, B2 and B3 are spatial positions for predicting CPk (k=1, 2, 3); T is temporal position for predicting CP4.

[00236] The coordinates of CP1, CP2, CP3 and CP4 is (0, 0), (W, 0), (H, 0) and (W, H), respectively, where W and H are the width and height of current block.

[00237] The motion information of each control point is obtained according to the following priority order:

[00238] For CP1, the checking priority is B2->B3->A2. B2 is used if it is available. Otherwise, if B2 is available, B3 is used. If both B2 and B3 are unavailable, A2 is used. If all the three candidates are unavailable, the motion information of CP1 cannot be obtained.

[00239] For CP2, the checking priority is B1->B0.

[00240] For CP3, the checking priority is A1->A0.

[00241] For CP4, T is used.

[00242] Secondly, the combinations of controls points are used to construct an affine merge candidate.

[00243] Motion information of three control points are needed to construct a 6-parameter affine candidate. The three control points can be selected from one of the following four combinations ({CP1, CP2, CP4}, {CP1, CP2, CP3}, {CP2, CP3, CP4}, {CP1, CP3, CP4}). Combinations {CP1, CP2, CP3}, {CP2, CP3, CP4}, {CP1, CP3, CP4} will be converted to a 6-parameter motion model represented by top-left, top-right and bottom-left control points.

[00244] Motion information of two control points are needed to construct a 4-parameter affine candidate. The two control points can be selected from one of the following six combinations ({CP1, CP4}, {CP2, CP3}, {CP1, CP2}, {CP2, CP4}, {CP1, CP3}, {CP3, CP4}). Combinations {CP1, CP4}, {CP2, CP3}, {CP2, CP4}, {CP1, CP3}, {CP3, CP4} will be converted to a 4-parameter motion model represented by top-left and top-right control points.

[00245] The combinations of constructed affine candidates are inserted into to candidate list as following order:
{CP1, CP2, CP3}, {CP1, CP2, CP4}, {CP1, CP3, CP4}, {CP2, CP3, CP4}, {CP1, CP2}, {CP1, CP3}, {CP2, CP3}, {CP1, CP4}, {CP2, CP4}, {CP3, CP4}

[00246] For reference list X (X being 0 or 1) of a combination, the reference index with highest usage ratio in the control points is selected as the reference index of list X, and motion vectors point to difference reference picture will be scaled.

[00247] After a candidate is derived, full pruning process is performed to check whether same candidate has been inserted into the list. If a same candidate exists, the derived candidate is discarded.

[00248] (3) Padding with zero motion vectors

[00249] If the number of candidates in affine merge candidate list is less than 5, zero motion vectors with zero reference indices are insert into the candidate list, until the list is full.

[00250] **2.2.11. Pattern matched motion vector derivation**

[00251] 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.

[00252] 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.

[00253] 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.

[00254] 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.

[00255] 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.

[00256] $$M = \max\{4, \min\{\frac{M}{2^D}, \frac{N}{2^D}\}\} \quad \text{Eq. (30)}$$

[00257] As shown in the FIG. 26, 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.

[00258] As shown in FIG. 27, template matching is used to derive motion information of the current CU by finding the closest match between a template (top and/or left neighboring 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.

[00259] **2.2.11.1. CU level MV candidate set**

[00260] The MV candidate set at CU level consists of:

[00261] (i) Original AMVP candidates if the current CU is in AMVP mode

[00262] (ii) all merge candidates,

[00263] (iii) several MVs in the interpolated MV field, which is introduced in section [00279].

[00264] (iv) top and left neighboring motion vectors

[00265] 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.

[00266] 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.

[00267] When FRUC is applied in AMVP mode, the original AMVP candidates are also added to CU level MV candidate set.

[00268] 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.

[00269] **2.2.11.2. Sub-CU level MV candidate set**

[00270] The MV candidate set at sub-CU level consists of:

[00271] (i) an MV determined from a CU-level search,

[00272] (ii) top, left, top-left and top-right neighboring MVs,

[00273] (iii) scaled versions of collocated MVs from reference pictures,

[00274] (iv) up to 4 ATMVP candidates,

[00275] (v) up to 4 STMVP candidates

[00276] 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.

[00277] ATMVP and STMVP candidates are limited to the four first ones.

[00278] At the sub-CU level, up to 17 MVs are added to the candidate list.

[00279] **2.2.11.3. Generation of interpolated MV field**

[00280] 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.

[00281] 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. 28) 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.

[00282] **2.2.11.4. Interpolation and matching cost**

[00283] 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.

[00284] 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:

[00285] $C = SAD + w \cdot (|MV_x - MV_x^s| + |MV_y - MV_y^s|)$ Eq. (31)

[00286] 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.

[00287] 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.

[00288] **2.2.11.5. MV refinement**

[00289] 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.

[00290] **2.2.11.6. Selection of prediction direction in template matching FRUC merge mode**

[00291] 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;

[00292] 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.

[00293] The inter prediction direction selection is only applied to the CU-level template matching process.

[00294] **2.2.12. Generalized Bi-prediction**

[00295] In conventional bi-prediction, the predictors from L0 and L1 are averaged to generate the final predictor using the equal weight 0.5. The predictor generation formula is shown as in Eq. (32):

[00296] $$P_{TraditionalBiPred} = (P_{L0} + P_{L1} + RoundingOffset) \gg ShiftNum \quad \text{Eq. (32)}$$

[00297] In Eq. (32), $P_{TraditionalBiPred}$ is the final predictor for the conventional bi-prediction, P_{L0} and P_{L1} are predictors from L0 and L1, respectively, and *RoundingOffset and shiftNum* are used to normalize the final predictor.

[00298] Generalized Bi-prediction (GBI) proposed to allow applying different weights to predictors from L0 and L1. The predictor generation is shown in Eq. (33).

[00299] $$P_{GBi} =$$

[00300] $$\left((1 - w_1) \times P_{L0} + w_1 \times P_{L1} + RoundingOffset_{GBi}\right) \gg ShiftNum_{GBi} \quad \text{Eq. (33)}$$

[00301] In Eq. (33), P_{GBi} is the final predictor of GBi. (1-w_1) and w_1 are the selected GBI weights applied to the predictors of L0 and L1, respectively. $RoundingOffset_{GBi}$ and $shiftNum_{GBi}$ are used to normalize the final predictor in GBi.

[00302] The supported weights of w_1 is {-1/4, 3/8, 1/2, 5/8, 5/4}. One equal-weight set and four unequal-weight sets are supported. For the equal-weight case, the process to generate the final predictor is exactly the same as that in the conventional bi-prediction mode. For the true bi-prediction cases in random access (RA) condition, the number of candidate weight sets is reduced to three.

[00303] For advanced motion vector prediction (AMVP) mode, the weight selection in GBI is explicitly signaled at CU-level if this CU is coded by bi-prediction. For merge

mode, the weight selection is inherited from the merge candidate. In this proposal, GBI supports DMVR to generate the weighted average of template as well as the final predictor for BMS-1.0.

[00304] 2.2.13. Multi-hypothesis inter prediction

[00305] In the multi-hypothesis inter prediction mode, one or more additional prediction signals are signaled, in addition to the conventional uni/bi prediction signal. The resulting overall prediction signal is obtained by sample-wise weighted superposition. With the uni/bi prediction signal $p_{uni/bi}$ and the first additional inter prediction signal/hypothesis h_3, the resulting prediction signal p_3 is obtained as follows:

[00306] $p_3 = (1-\alpha)p_{uni/bi} + \alpha h_3$ Eq. (34)

[00307] The changes to the prediction unit syntax structure are shown as bold text below:

prediction_unit(x0, y0, nPbW, nPbH) {	**Descriptor**
…	
if(! cu_skip_flag[x0][y0]) {	
i = 0	
readMore = 1	
while(i < MaxNumAdditionalHypotheses && readMore) {	
additional_hypothesis_flag[x0][y0][i]	**ae(v)**
if(additional_hypothesis_flag[x0][y0][i]) {	
ref_idx_add_hyp[x0][y0][i]	**ae(v)**
mvd_coding(x0, y0, 2+i)	
mvp_add_hyp_flag[x0][y0][i]	**ae(v)**
add_hyp_weight_idx[x0][y0][i]	**ae(v)**
}	
readMore = additional_hypothesis_flag[x0][y0][i]	
i++	
}	
}	
}	

[00308] The weighting factor α is specified by the syntax element add_hyp_weight_idx, according to the following mapping:

add_hyp_weight_idx	α
0	1/4
1	-1/8

[00309] Note that for the additional prediction signals, the concept of prediction list0/list1 is abolished, and instead one combined list is used. This combined list is generated by alternatingly inserting reference frames from list0 and list1 with increasing reference index, omitting reference frames which have already been inserted, such that

double entries are avoided.

[00310] Analogously to above, more than one additional prediction signals can be used. The resulting overall prediction signal is accumulated iteratively with each additional prediction signal.

[00311] $p_{n+1} = (1 - \alpha_{n+1})p_n + \alpha_{n+1}h_{n+1}$ Eq. (35)

[00312] The resulting overall prediction signal is obtained as the last p_n (i.e., the p_n having the largest index n).

[00313] Note that also for inter prediction blocks using MERGE mode (but not SKIP mode), additional inter prediction signals can be specified. Further note, that in case of MERGE, not only the uni/bi prediction parameters, but also the additional prediction parameters of the selected merging candidate can be used for the current block.

[00314] **2.2.14. Multi-hypothesis prediction for uni-prediction of AMVP mode**

[00315] When the multi-hypothesis prediction is applied to improve uni-prediction of AMVP mode, one flag is signaled to enable or disable multi-hypothesis prediction for inter_dir equal to 1 or 2, where 1, 2, and 3 represent list 0, list 1, and bi-prediction, respectively. Moreover, one more merge index is signaled when the flag is true. In this way, multi-hypothesis prediction turns uni-prediction into bi-prediction, where one motion is acquired using the original syntax elements in AMVP mode while the other is acquired using the merge scheme. The final prediction uses 1:1 weights to combine these two predictions as in bi-prediction. The merge candidate list is first derived from merge mode with sub-CU candidates (e.g., affine, alternative temporal motion vector prediction (ATMVP)) excluded. Next, it is separated into two individual lists, one for list 0 (L0) containing all L0 motions from the candidates, and the other for list 1 (L1) containing all L1 motions. After removing redundancy and filling vacancy, two merge lists are generated for L0 and L1 respectively. There are two constraints when applying multi-hypothesis prediction for improving AMVP mode. First, it is enabled for those CUs with the luma coding block (CB) area larger than or equal to 64. Second, it is only applied to L1 when in low delay B pictures.

[00316] **2.2.15. Multi-hypothesis prediction for skip/merge mode**

[00317] When the multi-hypothesis prediction is applied to skip or merge mode,

whether to enable multi-hypothesis prediction is explicitly signaled. An extra merge indexed prediction is selected in addition to the original one. Therefore, each candidate of multi-hypothesis prediction implies a pair of merge candidates, containing one for the 1st merge indexed prediction and the other for the 2nd merge indexed prediction. However, in each pair, the merge candidate for the 2nd merge indexed prediction is implicitly derived as the succeeding merge candidate (i.e., the already signaled merge index plus one) without signaling any additional merge index. After removing redundancy by excluding those pairs, containing similar merge candidates and filling vacancy, the candidate list for multi-hypothesis prediction is formed. Then, motions from a pair of two merge candidates are acquired to generate the final prediction, where 5:3 weights are applied to the 1st and 2nd merge indexed predictions, respectively. Moreover, a merge or skip CU with multi-hypothesis prediction enabled can save the motion information of the additional hypotheses for reference of the following neighboring CUs in addition to the motion information of the existing hypotheses. Note that sub-CU candidates (e.g., affine, ATMVP) are excluded from the candidate list, and for low delay B pictures, multi-hypothesis prediction is not applied to skip mode. Moreover, when multi-hypothesis prediction is applied to merge or skip mode, for those CUs with CU width or CU height less than 16, or those CUs with both CU width and CU height equal to 16, bi-linear interpolation filter is used in motion compensation for multiple hypotheses. Therefore, the worst-case bandwidth (required access samples per sample) for each merge or skip CU with multi-hypothesis prediction enabled is calculated in Table 1 and each number is less than half of the worst-case bandwidth for each 4x4 CU with multi-hypothesis prediction disabled.

[00318] 2.2.16. Ultimate motion vector expression

[00319] Ultimate motion vector expression (UMVE) is used for either skip or merge modes with a proposed motion vector expression method.

[00320] UMVE re-uses merge candidate as same as using in VVC. Among the merge candidates, a candidate can be selected, and is further expanded by the proposed motion vector expression method.

[00321] UMVE provides a new motion vector expression with simplified signaling. The

expression method includes starting point, motion magnitude, and motion direction.

[00322] FIG. 29 shows an example of UMVE Search Process

[00323] FIG. 30 shows examples of UMVE Search Points.

[00324] This proposed technique uses a merge candidate list as it is. But only candidates which are default merge type (MRG_TYPE_DEFAULT_N) are considered for UMVE's expansion.

[00325] Base candidate index defines the starting point. Base candidate index indicates the best candidate among candidates in Table 2 as follows.

Table 2. Example Base Candidate IDX

Base candidate IDX	0	1	2	3
Nth MVP	1st MVP	2nd MVP	3rd MVP	4th MVP

[00326] If the number of base candidate is equal to 1, Base candidate IDX is not signaled.

[00327] Distance index is motion magnitude information. Distance index indicates the pre-defined distance from the starting point information. Example pre-defined distance is shown in Table 3 as follows:

Table 3. Example Distance IDX

Distance IDX	0	1	2	3	4	5	6	7
Pixel distance	1/4-pel	1/2-pel	1-pel	2-pel	4-pel	8-pel	16-pel	32-pel

[00328] Direction index represents the direction of the MVD relative to the starting point. The direction index can represent of the four directions as shown in Table 4 below.

Table 4. Example Direction IDX

Direction IDX	00	01	10	11
x-axis	+	–	N/A	N/A
y-axis	N/A	N/A	+	–

[00329] UMVE flag is singnaled right after sending a skip flag and merge flag. If skip and merge flag is true, UMVE flag is parsed. If UMVE flage is equal to 1, UMVE syntaxes are parsed. But, if not 1, AFFINE flag is parsed. If AFFINE flag is equal to 1, that is AFFINE mode, But, if not 1, skip/merge index is parsed for VTM's skip/merge mode.

[00330] Additional line buffer due to UMVE candidates is not needed. Because a skip/merge candidate of software is directly used as a base candidate. Using input UMVE index, the supplement of MV is decided right before motion compensation. There is no need to hold long line buffer for this.

[00331] **2.2.17. Affine merge mode with prediction offsets**

[00332] UMVE is extended to affine merge mode, also referred to as UMVE affine mode thereafter. The proposed method selects the first available affine merge candidate as a base predictor. Then it applies a motion vector offset to each control point's motion vector value from the base predictor. If there's no affine merge candidate available, this proposed method will not be used.

[00333] The selected base predictor's inter prediction direction, and the reference index of each direction is used without change.

[00334] In the current implementation, the current block's affine model is assumed to be a 4-parameter model, only 2 control points need to be derived. Thus, only the first 2 control points of the base predictor will be used as control point predictors.

[00335] For each control point, a zero_MVD flag is used to indicate whether the control point of current block has the same MV value as the corresponding control point predictor. If zero_MVD flag is true, there's no other signaling needed for the control point. Otherwise, a distance index and an offset direction index is signaled for the control point.

[00336] A distance offset table with size of 5 is used as shown in the table below. Distance index is signaled to indicate which distance offset to use. The mapping of distance index and distance offset values is shown in Table 5.

Table 5 - Example Distance Offset

Distance IDX	0	1	2	3	4
Distance-offset	1/2-pel	1-pel	2-pel	4-pel	8-pel

[00337] FIG. 31 shows an example of distance index and distance offset mapping.

[00338] The direction index can represent four directions as shown below, where only x or y direction may have an MV difference, but not in both directions.

Offset Direction IDX	00	01	10	11
x-dir-factor	+1	–1	0	0
y-dir-factor	0	0	+1	–1

[00339] If the inter prediction is uni-directional, the signaled distance offset is applied on the offset direction for each control point predictor. Results will be the MV value of each control point.

[00340] For example, when base predictor is uni-directional, and the motion vector values of a control point is MVP (v_{px}, v_{py}). When distance offset and direction index are signaled, the motion vectors of current block's corresponding control points will be calculated as below.

[00341] MV(v_x, v_y) = MVP (v_{px}, v_{py}) + MV(x-dir-factor * distance-offset, y-dir-factor * distance-offset) Eq. (36)

[00342] If the inter prediction is bi-directional, the signaled distance offset is applied on the signaled offset direction for control point predictor's L0 motion vector; and the same distance offset with opposite direction is applied for control point predictor's L1 motion vector. Results will be the MV values of each control point, on each inter

prediction direction.

[00343] For example, when base predictor is uni-directional, and the motion vector values of a control point on L0 is MVP_{L0} (v_{0px}, v_{0py}), and the motion vector of that control point on L1 is MVP_{L1} (v_{1px}, v_{1py}). When distance offset and direction index are signaled, the motion vectors of current block's corresponding control points will be calculated as below.

[00344] $MV_{L0}(v_{0x}, v_{0y}) = MVP_{L0}(v_{0px}, v_{0py})$ + MV(x-dir-factor * distance-offset, y-dir-factor * distance-offset) Eq. (37)

[00345] $MV_{L1}(v_{0x}, v_{0y}) = MVP_{L1}(v_{0px}, v_{0py})$ + MV(- x-dir-factor * distance-offset, - y-dir-factor * distance-offset) Eq. (38)

[00346] **2.2.18. Bi-directional optical flow**

[00347] Bi-directional Optical flow (BIO) is sample-wise motion refinement which is performed on top of block-wise motion compensation for bi-prediction. The sample-level motion refinement doesn't use signalling.

[00348] Let $I^{(k)}$ be the luma value from reference k (k=0, 1) after block motion compensation, and $\partial I^{(k)}/\partial x$, $\partial I^{(k)}/\partial y$ are horizontal and vertical components of the $I^{(k)}$ gradient, respectively. Assuming the optical flow is valid, the motion vector field (v_x, v_y) is given by an equation:

[00349] $$\partial I^{(k)}/\partial t + v_x \partial I^{(k)}/\partial x + v_y \partial I^{(k)}/\partial y = 0. \quad \text{Eq. (39)}$$

[00350] Combining this optical flow equation with Hermite interpolation for the motion trajectory of each sample results in a unique third-order polynomial that matches both the function values $I^{(k)}$ and derivatives $\partial I^{(k)}/\partial x$, $\partial I^{(k)}/\partial y$ at the ends. The value of this polynomial at t=0 is the BIO prediction:

[00351] $pred_{BIO} =$

[00352] $$1/2 \cdot (I^{(0)} + I^{(1)} + \frac{v_x}{2} \cdot \left(\frac{\tau_1 \partial I^{(1)}}{\partial x} - \frac{\tau_0 \partial I^{(0)}}{\partial x}\right) + \frac{v_y}{2} \cdot \left(\frac{\tau_1 \partial I^{(1)}}{\partial y} - \frac{\tau_0 \partial I^{(0)}}{\partial y}\right)) \quad \text{Eq. (40)}$$

[00353] Here, τ_0 and τ_1 denote the distances to the reference frames as shown in FIG. 31. Distances τ_0 and τ_1 are calculated based on POC for Ref0 and Ref1:

τ_0=POC(current) − POC(Ref0), τ_1= POC(Ref1) − POC(current). If both predictions come from the same time direction (either both from the past or both from the future) then the signs are different (i.e., $\tau_0 \cdot \tau_1 < 0$). In this case, BIO is applied only if the prediction is not from the same time moment (i.e., $\tau_0 \neq \tau_1$), both referenced regions have non-zero motion ($MVx_0, MVy_0, MVx_1, MVy_1 \neq 0$) and the block motion vectors are proportional to the time distance ($MVx_0/MVx_1 = MVy_0/MVy_1 = -\tau_0/\tau_1$).

[00354] The motion vector field (v_x, v_y) is determined by minimizing the difference Δ between values in points A and B (intersection of motion trajectory and reference frame planes on FIG. 32). Model uses only first linear term of a local Taylor expansion for Δ:

[00355] $$\Delta = \left(I^{(0)} - I^{(1)}_0 + v_x\left(\tau_1 \partial I^{(1)}/\partial x + \tau_0 \partial I^{(0)}/\partial x\right) + v_y\left(\tau_1 \partial I^{(1)}/\partial y + \tau_0 \partial I^{(0)}/\partial y\right)\right) \quad \text{Eq. (41)}$$

[00356] All values in Eq. (41) depend on the sample location (i', j'), which was omitted from the notation so far. Assuming the motion is consistent in the local surrounding area, we minimize Δ inside the (2M+1)×(2M+1) square window Ω centered on the currently predicted point (i, j), where M is equal to 2:

[00357] $$(v_x, v_y) = \underset{v_x, v_y}{\arg\min} \sum_{[i', j] \in \Omega} \Delta^2[i', j'] \quad \text{Eq. (42)}$$

[00358] For this optimization problem, the JEM uses a simplified approach making first a minimization in the vertical direction and then in the horizontal direction. This results in

[00359] $$v_x = (s_1 + r) > m ? clip3\left(-thBIO, thBIO, -\frac{s_3}{(s_1 + r)}\right) : 0 \quad \text{Eq. (43)}$$

[00360] $$v_y = (s_5 + r) > m ? clip3\left(-thBIO, thBIO, -\frac{s_6 - v_x s_2/2}{(s_5 + r)}\right) : 0 \quad \text{Eq. (44)}$$

[00361] where,

$$s_1 = \sum_{[i',j]\in\Omega}\left(\tau_1\partial I^{(1)}/\partial x + \tau_0\,\partial I^{(0)}/\partial x\right)^2; s_3 = \sum_{[i',j]\in\Omega}\left(I^{(1)} - I^{(0)}\right)\left(\tau_1\partial I^{(1)}/\partial x + \tau_0\partial I^{(0)}/\partial x\right);$$

[00362] $$s_2 = \sum_{[i',j]\in\Omega}\left(\tau_1\partial I^{(1)}/\partial x + \tau_0\,\partial I^{(0)}/\partial x\right)\left(\tau_1\partial I^{(1)}/\partial y + \tau_0\,\partial I^{(0)}/\partial y\right);$$ Eq. (45)

$$s_5 = \sum_{[i',j]\in\Omega}\left(\tau_1\,\partial I^{(1)}/\partial y + \tau_0\partial I^{(0)}/\partial y\right)^2; s_6 = \sum_{[i',j]\in\Omega}\left(I^{(1)} - I^{(0)}\right)\left(\tau_1\partial I^{(1)}/\partial y + \tau_0\partial I^{(0)}/\partial y\right)$$

[00363] In order to avoid division by zero or a very small value, regularization parameters *r* and *m* are introduced in Eq. (43) and Eq. (44).

[00364] $$r = 500 \cdot 4^{d-8}$$ Eq. (46)

[00365] $$m = 700 \cdot 4^{d-8}$$ Eq. (47)

[00366] Here d is bit depth of the video samples.

[00367] In order to keep the memory access for BIO the same as for regular bi-predictive motion compensation, all prediction and gradients values, $I^{(k)}, \partial I^{(k)}/\partial x, \partial I^{(k)}/\partial y$, are calculated only for positions inside the current block. In Eq. (45), (2M+1)×(2M+1) square window Ω centered in currently predicted point on a boundary of predicted block needs to accesses positions outside of the block (as shown in FIG. 33A). In the JEM, values of $I^{(k)}, \partial I^{(k)}/\partial x, \partial I^{(k)}/\partial y$ outside of the block are set to be equal to the nearest available value inside the block. For example, this can be implemented as padding, as shown in FIG. 33B.

[00368] With BIO, it's possible that the motion field can be refined for each sample. To reduce the computational complexity, a block-based design of BIO is used in the JEM. The motion refinement is calculated based on 4×4 block. In the block-based BIO, the values of s_n in Equation **Error! Reference source not found.** of all samples in a 4×4 block are aggregated, and then the aggregated values of s_n in are used to derived BIO motion vectors offset for the 4×4 block. More specifically, the following formula is used for block-based BIO derivation:

$$s_{1,b_k} = \sum_{(x,y)\in b_k} \sum_{[i',j]\in\Omega(x,y)} \left(\tau_1 \partial I^{(1)}/\partial x + \tau_0 \partial I^{(0)}/\partial x\right)^2; s_{3,b_k} = \sum_{(x,y)\in b_k} \sum_{[i',j]\in\Omega} \left(I^{(1)} - I^{(0)}\right)\left(\tau_1 \partial I^{(1)}/\partial x + \tau_0 \partial I^{(0)}/\partial x\right);$$
$$s_{2,b_k} = \sum_{(x,y)\in b_k} \sum_{[i',j]\in\Omega} \left(\tau_1 \partial I^{(1)}/\partial x + \tau_0 \partial I^{(0)}/\partial x\right)\left(\tau_1 \partial I^{(1)}/\partial y + \tau_0 \partial I^{(0)}/\partial y\right); \quad \text{Eq. (48)}$$
$$s_{5,b_k} = \sum_{(x,y)\in b_k} \sum_{[i',j]\in\Omega} \left(\tau_1 \partial I^{(1)}/\partial y + \tau_0 \partial I^{(0)}/\partial y\right)^2; s_{6,b_k} = \sum_{(x,y)\in b_k} \sum_{[i',j]\in\Omega} \left(I^{(1)} - I^{(0)}\right)\left(\tau_1 \partial I^{(1)}/\partial y + \tau_0 \partial I^{(0)}/\partial y\right)$$

[00369] where b_k denotes the set of samples belonging to the k-th 4x4 block of the predicted block. s_n in Eq. (43) and Eq. (44) are replaced by (($s_{n,bk}$) >> 4) to derive the associated motion vector offsets.

[00370] In some cases, MV regiment of BIO might be unreliable due to noise or irregular motion. Therefore, in BIO, the magnitude of MV regiment is clipped to a threshold value thBIO. The threshold value is determined based on whether the reference pictures of the current picture are all from one direction. If all the reference pictures of the current picture are from one direction, the value of the threshold is set to $12 \times 2^{14-d}$; otherwise, it is set to $12 \times 2^{13-d}$.

[00371] Gradients for BIO are calculated at the same time with motion compensation interpolation using operations consistent with HEVC motion compensation process (2D separable FIR). The input for this 2D separable FIR is the same reference frame sample as for motion compensation process and fractional position (*fracX*, *fracY*) according to the fractional part of block motion vector. In case of horizontal gradient $\partial I/\partial x$ signal first interpolated vertically using *BIOfilterS* corresponding to the fractional position *fracY* with de-scaling shift *d*−8, then gradient filter *BIOfilterG* is applied in horizontal direction corresponding to the fractional position *fracX* with de-scaling shift by 18−*d*. In case of vertical gradient $\partial I/\partial y$ first gradient filter is applied vertically using *BIOfilterG* corresponding to the fractional position *fracY* with de-scaling shift *d*−8, then signal displacement is performed using *BIOfilterS* in horizontal direction corresponding to the fractional position *fracX* with de-scaling shift by 18−*d*. The length of interpolation filter for gradients calculation BIOfilterG and signal displacement BIOfilterF is shorter (6-tap) in order to maintain reasonable complexity. Table 6 shows the filters used for gradients calculation for different fractional positions of block motion vector in BIO. Table 7 shows the interpolation filters used for prediction signal generation in BIO.

Table 6: Example filters for gradients calculation in BIO

Fractional pel position	Interpolation filter for gradient(BIOfilterG)
0	{ 8, −39, −3, 46, −17, 5}
1/16	{ 8, −32, −13, 50, −18, 5}
1/8	{ 7, −27, −20, 54, −19, 5}
3/16	{ 6, −21, −29, 57, −18, 5}
1/4	{ 4, −17, −36, 60, −15, 4}
5/16	{ 3, −9, −44, 61, −15, 4}
3/8	{ 1, −4, −48, 61, −13, 3}
7/16	{ 0, 1, −54, 60, −9, 2}
1/2	{ −1, 4, −57, 57, −4, 1}

Table 7: Example interpolation filters for prediction signal generation in BIO

Fractional pel position	Interpolation filter for prediction signal(BIOfilterS)
0	{ 0, 0, 64, 0, 0, 0}
1/16	{ 1, −3, 64, 4, −2, 0}
1/8	{ 1, −6, 62, 9, −3, 1}
3/16	{ 2, −8, 60, 14, −5, 1}
1/4	{ 2, −9, 57, 19, −7, 2}
5/16	{ 3, −10, 53, 24, −8, 2}
3/8	{ 3, −11, 50, 29, −9, 2}
7/16	{ 3, −11, 44, 35, −10, 3}
1/2	{ 3, −10, 35, 44, −11, 3}

[00372] In the JEM, BIO is applied to all bi-predicted blocks when the two predictions are from different reference pictures. When LIC is enabled for a CU, BIO is disabled.

[00373] In the JEM, OBMC is applied for a block after normal MC process. To reduce the computational complexity, BIO is not applied during the OBMC process. This means that BIO is only applied in the MC process for a block when using its own MV and is not

applied in the MC process when the MV of a neighboring block is used during the OBMC process.

[00374] 2.2.19. Decoder-side motion vector refinement

[00375] 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.

[00376] 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. 34. 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. 33, are used for generating the final bi-prediction results. A sum of absolute differences (SAD) is used as the cost measure. Please note that when calculating the cost of a prediction block generated by one surrounding MV, the rounded MV (to integer pel) is actually used to obtain the prediction block instead of the real MV.

[00377] 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.

[00378] 3. Related tools

[00379] **3.1.1. Diffusion filter**

[00380] In JVET-L0157, diffusion filter is proposed, wherein the intra/inter prediction signal of the CU may be further modified by diffusion filters.

[00381] **3.1.1.1. Uniform Diffusion Filter**

[00382] The Uniform Diffusion Filter is realized by convolving the prediction signal with a fixed mask that is either given as h^I or as h^{IV}, defined below.

Besides the prediction signal itself, one line of reconstructed samples left and above of the block are used as an input for the filtered signal, where the use of these reconstructed samples can be avoided on inter blocks.

[00383] Let $pred$ be the prediction signal on a given block obtained by intra or motion compensated prediction. In order to handle boundary points for the filters, the prediction signal needs to be extended to a prediction signal $pred_{ext}$. This extended prediction can be formed in two ways: Either, as an intermediate step, one line of reconstructed samples left and above the block are added to the prediction signal and then the resulting signal is mirrored in all directions. Or only the prediction signal itself is mirrored in all directions. The latter extension is used for inter blocks. In this case, only the prediction signal itself comprises the input for the extended prediction signal $pred_{ext}$.

[00384] If the filter h^I is to be used, it is proposed to replace the prediction signal $pred$ by $h^I * pred$, using the aforementioned boundary extension. Here, the filter mask h^I is given as:

[00385] $$h^I = (0.25)^4 \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 6 & 0 & 16 & 0 & 6 & 0 & 0 \\ 0 & 4 & 0 & 24 & 0 & 24 & 0 & 4 & 0 \\ 1 & 0 & 16 & 0 & 36 & 0 & 16 & 0 & 1 \\ 0 & 4 & 0 & 24 & 0 & 24 & 0 & 4 & 0 \\ 0 & 0 & 6 & 0 & 16 & 0 & 6 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \quad \text{Eq. (49)}$$

[00386] If the filter h^{IV} is to be used, it is proposed to replace the prediction signal $pred$ by $h^{IV} * pred$. Here, the filter h^{IV} is given as:

[00387] $$h^{IV} = h^I * h^I * h^I * h^I \quad \text{Eq. (50)}$$

[00388] **3.1.1.2. Directional Diffusion Filter**

[00389] Instead of using signal adaptive diffusion filters, directional filters, a horizontal filter h^{hor} and a vertical filter h^{ver}, are used which still have a fixed mask. More precisely, the uniform diffusion filtering corresponding to the mask h^I of the previous section is simply restricted to be either applied only along the vertical or along the horizontal direction. The vertical filter is realized by applying the fixed filter mask $h_{ver} = (0.5)^4 \begin{pmatrix} 1 \\ 0 \\ 4 \\ 0 \\ 6 \\ 0 \\ 4 \\ 0 \\ 1 \end{pmatrix}$ to the prediction signal and the horizontal filter is realized by using the transposed mask $h_{hor} = h_{ver}^t$.

[00390] **3.1.2. Bilateral filter**

[00391] Bilateral filter is proposed in JVET-L0406, and it is always applied to luma blocks with non-zero transform coefficients and slice quantization parameter larger than 17. Therefore, there is no need to signal the usage of the bilateral filter. Bilateral filter, if applied, is performed on decoded samples right after the inverse transform. In addition, the filter parameters, i.e., weights are explicitly derived from the coded information.

[00392] The filtering process is defined as:

[00393] $$P'_{0,0} = P_{0,0} + \sum_{k=1}^{K} W_k\big(abs(P_{k,0} - P_{0,0})\big) \times (P_{k,0} - P_{0,0}), \quad \text{Eq. (51)}$$

[00394] where $P_{0,0}$ is the intensity of the current sample and $P'_{0,0}$ is the modified intensity of the current sample, $P_{k,0}$ and W_k are the intensity and weighting parameter for the *k*-th neighboring sample, respectively. An example of one current sample and its four neighboring samples (i.e., K=4) is depicted in FIG. 35.

[00395] More specifically, the weight $W_k(x)$ associated with the *k*-th neighboring sample is defined as follows:

[00396] $$W_k(x) = Distance_k \times Range_k(x) \quad \text{Eq. (52)}$$

[00397] wherein

[00398] $$Distance_k = e^{\left(-\frac{10000}{2\sigma_d^2}\right)} \Big/ 1 + 4 * e^{\left(-\frac{10000}{2\sigma_d^2}\right)}, \quad \text{Eq. (53)}$$

[00399] $Range_k(x) = e^{\left(-\frac{x^2}{8*(QP-17)*(QP-17)}\right)}$ Eq. (54)

[00400] σ_d is dependent on the coded mode and coding block sizes. The described filtering process is applied to intra-coded blocks, and inter-coded blocks when TU is further split, to enable parallel processing.

[00401] To better capture statistical properties of video signal, and improve performance of the filter, weights function resulted from Eq. (52) are being adjusted by the σ_d parameter, shown in Table 8 as being dependent on coding mode and parameters of block partitioning (minimal size).

Table 8 Example value of $\boldsymbol{\sigma_d}$ for different block sizes and coding modes

Min(block width, block height)	Intra mode	Inter mode
4	82	62
8	72	52
Other	52	32

[00402] To further improve the coding performance, for inter-coded blocks when TU is not split, the intensity difference between current sample and one of its neighboring samples is replaced by a representative intensity difference between two windows covering current sample and the neighboring sample. Therefore, the equation of filtering process is revised to:

[00403] $P'_{0,0} = P_{0,0} + \sum_{k=1}^{N} W_k \left(\frac{1}{M}\sum_{m=-M/2}^{M/2} abs(P_{k,m} - P_{0,m})\right) \times (P_{k,0} - P_{0,0})$ Eq. (55)

[00404] wherein $P_{k,m}$ and $P_{0,m}$ represent the *m*-th sample value within the windows centered at $P_{k,0}$ and $P_{0,0}$, respectively. In this proposal, the window size is set to 3×3. An example of two windows covering $P_{2,0}$ and $P_{0,0}$ are depicted in FIG. 36.

[00405] **3.1.3. Intra block copy**

[00406] **Decoder aspect:**

[00407] In some cases, the current (partially) decoded picture is considered as a reference picture. This current picture is put in the last position of reference picture list

0. Therefore, for a slice using the current picture as the only reference picture, its slice type is considered as a P slice. The bitstream syntax in this approach follows the same syntax structure for inter coding while the decoding process is unified with inter coding. The only outstanding difference is that the block vector (which is the motion vector pointing to the current picture) always uses integer-pel resolution.

[00408] Changes from block level CPR_flag approach are:

[00409] 1. In encoder search for this mode, both block width and height are smaller than or equal to 16.

[00410] 2. Enable chroma interpolation when luma block vector is an odd integer number.

[00411] 3. Enable adaptive motion vector resolution (AMVR) for CPR mode when the SPS flag is on. In this case, when AMVR is used, a block vector can switch between 1-pel integer and 4-pel integer resolutions at block level.

[00412] **Encoder aspect:**

[00413] The encoder performs RD check for blocks with either width or height no larger than 16. For non-merge mode, the block vector search is performed using hash-based search first. If there is no valid candidate found from hash search, block matching based local search will be performed.

[00414] In the hash-based search, hash key matching (32-bit CRC) between the current block and a reference block is extended to all allowed block sizes. The hash key calculation for every position in current picture is based on 4x4 blocks. For the current block of a larger size, a hash key matching to a reference block happens when all its 4×4 blocks match the hash keys in the corresponding reference locations. If multiple reference blocks are found to match the current block with the same hash key, the block vector costs of each candidates are calculated and the one with minimum cost is selected.

[00415] In block matching search, the search range is set to be 64 pixels to the left and on top of current block.

[00416] **3.1.4. History based motion vector prediction**

[00417] PCT Application No. PCT/CN2018/093987, titled “MOTION VECTOR

PREDICTION BASED ON LOOK-UP TABLES," filed on July 2, 2018, the content of which is incorporated herein by reference, describes one or more look up tables with at least one motion candidate stored to predict motion information of a block.

[00418] A history-based MVP (HMVP) method is proposed wherein a HMVP candidate is defined as the motion information of a previously coded block. A table with multiple HMVP candidates is maintained during the encoding/decoding process. The table is emptied when a new slice is encountered. Whenever there is an inter-coded block, the associated motion information is added to the last entry of the table as a new HMVP candidate. The overall coding flow is depicted in FIG. 37. In one example, the table size is set to be L (e.g., L = 16 or 6, or 44), which indicates up to L HMVP candidates may be added to the table.

[00419] (1) In one embodiment, if there are more than L HMVP candidates from the previously coded blocks, a First-In-First-Out (FIFO) rule is applied so that the table always contains the latest previously coded L motion candidates. FIG. 38 depicts an example wherein the FIFO rule is applied to remove a HMVP candidate and add a new one to the table used in the proposed method.

[00420] (2) In another embodiment, whenever adding a new motion candidate (such as the current block is inter-coded and non-affine mode), a redundancy checking process is applied firstly to identify whether there are identical or similar motion candidates in LUTs.

[00421] 3.2 JVET-M0101

[00422] 3.2.1. Adding HEVC-style Weighted Prediction (WP)

[00423] In order to implement the HEVC-style weighted prediction, the following syntax changes (shown in bold text) can be made to PPS and slice header:

pic_parameter_set_rbsp( ) {	**Descriptor**
pps_pic_parameter_set_id	ue(v)
pps_seq_parameter_set_id	ue(v)
init_qp_minus26	se(v)
transform_skip_enabled_flag	u(1)
cu_qp_delta_enabled_flag	u(1)
if(cu_qp_delta_enabled_flag)	

diff_cu_qp_delta_depth	ue(v)
pps_cb_qp_offset	se(v)
pps_cr_qp_offset	se(v)
pps_slice_chroma_qp_offsets_present_flag	u(1)
weighted_pred_flag	**u(1)**
weighted_bipred_flag	**u(1)**
deblocking_filter_control_present_flag	u(1)
if(deblocking_filter_control_present_flag) {	
deblocking_filter_override_enabled_flag	u(1)
pps_deblocking_filter_disabled_flag	u(1)
if(!pps_deblocking_filter_disabled_flag) {	
pps_beta_offset_div2	se(v)
pps_tc_offset_div2	se(v)
}	
}	
rbsp_trailing_bits( )	
}	

slice_header( ) {	Descriptor
slice_pic_parameter_set_id	ue(v)
slice_address	u(v)
slice_type	ue(v)
if(partition_constraints_override_enabled_flag) {	
partition_constraints_override_flag	ue(v)
if(partition_constraints_override_flag) {	
slice_log2_diff_min_qt_min_cb_luma	ue(v)
slice_max_mtt_hierarchy_depth_luma	ue(v)
if(slice_max_mtt_hierarchy_depth_luma != 0)	
slice_log2_diff_max_bt_min_qt_luma	ue(v)
slice_log2_diff_max_tt_min_qt_luma	ue(v)
}	
if(slice_type = = I && qtbtt_dual_tree_intra_flag) {	
slice_log2_diff_min_qt_min_cb_chroma	ue(v)
slice_max_mtt_hierarchy_depth_chroma	ue(v)
if(slice_max_mtt_hierarchy_depth_chroma != 0)	
slice_log2_diff_max_bt_min_qt_chroma	ue(v)

slice_log2_diff_max_tt_min_qt_chroma	ue(v)
}	
}	
}	
}	
if (slice_type != I) {	
if(sps_temporal_mvp_enabled_flag)	
slice_temporal_mvp_enabled_flag	u(1)
if(slice_type == B)	
mvd_l1_zero_flag	u(1)
if(slice_temporal_mvp_enabled_flag) {	
if(slice_type == B)	
collocated_from_l0_flag	u(1)
}	
if((weighted_pred_flag && slice_type == P) || (weighted_bipred_flag && slice_type == B))	
pred_weight_table( )	
six_minus_max_num_merge_cand	ue(v)
if(sps_affine_enable_flag)	
five_minus_max_num_subblock_merge_cand	ue(v)
}	
slice_qp_delta	se(v)
if(pps_slice_chroma_qp_offsets_present_flag) {	
slice_cb_qp_offset	se(v)
slice_cr_qp_offset	se(v)
}	
if(sps_sao_enabled_flag) {	
slice_sao_luma_flag	u(1)
if(ChromaArrayType != 0)	
slice_sao_chroma_flag	u(1)
}	
if(sps_alf_enabled_flag) {	
slice_alf_enabled_flag	u(1)
if(slice_alf_enabled_flag)	
alf_data( )	
}	

dep_quant_enabled_flag	u(1)
if(!dep_quant_enabled_flag)	
sign_data_hiding_enabled_flag	u(1)
if(deblocking_filter_override_enabled_flag)	
deblocking_filter_override_flag	u(1)
if(deblocking_filter_override_flag) {	
slice_deblocking_filter_disabled_flag	u(1)
if(!slice_deblocking_filter_disabled_flag) {	
slice_beta_offset_div2	se(v)
slice_tc_offset_div2	se(v)
}	
}	
byte_alignment( )	
}	

pred_weight_table( ) {	Descriptor
luma_log2_weight_denom	ue(v)
if(ChromaArrayType != 0)	
delta_chroma_log2_weight_denom	se(v)
for(i = 0; i <= num_ref_idx_l0_active_minus1; i++)	
luma_weight_l0_flag[i]	u(1)
if(ChromaArrayType != 0)	
for(i = 0; i <= num_ref_idx_l0_active_minus1; i++)	
chroma_weight_l0_flag[i]	u(1)
for(i = 0; i <= num_ref_idx_l0_active_minus1; i++) {	
if(luma_weight_l0_flag[i]) {	
delta_luma_weight_l0[i]	se(v)
luma_offset_l0[i]	se(v)
}	
if(chroma_weight_l0_flag[i])	
for(j = 0; j < 2; j++) {	
delta_chroma_weight_l0[i][j]	se(v)
delta_chroma_offset_l0[i][j]	se(v)
}	
}	
if(slice_type = = B) {	
for(i = 0; i <= num_ref_idx_l1_active_minus1; i++)	
luma_weight_l1_flag[i]	u(1)
if(ChromaArrayType != 0)	
for(i = 0; i <= num_ref_idx_l1_active_minus1; i++)	
chroma_weight_l1_flag[i]	u(1)
for(i = 0; i <= num_ref_idx_l1_active_minus1; i++) {	
if(luma_weight_l1_flag[i]) {	
delta_luma_weight_l1[i]	se(v)
luma_offset_l1[i]	se(v)
}	
if(chroma_weight_l1_flag[i])	
for(j = 0; j < 2; j++) {	
delta_chroma_weight_l1[i][j]	se(v)
delta_chroma_offset_l1[i][j]	se(v)
}	

}	
}	
}	

[00424] 3.2.2. Invoking (bi-prediction bi-prediction with weighted averaging (BWA), a.k.a GBI) and WP in a mutually exclusive manner

[00425] Though BWA and WP both apply weights to the motion compensated prediction signals, the specific operations are different. In the bi-prediction case, WP applies a linear weight and a constant offset to each of the prediction signals depending on the (weight, offset) parameters associated with the ref_idx that is used to generate the prediction signal. Whereas there are some range constraints on these (weight, offset) values, the allowed ranges are relatively large and there is no normalization constraint on the parameter values. WP parameters are signaled at the picture level (in slice headers). WP has been shown to provide very large coding gain for fading sequences, and is generally expected to be enabled for such content.

[00426] BWA uses a CU level index to indicate how to combine the two prediction signals in the case of bi-prediction. The BWA applies weighted averaging, i.e., the two weights add up to 1. BWA provides about 0.66% coding gain for the CTC RA configuration, but is much less effective than WP for the fading content (JVET-L0201).

[00427] Method #1:

[00428] The PPS syntax element weight_bipred_flag is checked to determine if the gbi_idx is signaled at the current CU. The syntax signaling is modified as follows in bold text:

coding_unit(x0, y0, cbWidth, cbHeight, treeType) {	Descriptor
…	
if(sps_gbi_enabled_flag && inter_pred_idc[x0][y0] = = PRED_BI && **!weighted_bipred_flag &&** cbWidth * cbHeight >= 256)	
gbi_idx[x0][y0]	ae(v)
…	

}	

[00429] The first alternative will disable BWA for all the reference pictures of the current slice, if the current slice refers to a PPS for which weighted_bipred_flag is set to 1.

[00430] Method #2:

[00431] BWA is disabled only if both of the reference pictures used in the bi-prediction have turned on weighted prediction, i.e., the (weight, offset) parameters of these reference pictures have non-default values. This allows the bi-predicted CUs that use reference pictures with default WP parameters (i.e. WP is not invoked for these CUs) to still be able to use BWA. The syntax signaling is modified as follows in bold text:

coding_unit(x0, y0, cbWidth, cbHeight, treeType) {	Descriptor
…	
if(sps_gbi_enabled_flag && inter_pred_idc[x0][y0] = = PRED_BI && **luma_weight_l0_flag [ref_idx_l0[x0][y0]] == 0 && chroma_weight_l0_flag [ref_idx_l0[x0][y0]] == 0 && luma_weight_l1_flag [ref_idx_l1[x0][y0]] == 0 && chroma_weight_l1_flag [ref_idx_l1[x0][y0]] == 0 &&** cbWidth * cbHeight >= 256)	
gbi_idx[x0][y0]	ae(v)
…	
}	

[00432] **4. Problems**

[00433] In LIC, two parameters including scaling parameter *a* and offset *b* need to be derived by using neighboring reconstructed pixels, which may cause latency issue.

[00434] Hybrid intra and inter prediction, diffusion filter, bilateral filter, OBMC and LIC need to further modify the inter prediction signal in different ways, and they all have latency issue.

[00435] **5. Some example embodiments and techniques**

[00436] The detailed inventions below should be considered as examples to explain general concepts. These inventions should not be interpreted in a narrow way. Furthermore, these inventions can be combined in any manner.

[00437] Example 1:

[00438] In some embodiments, LIC can only be performed for blocks located at CTU boundaries (called boundary blocks), and neighboring reconstructed samples out of the current CTU may be used for deriving LIC parameters. In this case, for those blocks that are not located CTU boundaries (called inner blocks), LIC is always disabled without being signaled.

[00439] In some embodiments, selection of neighboring reconstructed samples outside current CTU may depend on the position of the block relative to the CTU covering the current block.

[00440] In some embodiments, for blocks at the left boundary of a CTU, only left reconstructed neighboring samples of the CU may be used to derive the LIC parameters.

[00441] In some embodiments, for blocks at the top boundary of a CTU, only above reconstructed neighboring samples of the block may be used to derive the LIC parameters.

[00442] In some embodiments, for blocks at the top-left corner of a CTU, left and/or above reconstructed neighboring samples of the block may be used to derive the LIC parameters.

[00443] In some embodiments, for blocks at the top-right corner of a CTU, top-right and/or above reconstructed neighboring samples of the block are used to derive the LIC parameters.

[00444] In some embodiments, for blocks at the bottom-left corner of a CTU, bottom-left and/or left reconstructed neighboring samples of the block are used to derive the LIC parameters.

[00445] Example 2:

[00446] In some embodiments, sets of LIC parameters can be derived for blocks located at CTU boundary only, and inner blocks of the CTU may inherit from one or multiple sets of these LIC parameter sets.

[00447] In some embodiments, a LIC parameter lookup table is maintained for each CTU, and each set of the derived LIC parameter is inserted into the LIC parameter

table. Method described in PCT Application No., PCT/CN2018/093987, titled "MOTION VECTOR PREDICTION BASED ON LOOK-UP TABLES," filed on July 2, 2018, the content of which is incorporated herein by reference, can be used to maintain the LIC parameter lookup table.

[00448] In some embodiments, such lookup table is maintained for each reference picture, and when deriving the LIC parameters, LIC parameters are derived for all reference pictures.

[00449] In some embodiments, such lookup table is emptied at the beginning of each CTU or CTU row or slice or tile or tile group or picture.

[00450] In some embodiments, for inner block coded with AMVP mode or affine inter mode, if LIC flag is true, the used set of LIC parameters is explicitly signaled. For example, an index is signaled to indicate which entry of the lookup table is used for each reference picture. In some embodiments, for uni-predicted block, if there is no valid entry in the lookup table for the reference picture, the LIC flag shall be false. In some embodiments, for bi-predicted block, if there is no valid entry in the lookup table for any of the reference picture, the LIC flag shall be false. In some embodiments, for bi-predicted block, if there is valid entry in the lookup table for at least one of the reference pictures, the LIC flag can be true. One LIC parameter index is signaled for each reference picture with valid LIC parameters.

[00451] In some embodiments, more than one lookup tables are maintained. In some embodiments, when encoding/decoding the current CTU, it only uses parameters from lookup tables generated by some previously encoded/decoded CTUs, and the lookup table generated by the current CTU is used for some following CTUs. In some embodiments, each lookup table may correspond to one reference picture/one reference picture list/a certain range of block sizes/a certain coded mode/a certain block shape, etc. al.

[00452] In some embodiments, if an inner block is coded with merge or affine merge mode, for spatial or/and temporal merge candidate, both LIC flag and LIC parameters are inherited from the corresponding neighboring block.

[00453] In some embodiments, the LIC flag is inherited and the LIC parameter may be

signaled explicitly. In some embodiments, the LIC flag is signaled explicitly and the LIC parameter may be inherited. In some embodiments, both LIC flag and LIC parameter are signaled explicitly. In some embodiments, the differences between the LIP parameters of the block and the inherited parameters are signaled.

[00454] In some embodiments, for boundary block coded with AMVP mode or affine inter mode, if LIC flag is true, the used LIC parameter is derived implicitly using same methods as in [00148].

[00455] In some embodiments, if a boundary block is coded with merge or affine merge mode, for spatial or/and temporal merge candidate, both LIC flag and LIC parameter are inherited from the corresponding neighboring block.

[00456] In some embodiments, the LIC flag is inherited and the LIC parameters may be signaled explicitly. In some embodiments, the LIC flag is signaled explicitly and the LIC parameters may be inherited. In some embodiments, both LIC flag and LIC parameter are signaled explicitly.

[00457] In some embodiments, if a block is coded with merge mode, for combined merge candidate or average merge candidate, LIC is always disabled.

[00458] In some embodiments, if LIC flag of any of the two merge candidates that are used for generating the combined merge candidate or average merge candidate is true, the LIC flag is set to true._In some embodiments, for an inner block, if both merge candidates that are used for generating the combined merge candidate or average merge candidate use LIC, it may inherit LIC parameters from any of them. In some embodiments, the LIC flag is inherited and the LIC parameter may be signaled explicitly. In some embodiments, the LIC flag is signaled explicitly and the LIC parameter may be inherited. In some embodiments, both LIC flag and LIC parameter are signaled explicitly.

[00459] In some embodiments, if a block is coded with merge mode from a HMVP merge candidate, LIC is always disabled. In some embodiments, if a block is coded with merge mode from a HMVP merge candidate, LIC is always enabled. In some embodiments, if LIC flag of the HMVP merge candidate is true, the LIC flag is set to be true. For example, LIC flag can be signaled explicitly. In some embodiments, if LIC flag

of the HMVP merge candidate is true, the LIC flag is set to be true, and the LIP parameters are inherited from the HMVP merge candidate. In some embodiments, LIC parameters are signaled explicitly. In some embodiments, LIC parameters are derived implicitly.

[00460] Example 3:

[00461] LIC may be used together with intra block copy (IBC, or current picture referencing) mode.

[00462] In some embodiments, if a block is coded with intra block copy mode, indication of LIC usage (e.g., a LIC flag) may be further signaled. In some embodiments, in merge mode, if an IBC coded block inherits motion information from a neighboring block, it may also inherit the LIC flag. In some embodiments, it may also inherit the LIC parameters.

[00463] In some embodiments, if LIC flag is true, LIC parameter is derived implicitly using same method as in [00148]. In some embodiments, method described in Example 1 and/or Example 2 may be used. In some embodiments, if a block is coded with LIC enabled, indication of IBC usage may be further signaled.

[00464] Example 4:

[00465] In some embodiments, hybrid intra and inter prediction, diffusion filter, bilateral filter, OBMC, LIC or any other tools that modify the inter prediction signal or modify the reconstructed block from motion compensation (e.g., causing latency issues) are exclusively used.

[00466] In some embodiments, if LIC is enabled, all other tools are disabled implicitly. If LIC flag is true (either explicitly signaled or implicitly derived), on/off flag of other tools (if there is any) are not signaled and implicitly derived to be off.

[00467] In some embodiments, if hybrid intra and inter prediction is enabled, all other tools are disabled implicitly. If hybrid intra and inter prediction flag is true (either explicitly signaled or implicitly derived), on/off flag of other tools (if there is any) are not signaled and implicitly derived to be off.

[00468] In some embodiments, if diffusion filter is enabled, all other tools are disabled implicitly. If diffusion filter flag is true (either explicitly signaled or implicitly derived),

on/off flag of other tools (if there is any) are not signaled and implicitly derived to be off.

[00469] In some embodiments, if OBMC is enabled, all other tools are disabled implicitly. If OBMC flag is true (either explicitly signaled or implicitly derived), on/off flag of other tools (if there is any) are not signaled and implicitly derived to be off.

[00470] In some embodiments, if bilateral filter is enabled, all other tools are disabled implicitly. If bilateral filter flag is true (either explicitly signaled or implicitly derived), on/off flag of other tools (if there is any) are not signaled and implicitly derived to be off.

[00471] In some embodiments, different tools may be checked in order. Furthermore, such checking process terminates when one of the above tools is decided to be enabled. In some embodiments, the checking order is LIC → diffusion filter → hybrid intra and inter prediction → OBMC → bilateral filter. In some embodiments, the checking order is LIC → diffusion filter → bilateral filter → hybrid intra and inter prediction → OBMC.

[00472] In some embodiments, the checking order is LIC → OBMC → diffusion filter → bilateral filter → hybrid intra and inter prediction. In some embodiments, the checking order is LIC → OBMC → diffusion filter → hybrid intra and inter prediction → bilateral filter. In some embodiments, the order may be adaptively changed based on previously coded information and/or based on coded information of current block (e.g., block size/reference picture/MV information/Low delay check flag/tile/picture/slice types), such as based on the modes of neighboring blocks.

[00473] Example 5:

[00474] In some embodiments, LIC is used exclusively with GBI or multi-hypothesis inter prediction.

[00475] In some embodiments, GBI information may be signaled after LIC information. In some embodiments, whether to signal GBI information may depend on the signaled/inferred LIC information. Furthermore, whether to signal GBI information may depend on both LIC information and weighted prediction information associated with at least one reference picture of current block or of current tile/tile group/picture containing current block. In some embodiments, the information may be signaled in SPS/PPS/slice header/tile group header/tile/CTU/CU.

[00476] In some embodiments, LIC information may be signaled after GBI information. In some embodiments, whether to signal LIC information may depend on the signaled/inferred GBI information. Furthermore, whether to signal LIC information may depend on both GBI information and weighted prediction information associated with at least one reference picture of current block or of current tile/tile group/picture containing current block. In some embodiments, the information may be signaled in SPS/PPS/slice header/tile group header/tile/CTU/CU.

[00477] In some embodiments, if LIC flag is true, syntax elements required by GBI or multi-hypothesis inter prediction are not signaled.

[00478] In some embodiments, if GBI flag is true (e.g., unequal weights are applied to two or multiple reference pictures), syntax elements required by LIC or multi-hypothesis inter prediction are not signaled.

[00479] In some embodiments, if GBI is enabled for a block with unequal weights for two or multipole reference pictures, syntax elements required by LIC or multi-hypothesis inter prediction are not signaled.

[00480] In some embodiments, LIC is used exclusively with sub-block technologies, such as affine mode. In some embodiments, LIC is used exclusively with triangular prediction mode. In some embodiments, when triangular prediction mode is enabled for one block, LIC is always disabled. In some embodiments, the LIC flag of one TPM merge candidate may be inherited from a spatial or temporal block or other kinds of motion candidates (e.g., HMVP candidates). In some embodiments, the LIC flag of one TPM merge candidate may be inherited from some spatial or temporal blocks (e.g., only A1, B1). In some embodiments, LIC flag of one TPM merge candidate may be always set to false.

[00481] In some embodiments, if multi-hypothesis inter prediction is enabled, syntax elements required by LIC or GBI are not signaled.

[00482] In some embodiments, different tools may be checked in a certain order. Furthermore, such checking process terminates when one of the above tools is decided to be enabled. In some embodiments, the checking order is LIC -> GBI -> multi-hypothesis inter prediction. In some embodiments, the checking order is LIC -> multi-

hypothesis inter prediction -> GBI. In some embodiments, the checking order is GBI -> LIC -> multi-hypothesis inter prediction.

[00483] In some embodiments, the checking order is GBI -> multi-hypothesis inter prediction -> LIC. In some embodiments, the order may be adaptively changed based on previously coded information and/or based on coded information of current block (e.g., block size/reference picture/MV information/Low delay check flag/tile/picture/slice types), such as based on the modes of neighboring blocks.

[00484] In some embodiments, the above methods may be applied only when current block is uni-prediction.

[00485] Example 6:

[00486] In some embodiments, LIC is used exclusively with combined inter-intra prediction (CIIP). In some embodiments, CIIP information may be signaled after LIC information. In some embodiments, whether to signal CIIP information may depend on the signaled/inferred LIC information. Furthermore, whether to signal CIIP information may depend on both LIC information and weighted prediction information associated with at least one reference picture of current block or of current tile/tile group/picture containing current block. In some embodiments, the information may be signaled in SPS/PPS/slice header/tile group header/tile/CTU/CU.

[00487] In some embodiments, LIC information may be signaled after CIIP information.

[00488] In some embodiments, whether to signal LIC information may depend on the signaled/inferred CIIP information. Furthermore, whether to signal LIC information may depend on both CIIP information and weighted prediction information associated with at least one reference picture of current block or of current tile/tile group/picture containing current block. In some embodiments, the information may be signaled in SPS/PPS/slice header/tile group header/tile/CTU/CU.

[00489] In some embodiments, if LIC flag is true, syntax elements required by CIIP are not signaled. In some embodiments, if CIIP flag is true, syntax elements required by LIC are not signaled. In some embodiments, if CIIP is enabled, syntax elements required by LIC are not signaled.

[00490] Example 7:

[00491] In some embodiments, weighted prediction cannot be applied when one or some of the coding tools below is/are applied (exclusive with weighted prediction):

[00492] a. BIO (a.k.a. BDOF)

[00493] b. CIIP

[00494] c. Affine prediction

[00495] d. Overlapped block motion compensation (OBMC)

[00496] e. Decoder side motion vector refinement (DMVR)

[00497] In some embodiments, due to the weighted prediction being enabled/disabled, then some of the coding tools are disabled/enabled accordingly. In some embodiments, due to some of the coding tools being enabled/disabled, the weighted prediction is disabled/enabled accordingly.

[00498] In some embodiments, if a coding tool and the weighted prediction are mutually exclusive, the information (e.g., a flag) indicating whether the coding tool is used is not signaled when weighted prediction is applied. Such information (e.g., the flag) can be inferred to be zero. In some embodiments, the information may still be signaled expressively in SPS/PPS/slice header/tile group header/tile/CTU/CU.

[00499] Example 8:

[00500] In some embodiments, LIC may be used exclusively with weighted prediction at block level.

[00501] In some embodiments, when current block is bi-prediction, the signalling of LIC information may depend on the weighted_bipred_flag.

[00502] In some embodiments, when current block is uni-prediction, the signalling of LIC information may depend on the weighted_pred_flag.

[00503] In some embodiments, signaling of LIC information may depend on the weighted prediction parameters associated with one or all reference pictures associated with current block.

[00504] In some embodiments, if weighted prediction is enabled for some reference picture or all reference pictures of a block, LIC may be disabled for the block and LIC related syntax elements are not signaled.

[00505] In some embodiments, even if weighted prediction is enabled for some reference picture or all reference pictures of a block, LIC may be still applied and weighted prediction may be disabled for the block. In some embodiments, LIC may be applied on reference pictures where weighted prediction are not applied, and LIC may be disabled on reference pictures where weighted prediction are applied.

[00506] In some embodiments, LIC may be used with weighted prediction together for one block.

[00507] Example 9:

[00508] In some embodiments, LIC may be used exclusively with weighted prediction at picture level/slice level/tile group level/CTU group level.

[00509] In some embodiments, if weighted prediction is enabled for some reference picture or all reference pictures of a picture/slice/tile group/CTU group, LIC is disabled and all related syntax element are not signaled.

[00510] In some embodiments, if LIC is enabled for a picture/slice/tile group/CTU group, weighted prediction is disabled for all its reference pictures and all related syntax elements are not signaled.

[00511] In some embodiments, LIC may be used with weighted prediction together.

[00512] Example 10:

[00513] In some embodiments, LIC may be used exclusively with CPR mode. In some embodiments, when CPR mode is enabled for a block, signalling of indications of LIC usage and/or side information may be skipped. In some embodiments, when LIC mode is enabled for a block, signalling of indications of CPR usage and/or side information may be skipped.

[00514] Example 11:

[00515] In some embodiments, LIC may be used together with multi-hypothesis prediction (as described in [00304], [00314], [00316]).

[00516] In some embodiments, LIC flag is explicitly signaled for multi-hypothesis AMVP and merge mode (e.g., as described in [00314]). In some embodiments, the explicitly signaled LIC flag is applied to both AMVP mode and merge mode. In some embodiments, the explicitly signaled LIC flag is applied only to AMVP mode, while the

LIC flag for the merge mode is inherited from the corresponding merge candidate. Different LIC flag may be used for AMVP mode and merge mode. Meanwhile, different LIC parameters may be derived/inherited for AMVP mode and merge mode. In some embodiments, LIC is always disabled for the merge mode. In some embodiment, LIC flag is not signaled and LIC is always disabled for AMVP mode. However, for merge mode, LIC flag or/and LIC parameter may be inherited or derived.

[00517] In some embodiments, LIC flag is inherited from the corresponding merge candidates for multi-hypothesis merge mode (e.g., as described in [00316]). In some embodiments, LIC flag is inherited for each of the two selected merge candidates, therefore, different LIC flag may be inherited for the two selected merge candidates. Meanwhile, different LIC parameter may be derived/inherited for the two selected merge candidates. In some embodiments, LIC flag is only inherited for the 1st selected merge candidate and LIC is always disabled for the 2nd selected merge candidate.

[00518] In some embodiments, LIC flag is explicitly signaled for multi-hypothesis inter prediction mode (e.g., as described in [00304]). In some embodiments, if the block is predicted with merge mode (or UMVE mode) and additional motion information, the explicitly signaled LIC flag may be applied to both merge mode (or UMVE mode) and the additional motion information. In some embodiments, if the block is predicted with merge mode (or UMVE mode) and additional motion information, the explicitly signaled LIC flag may be applied to the additional motion information. While for merge mode, LIC flag or/and LIC parameter may be inherited or derived. In some embodiments, LIC is always disabled for the merge mode.

[00519] In some embodiments, if the block is predicted with merge mode (or UMVE mode) and additional motion information, LIC flag is not signaled and disabled for the additional motion information. While for merge mode, LIC flag or/and LIC parameter may be inherited or derived. In some embodiments, if the block is predicted with AMVP mode and additional motion information, the explicitly signaled LIC flag may be applied to both AMVP mode and the additional motion information. In some embodiments, LIC is always disabled for the additional motion information. In some embodiments, different LIC parameters may be derived/inherited for the merge mode (or UMVE mode) / AMVP

mode and the additional motion information.

[00520] In some embodiments, when multi-hypothesis is applied to a block, illumination compensation may be applied to certain prediction signals, instead of all prediction signals. In some embodiments, when multi-hypothesis is applied to a block, more than one flag may be signaled/derived to indicate the usage of illumination compensation for prediction signals.

[00521] Example 12:

[00522] In some embodiments, LIC flag may be inherited from the base merge candidate in UMVE mode. In some embodiments, LIC parameters are derived implicitly as described in [00148]. In some embodiments, for boundary block coded in UMVE mode, LIC parameters are derived implicitly as described in [00148]. In some embodiments, for inner block coded in UMVE mode, LIC parameters are inherited from base merge candidate. In some embodiments, LIC flag may be signaled explicitly in UMVE mode.

[00523] Example 13:

[00524] The above proposed methods or LIC may be applied under certain conditions, such as block sizes, slice/picture/tile types, or motion information.

[00525] In some embodiments, when a block size contains smaller than M*H samples, e.g., 16 or 32 or 64 luma samples, proposed method or LIC are not allowed.

[00526] In some embodiments, when minimum size of a block's width or/and height is smaller than or no larger than X, proposed method or LIC are not allowed. In one example, X is set to 8.

[00527] In some embodiments, when a block's width > th1 or >=th1 and/or a block's height > th2 or >=th2, proposed method or LIC are not allowed. In one example, th1 and/or th2 is set to 8.

[00528] In some embodiments, when a block's width < th1 or <=th1 and/or a block's height < th2 or <a=th2, proposed method or LIC are not allowed. In one example, th1 and/or th2 is set to 8.

[00529] In some embodiments, LIC is disabled for affine inter modes or/and affine merge mode.

[00530] In some embodiments, LIC is disabled for sub-block coding tools like ATMVP or/and STMVP or/and planar motion vector prediction modes.

[00531] In some embodiments, LIC is only applied to some components. For example, LIC is applied to luma component. Alternatively, LIC is applied to chroma component.

[00532] In some embodiments, BIO or/and DMVR is disabled if LIC flag is true.

[00533] In some embodiments, LIC is disabled for bi-predicted block.

[00534] In some embodiments, LIC is disabled for inner block coded with AMVP mode.

[00535] In some embodiments, LIC is only allowed for uni-predicted block.

[00536] Example 14:

[00537] In some embodiments, selection of neighboring samples used for deriving LIC parameters may depend on coded information, block shape, etc. al.

[00538] In some embodiments, if width >= height or width > height, only above neighboring pixels are used for deriving LIC parameters.

[00539] In some embodiments, if width < height, only left neighboring pixels are used for deriving LIC parameters.

[00540] Example 15:

[00541] In some embodiments, only one LIC flag may be signaled for a block with geometry portioning structure (such as triangular prediction mode). In this case, all partitions of the block (all PUs) share the same value of LIC enabling flag.

[00542] In some embodiments, for some PU, the LIC may be always disabled regardless the signaled LIC flag.

[00543] In some embodiments, if the block is split from the top-right corner to the bottom-left corner, one set of LIC parameters is derived and used for both PUs. In some embodiments, LIC is always disabled for the bottom PU. In some embodiments, if the top PU is coded in merge mode, LIC flag is not signaled.

[00544] In some embodiments, if the block is split from the top-left corner to the bottom-right corner, LIC parameters are derived for each PU. In some embodiments, above neighboring samples of the block are used for deriving LIC parameters of the top

PU and left neighboring sample of the block are used for deriving LIC parameters of the left PU. In some embodiments, one set of LIC parameters is derived and used for both PUs.

[00545] In some embodiments, if both PUs are code in merge mode, LIC flag is not signaled and may be inherited from merge candidates. LIC parameters may be derived or inherited.

[00546] In some embodiments, if one PU is coded in AMVP mode and another PU is coded in merge mode, the signaled LIC flag may be applied to PU coded in AMVP mode only. For PU coded in merge mode, LIC flag or/and LIC parameters are inherited. In some embodiments, LIC flag is not signaled and is disabled for the PU coded in AMVP mode. While, for PU coded in merge mode, LIC flag or/and LIC parameters are inherited. In some embodiments, LIC is disabled for the PU coded in merge mode.

[00547] In some embodiments, if both PUs are coded in AMVP mode, one LIC flag may be signaled for each PU.

[00548] In some embodiments, one PU may utilize reconstructed samples from another PU within current block which has been reconstructed to derive LIC parameters.

[00549] Example 16:

[00550] In some embodiments, LIC may be performed for partial pixels instead of the whole block.

[00551] In some embodiments, LIC may be only performed for pixels around block boundaries only, and for other pixels within the block, LIC is not performed.

[00552] In some embodiments, LIC is performed for the top WxN rows or NxH columns, wherein N is an integer number, W and H denote the width and the height of the block. For example, N is equal to 4.

[00553] In some embodiments, LIC is performed for the top-left (W-m)x(H-n) region, wherein m and n are integer numbers, W and H denote the width and the height of the block. For example, m and n are equal to 4.

[00554] Example 17:

[00555] In some embodiments, LIC flags may be utilized to update HMVP tables.

[00556] In some embodiments, an HMVP candidate may also include the LIC flag in addition to the motion vectors and other stuff.

[00557] In some embodiments, LIC flags are not utilized to update HMVP tables. For example, a default LIC flag value is set for each HMVP candidate.

[00558] Example 18:

[00559] In some embodiments, if one block is coded with LIC, updating of HMVP tables may be skipped. In some embodiments, LIC coded blocks may also be utilized to update HMVP tables.

[00560] Example 19:

[00561] In some embodiments, motion candidates may be reordered according to LIC flags.

[00562] In some embodiments, candidates with LIC enabled may be put before all or partial of candidates with LIC disabled in the merge/AMVP or other motion candidate list.

[00563] In some embodiments, candidates with LIC disabled may be put before all or partial of candidates with LIC enabled in the merge/AMVP or other motion candidate list.

[00564] In some embodiments, for LIC coded blocks and non-LIC coded blocks, the merge/AMVP or other motion candidate list construction process may be different.

[00565] In some embodiments, for LIC coded blocks, the merge/AMVP or other motion candidate list may not contain motion candidates derived from non-LIC-coded spatial/temporal neighboring or non-adjacent blocks or HMVP candidates with LIC flag equal to false.

[00566] In some embodiments, for non-LIC coded blocks, the merge/AMVP or other motion candidate list may not contain motion candidates derived from LIC-coded spatial/temporal neighboring or non-adjacent blocks or HMVP candidates with LIC flag equal to true.

[00567] Example 20:

[00568] In some embodiments, whether to enable or disable the above methods may be signaled in SPS/PPS/VPS/sequence header/picture header/slice header/tile group

header/group of CTUs, etc. al. In some embodiments, which method to be used may be signaled in SPS/PPS/VPS/sequence header/picture header/slice header/tile group header/group of CTUs, etc. al. In some embodiments, whether to enable or disable the above methods and/or which method to be applied may be dependent on block dimension, video processing data unit (VPDU), picture type, low delay check flag, coded information of current block (such as reference pictures, uni or bi-prediction) or previously coded blocks.

[00569] **6. Additional Embodiments**

[00570] Method #1:

[00571] The PPS syntax element weight_bipred_flag is checked to determine if the LIC_enabled_flag is signaled at the current CU. The syntax signalling is modified shown as bold text below:

coding_unit(x0, y0, cbWidth, cbHeight, treeType) {	Descriptor
…	
if**(sps_LIC_enabled_flag && ((inter_pred_idc[x0][y0] = = PRED_BI && !weighted_bipred_flag) || (inter_pred_idc[x0][y0] = = PRED_UNI && !weighted_pred_flag))**) {	
LIC_enabled_flag [x0][y0]	ae(v)
…	
}	
…	
}	

[00572] The first alternative can disable LIC for all the reference pictures of the current slice/tile/tile group. For example,

[00573] 1. if the current slice refers to a PPS for which weighted_bipred_flag is set to 1 and current block is bi-predicted;

[00574] 2. if the current slice refers to a PPS for which weighted_pred_flag is set to 1 and current block is uni-predicted.

[00575] Method #2:

[00576] *For the bi-prediction case:*

[00577] LIC is disabled only if both of the reference pictures used in the bi-prediction

have turned on weighted prediction, i.e., the (weight, offset) parameters of these reference pictures have non-default values. This allows the bi-predicted CUs that use reference pictures with default WP parameters (i.e. WP is not invoked for these CUs) to still be able to use LIC.

[00578] *For the uni-prediction case:*

[00579] LIC is disabled only if the reference picture used in the uni-prediction have turned on weighted prediction, i.e., the (weight, offset) parameters of these reference pictures have non-default values. This allows the uni-predicted CUs that use reference pictures with default WP parameters (i.e. WP is not invoked for these CUs) to still be able to use LIC. The syntax signalling is modified shown as bold text below:

coding_unit(x0, y0, cbWidth, cbHeight, treeType) {	Descriptor
…	
if(sps_gbi_enabled_flag && ((inter_pred_idc[x0][y0] = = PRED_BI && luma_weight_l0_flag [ref_idx_l0[x0][y0]] == 0 && chroma_weight_l0_flag [ref_idx_l0[x0][y0]] == 0 && luma_weight_l1_flag [ref_idx_l1[x0][y0]] == 0 && chroma_weight_l1_flag [ref_idx_l1[x0][y0]] == 0) ‖ (inter_pred_idc[x0][y0] = = PRED_UNI && luma_weight_lX_flag [ref_idx_lX[x0][y0]] == 0 && chroma_weight_lX_flag [ref_idx_lX[x0][y0]] == 0 &&)))	
LIC_enabled_flag [x0][y0]	ae(v)
…	
}	

[00580] In above table, X indicates the reference picture list (X being 0 or 1).

[00581] For above examples, the following may apply:

[00582] 1. sps_LIC_enabled_flag which controls the usage of LIC per sequence may be replaced by indications of usage of LIC per picture/view/slice/tile/tile groups/CTU

rows/regions/multiple CTUs/CTU.

[00583] 2. The control of signalling of LIC_enabled_flag may depend on block dimensions.

[00584] 3. The control of signalling of LIC_enabled_flag may depend on gbi index.

[00585] 4. The control of signalling of gbi index may further depend on LIC_enabled_flag.

[00586] FIG. 39 is a block diagram illustrating an example of the architecture for a computer system or other control device 2600 that can be utilized to implement various portions of the presently disclosed technology. In FIG. 39, the computer system 2600 includes one or more processors 2605 and memory 2610 connected via an interconnect 2625. The interconnect 2625 may represent any one or more separate physical buses, point to point connections, or both, connected by appropriate bridges, adapters, or controllers. The interconnect 2625, 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.”

[00587] The processor(s) 2605 may include central processing units (CPUs) to control the overall operation of, for example, the host computer. In certain embodiments, the processor(s) 2605 accomplish this by executing software or firmware stored in memory 2610. The processor(s) 2605 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.

[00588] The memory 2610 can be or include the main memory of the computer system. The memory 2610 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 2610 may contain, among other things, a set of machine instructions which, when executed by processor 2605, causes the processor 2605 to perform operations to implement embodiments of the presently disclosed technology.

[00589] Also connected to the processor(s) 2605 through the interconnect 2625 is a (optional) network adapter 2615. The network adapter 2615 provides the computer system 2600 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.

[00590] FIG. 40 shows a block diagram of an example embodiment of a device 2700 that can be utilized to implement various portions of the presently disclosed technology. The mobile device 2700 can be a laptop, a smartphone, a tablet, a camcorder, or other types of devices that are capable of processing videos. The mobile device 2700 includes a processor or controller 2701 to process data, and memory 2702 in communication with the processor 2701 to store and/or buffer data. For example, the processor 2701 can include a central processing unit (CPU) or a microcontroller unit (MCU). In some implementations, the processor 2701 can include a field-programmable gate-array (FPGA). In some implementations, the mobile device 2700 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 2702 can include and store processor-executable code, which when executed by the processor 2701, configures the mobile device 2700 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. To support various functions of the mobile device 2700, the memory 2702 can store information and data, such as instructions, software, values, images, and other data processed or referenced by the processor 2701. 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 2702. In some implementations, the mobile device 2700 includes an input/output (I/O) unit 2703 to interface the processor 2701 and/or memory 2702 to other modules, units or devices. For example, the I/O unit 2703 can interface the processor 2701 and memory 2702 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 2700 can interface with other devices using a wired connection via the I/O unit 2703. The mobile device 2700 can also interface with other external interfaces, such as data storage, and/or visual or audio display devices 2704, 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 2704 or an external device. For example, the display device 2704 can display a video frame modified based on the MVPs in accordance with the disclosed technology.

[00591] FIG. 41 is a flowchart representation of a method 4100 of video processing. The method 4100 includes determining (4102), in a conversion between a video block and a bitstream representation of the video block, that a video block is a boundary block of a coding tree unit (CTU) in which the video block is positioned and therefore local illumination compensation (LIC) coding tool is enabled for the video block, deriving (4104), based on the determining that LIC coding tool is enabled for the video block, parameters for local illumination compensation (LIC) of the video block, and performing (4106) the conversion by adjusting pixel values of the video block using LIC. For example, item 1 of Section 5 discloses some example variations and embodiments of the method 4100.

[00592] The following clause-based listing described certain features and aspects of the disclosed techniques listed in Section 5.

[00593] 1. A video processing method, comprising:

[00594] determining, in a conversion between a video block and a bitstream representation of the video block, that a video block is a boundary block of a coding tree unit (CTU) in which the video block is positioned and therefore local illumination compensation (LIC) coding tool is enabled for the video block;

[00595] deriving, based on the determining that LIC coding tool is enabled for the video block, parameters for local illumination compensation (LIC) of the video block; and

[00596] performing the conversion by adjusting pixel values of the video block using

LIC.

[00597] 2. The method of clause 1, wherein the conversion generates the video block from the bitstream representation.

[00598] 3. The method of clause 1, wherein the conversion generates the bitstream representation from the video block.

[00599] 4. The method of any of clauses 1 to 3, wherein the deriving uses samples of neighboring blocks depending on a position of the current block within the CTU.

[00600] 5. The method of clause 4, wherein (1) in case that the position of the current block is at a left boundary of the CTU, the deriving uses only left reconstructed neighboring samples, or (2) in case that the position of the current block is a top boundary of the CTU, the deriving uses only above reconstructed neighboring samples, or (3) in case that the position of the current block is at a top-left corner of the CTU, the deriving uses left and/or above reconstructed neighboring samples, or (4) in case that the position of the current block is at a top-right corner of the CTU, the deriving uses right and/or above reconstructed neighboring samples, or (5) in case that the position of the current block is at a bottom-left corner of the CTU, the deriving uses left and/or below reconstructed neighboring samples.

[00601] 6. A video processing method, comprising:

[00602] determining, in a conversion between a video block and a bitstream representation of the video block, that a video block is an inner block of a coding tree unit (CTU) in which the video block is positioned and therefore local illumination compensation (LIC) coding tool is disabled for the video block;

[00603] inheriting parameters for LIC of the video block; and

[00604] performing the conversion by adjusting pixel values of the video block using LIC.

[00605] 7. The method of clause 6, wherein the inheriting includes:

[00606] maintaining a number of look-up tables of LIC parameters previously used for other blocks; and

[00607] deriving LIC parameters for the video block based on one or more LIC parameters from the look-up table.

[00608] 8. The method of clause 7, wherein the other blocks includes blocks from the CTU or blocks from a reference picture used for the current block.

[00609] 9. The method of any of clauses 7 to 9, wherein the look-table is emptied and rebuilt for each CTU or a row of CTU or a slice or a tile or a tile group or a picture of the video block.

[00610] 10. The method of clause 7, wherein the current block uses an advanced motion vector prediction (AMVP) mode or an affine inter mode then the bitstream representation is configured to include a flag indicating which LIC parameters are used for the current block.

[00611] 11. The method of clause 10, wherein the flag is indicative of an index to an entry in the LUT.

[00612] 12. The method of any of clauses 7 to 11, wherein multiple LUTs are used.

[00613] 13. The method of clause 12, wherein at least one LUT is maintained for each reference picture or for each reference picture list used in the conversion of the current block.

[00614] 14. The method of clause 7, wherein the current block uses a merge mode or affine merge mode for spatial or temporal merge candidate motion vectors, and wherein the inheriting includes inheriting one or both of a neigboring' block's LIC flag and LIC parameters.

[00615] 15. The method of clause 7, wherein the current block uses a merge mode or affine merge mode for spatial or temporal merge candidate motion vectors, and wherein the inheriting includes configuring the bitstream with one or more difference values and computing LIC flag or LIC parameters based on the one or more difference values and one or both of a neigboring' block's LIC flag and LIC parameters.

[00616] 16. The method of clause 1, wherein the current block is encoded using an advanced motion vector prediction (AMVP) or an affine inter mode, and wherein the parameters are derived using a least square error minimization in which subsamples set of neigbouring samples is used for error minimization.

[00617] 17. A video processing method, comprising:

[00618] determining, in a conversion between a video block and a bitstream

representation of the video block, that both local illumination compensation and intra block copy coding tools are enabled for use by the current block; and

[00619] performing the conversion by performing local illumination compensation (LIC) and intra block copy operations on the video block.

[00620] 18. The method of clause 17, wherein an LIC flag in the bitstream is configured to indicate enablement of LIC for the current block.

[00621] 19. The method of clause 18, wherein the conversion generates the video block from the bitstream representation.

[00622] 20. The method of clause 18, wherein the conversion generates the bitstream representation from the video block.

[00623] 21. The method of clause 17, wherein the current block uses merge mode and wherein the current block inherits LIC flag value of a neighboring block.

[00624] 22. The method of clause 17, wherein the current block uses merge mode and wherein the current block inherits LIC parameters of a neighboring block.

[00625] 23. A video processing method, comprising:

[00626] determining, during a conversion between a bitstream of a video comprising multiple pictures having multiple blocks, a local illumination compensation (LIC) mode for a current block of the video based on an LIC mode rule; and

[00627] performing the conversion between the current block and a corresponding bitstream representation of the current block.

[00628] 24. The method of clause 23, wherein the LIC mode rule specifies to use the LIC mode exclusive of inter prediction, a diffusion filter, a bilateral filter, overlapped block motion compensation or, a tool that modifies an inter prediction signal for the current block.

[00629] 25. The method of clause 24, wherein the LIC mode rule is explicitly signaled in the bitstream.

[00630] 26. The method of clause 23, wherein the LIC mode rule specifies to use the LIC mode only when the current block does not use a generalized bi-prediction (GBI) coding mode.

[00631] 27. The method of any of clauses 23 or 26, wherein the LIC mode rule is

explicitly signaled in the bitstream and a GBI indication is omitted from the bitstream.

[00632] 28. The method of clause 23, wherein the LIC mode rule specifies to use the LIC mode only when the current block does not use a multi-hypothesis coding mode.

[00633] 29. The method of any of clauses 23 or 28, wherein the LIC mode rule is explicitly signaled in the bitstream and the multi-hypothesis coding mode is implicitly disabled.

[00634] 30. The method of any of clauses 28 to 29, wherein LIC mode rule specifies to apply LIC to a selected subset of prediction signals of the multi-hypothesis mode.

[00635] 31. The method of clause 23, wherein the LIC mode rule specifies to inherit an LIC flag from a base merge candidate for the current block that is also using an ultimate motion vector expression (UMVE) mode.

[00636] 32. The method of clause 31, wherein the current block inherits its LIC flag from a base merge candidate of the UMVE mode.

[00637] 33. The method of clause 32, wherein the current block inherits using a calculation in Section 2.2.7.

[00638] 34. The method of any of clauses 1 to 33 wherein LIC is used only when the current block also meets a condition related to size or a slice type or a tile type or a picture type of the current block or a type of motion information associated with the current block.

[00639] 35. The method of clause 34, wherein the condition excludes block sizes smaller than M*H samples, where M and H are pre-specified integer values.

[00640] 36. The method of clause 35, wherein M*H is equal to 16 or 32 or 64.

[00641] 37. The method of clause 34, wherein the condition specifies that the current block has a width or a height that is smaller than or no larger than X, where X is an integer.

[00642] 38. The method of clause 37, wherein X = 8.

[00643] 39. The method of clause 34, wherein the condition specifies that the current block correspond to luma samples.

[00644] 40. The method of clause 34, wherein the condition specifies that the current block correspond to chroma samples.

[00645] 41 . A video processing method, comprising:

[00646] determining, during a conversion between a video block of a video and a bitstream representation of the video, local illumination compensation (LIC) parameters for the video block using at least some samples of neighboring blocks of the video block; and

[00647] performing the conversion between the video block and the bitstream representation by performing LIC using the determined parameters.

[00648] 42. The method of clause 41, wherein the conversion generates the video block from the bitstream representation.

[00649] 43. The method of clause 41, wherein the conversion generates the bitstream representation from the video block.

[00650] 44. The method of any of clauses 41 to 43, wherein identities the at least some samples of neighboring block depend on code information of the current block or a shape of the current block.

[00651] 45. The method of clause 44, wherein, a width of the current block is greater than or equal to a height of the current block, and therefore only above neighboring pixels are used for deriving LIC parameters.

[00652] 46. A video processing method, comprising:

[00653] performing a conversion between a video and a bitstream representation of the video, wherein the video is represented as video frames comprising video blocks, and local illumination compensation (LIC) is enabled only for video blocks that use a geometric prediction structure including a triangular prediction mode.

[00654] 47. The method of clause 46, wherein the conversion generates the video block from the bitstream representation.

[00655] 48. The method of clause 46, wherein the conversion generates the bitstream representation from the video block.

[00656] 49. The method of any of clauses 46 to 48, wherein, for a current block for which LIC is enabled, all prediction unit partitions share a same value of LIC flag.

[00657] 50. The method of any of clauses 46 to 48, wherein, for a current block that is partitioned from a top-right corner to a bottom-left corner, a single set of LIC parameters

is used for both partitions of the current block.

[00658] 51. The method of any of clauses 46 to 48, wherein, for a current block that is partitioned from a top-left corner to a bottom-right corner, a single set of LIC parameters is used for each partition of the current block.

[00659] 52. A video processing method, comprising:

[00660] performing a conversion between a video and a bitstream representation of the video, wherein the video is represented as video frames comprising video blocks, and local illumination compensation (LIC) is implemented for less than all pixels of a current block in the conversion to its corresponding bitstream representation.

[00661] 53. The method of clause 52, wherein the conversion generates the video block from the bitstream representation.

[00662] 54. The method of clause 52, wherein the conversion generates the bitstream representation from the video block.

[00663] 55. The method of any of clauses 52 to 54, wherein LIC is performed only for pixels on a boundary of the current block.

[00664] 56. The method of any of clauses 52 to 54, wherein LIC is performed only for pixels in a top portion of the current block.

[00665] 57. The method of any of clauses 52 to 54, wherein LIC is performed only for pixels in a right-side portion of the current block.

[00666] 58. A method for video processing, comprising:

[00667] determining, in a conversion between a video block and a bitstream representation of the video block, that both local illumination compensation (LIC) and either generalized bi-prediction (GBi) or multi-hypothesis inter prediction coding tools are enabled for use by the current block; and

[00668] performing the conversion by performing LIC and either GBi or multi-hypothesis inter prediction operations on the video block.

[00669] 59. The method of clause 58, wherein information associated with the GBi coding tool is signaled after information associated with the LIC coding tool.

[00670] 60. The method of clause 58, wherein information associated with the LIC coding tool is signaled after information associated with the GBi coding tool.

[00671] 61. The method of clause 59 or 60, wherein the information is signaled in a sequence parameter set (SPS), a picture parameter set (PPS), a slice header, a tile group header, a tile, a coding unit (CU) or a coding tree unit (CTU).

[00672] 62. The method of clause 58, wherein an LIC flag is true, and wherein the bitstream representation excludes one or more syntax elements associated with the GBi or multi-hypothesis inter prediction coding tools.

[00673] 63. The method of clause 58, wherein a GBi flag is true, and wherein the bitstream representation excludes one or more syntax elements associated with the LIC or multi-hypothesis inter prediction coding tools.

[00674] 64. The method of clause 58, wherein a multi-hypothesis inter prediction LIC flag is true, and wherein the bitstream representation excludes one or more syntax elements associated with the LIC or GBi coding tools.

[00675] 65. A method for video processing, comprising:

[00676] determining, in a conversion between a video block and a bitstream representation of the video block, that both local illumination compensation (LIC) and combined inter-intra prediction (CIIP) coding tools are enabled for use by the current block; and

[00677] performing the conversion by performing LIC and CIIP operations on the video block.

[00678] 66. The method of clause 65, wherein information associated with the CIIP coding tool is signaled after information associated with the LIC coding tool.

[00679] 67. The method of clause 65, wherein information associated with the LIC coding tool is signaled after information associated with the CIIP coding tool.

[00680] 68. The method of clause 66 or 67, wherein the information is signaled in a sequence parameter set (SPS), a picture parameter set (PPS), a slice header, a tile group header, a tile, a coding unit (CU) or a coding tree unit (CTU).

[00681] 69. The method of clause 65, wherein an LIC flag is true, and wherein the bitstream representation excludes one or more syntax elements associated with the CIIP coding tool.

[00682] 70. The method of clause 65, wherein a CIIP flag is true, and wherein the

bitstream representation excludes one or more syntax elements associated with the LIC coding tool.

[00683] 71. The method of any of clauses 1 to 70 in which the bitstream is configured with a bit field controlling an operation of the conversion.

[00684] 72. The method of clause 71, wherein the bit field is includes at a sequence level, a picture level, a video level, a sequence header level, a picture header level, a slice header level, a tile level or a group of coding tree unit level.

[00685] 73. A video encoding apparatus comprising a processor configured to implement a method recited in any one or more of clauses 1 to 72.

[00686] 74. A video decoding apparatus comprising a processor configured to implement a method recited in any one or more of clauses 1 to 72.

[00687] 75. A computer-readable medium having code stored thereon, the code, when executed, causing a processor to implement a method recited in any one or more of clauses 1 to 72.

[00688] FIG. 42 is a flowchart representation of a method 4200 of video processing. The method 4200 includes generating (4210), for a conversion between a current block of a video and a bitstream representation of the video, a motion vector for the current block. The method 4200 includes invoking (4220) a weighted prediction processing tool or a second processing tool in a mutually exclusive manner. Here, invoking the second processing tool comprises refining the motion vector. The method 4200 also includes performing (4230) the conversion according to the invoking.

[00689] In some embodiments, the conversion generates the current block from the bitstream representation. In some embodiments, wherein the conversion generates the bitstream representation from the current block.

[00690] In some embodiments, invoking the weighted prediction processing tool comprises applying a linear weight and an offset to derive a final prediction.

[00691] In some embodiments, the invoking comprises determining, in case the weighted prediction processing tool is disabled for the current block, that the second processing tool is enabled for the current block. In some embodiments, the invoking comprises determining, in case the weighted prediction processing tool is enabled for

the current block, that the second processing tool is disabled for the current block. In some embodiments, the invoking comprises determining, in case the second processing tool is disabled for the current block, that weighted prediction processing tool is enabled for the current block. In some embodiments, the invoking comprises determining, in case the second processing tool is enabled for the current block, that weighted prediction processing tool is disabled for the current block.

[00692] In some embodiments, refining the motion vector comprises applying a motion vector offset to refine the motion vector. The motion vector offset is determined based on a sum of absolute differences derived from different reference lists of the current block. In some embodiments, the sum of absolute differences is calculated based on partial positions derived from different reference lists of the current block. In some embodiments, the second processing tool comprises a decoder side motion vector refinement (DMVR) process.

[00693] In some embodiments, the invoking further comprises invoking the weighted prediction processing tool or a third processing tool in a mutually exclusive manner. The third processing tool applies a prediction offset to refine the final prediction. The prediction offset is determined based on at least one of a horizontal or a vertical gradient derived from different reference lists of the current block. In some embodiments, the third processing tool comprises a bi-directional optical flow prediction (BDOF) process.

[00694] In some embodiments, the invoking further comprises invoking a weighted prediction processing tool or a fourth processing tool in a mutually exclusive manner. The fourth processing tool comprises one of: a combined inter-intra prediction (CIIP) processing tool that generates the final prediction based on an inter prediction and an intra prediction, an affine processing tool that generates the final prediction based on a control point motion vector, or an overlapped block motion compensation (OBMC) processing tool that generates the final prediction based on motion vectors of a sub-block and spatial neighboring sub-block.

[00695] In some embodiments, in case the weighted prediction processing tool is applied to the current block, information indicating whether the second processing tool

is applied is defaulted to have a value of zero. In some embodiments, in case the weighted prediction processing tool is applied to the current block, information indicating whether the second processing tool is applied is signaled in a sequence parameter set (SPS), a picture parameter set (PPS), a slice header, a tile group header, a tile, a coding unit (CU), or a coding tree unit (CTU).

[00696] 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.

[00697] 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.

[00698] 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).

[00699] 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.

[00700] 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.

[00701] 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.

[00702] 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

1. A video processing method, comprising:

generating, for a conversion between a current block of a video and a bitstream representation of the video, a motion vector for the current block;

invoking a weighted prediction processing tool or a second processing tool in a mutually exclusive manner, wherein the invoking the second processing tool comprises refining the motion vector; and

performing the conversion according to the invoking.

2. The method of claim 1, wherein the conversion generates the current block from the bitstream representation.

3. The method of claim 1, wherein the conversion generates the bitstream representation from the current block.

4. The method of claim 1, wherein the invoking the weighted prediction processing tool comprises applying a linear weight and an offset to derive a final prediction.

5. The method of any one or more of claims 1 to 4, wherein the invoking comprises:

determining, in case the weighted prediction processing tool is disabled for the current block, that the second processing tool is enabled for the current block.

6. The method of any one or more of claims 1 to 4, wherein the invoking comprises:

determining, in case the weighted prediction processing tool is enabled for the current block, that the second processing tool is disabled for the current block.

7. The method of any one or more of claims 1 to 4, wherein the invoking comprises:

determining, in case the second processing tool is disabled for the current block, that weighted prediction processing tool is enabled for the current block.

8. The method of any one or more of claims 1 to 4, wherein the invoking comprises:

determining, in case the second processing tool is enabled for the current block, that weighted prediction processing tool is disabled for the current block.

9. The method of any one of claims 1 to 8, wherein refining the motion vector comprises applying a motion vector offset to refine the motion vector, wherein the motion vector offset is determined based on a sum of absolute differences derived from different reference lists of the current block.

10. The method of claim 9, wherein the sum of absolute differences is calculated based on partial positions derived from different reference lists of the current block.

11. The method of claim 9, wherein the second processing tool comprises a decoder side motion vector refinement (DMVR) process.

12. The method of any one of claims 1 to 11, wherein the invoking further comprises invoking the weighted prediction processing tool or a third processing tool in a mutually exclusive manner, wherein the third processing tool applies a prediction offset to refine the final prediction, wherein the prediction offset is determined based on at least one of a horizontal or a vertical gradient derived from different reference lists of the current block.

13. The method of claim 12, wherein the third processing tool comprises a bi-directional optical flow prediction (BDOF) process.

14. The method of any one of claims 1 to 11, wherein the invoking further comprises invoking a weighted prediction processing tool or a fourth processing tool in a mutually exclusive manner, the fourth processing tool comprising one of: a combined inter-intra prediction (CIIP) processing tool that generates the final prediction based on an inter prediction and an intra prediction, an affine processing tool that generates the final prediction based on a control point motion vector, or an overlapped block motion

compensation (OBMC) processing tool that generates the final prediction based on motion vectors of a sub-block and spatial neighboring sub-block.

15. The method of any one or more of claims 1 to 14, wherein, in case the weighted prediction processing tool is applied to the current block, information indicating whether the second processing tool is applied is defaulted to have a value of zero.

16. The method of any one or more of claims 1 to 15, wherein, in case the weighted prediction processing tool is applied to the current block, information indicating whether the second processing tool is applied is signaled in a sequence parameter set (SPS), a picture parameter set (PPS), a slice header, a tile group header, a tile, a coding unit (CU), or a coding tree unit (CTU).

17. A video processing apparatus comprising a processor configured to implement a method recited in any one or more of claims 1 to 16.

18. A computer-readable medium having code stored thereon, the code, when executed, causing a processor to implement a method recited in any one or more of claims 1 to 16.

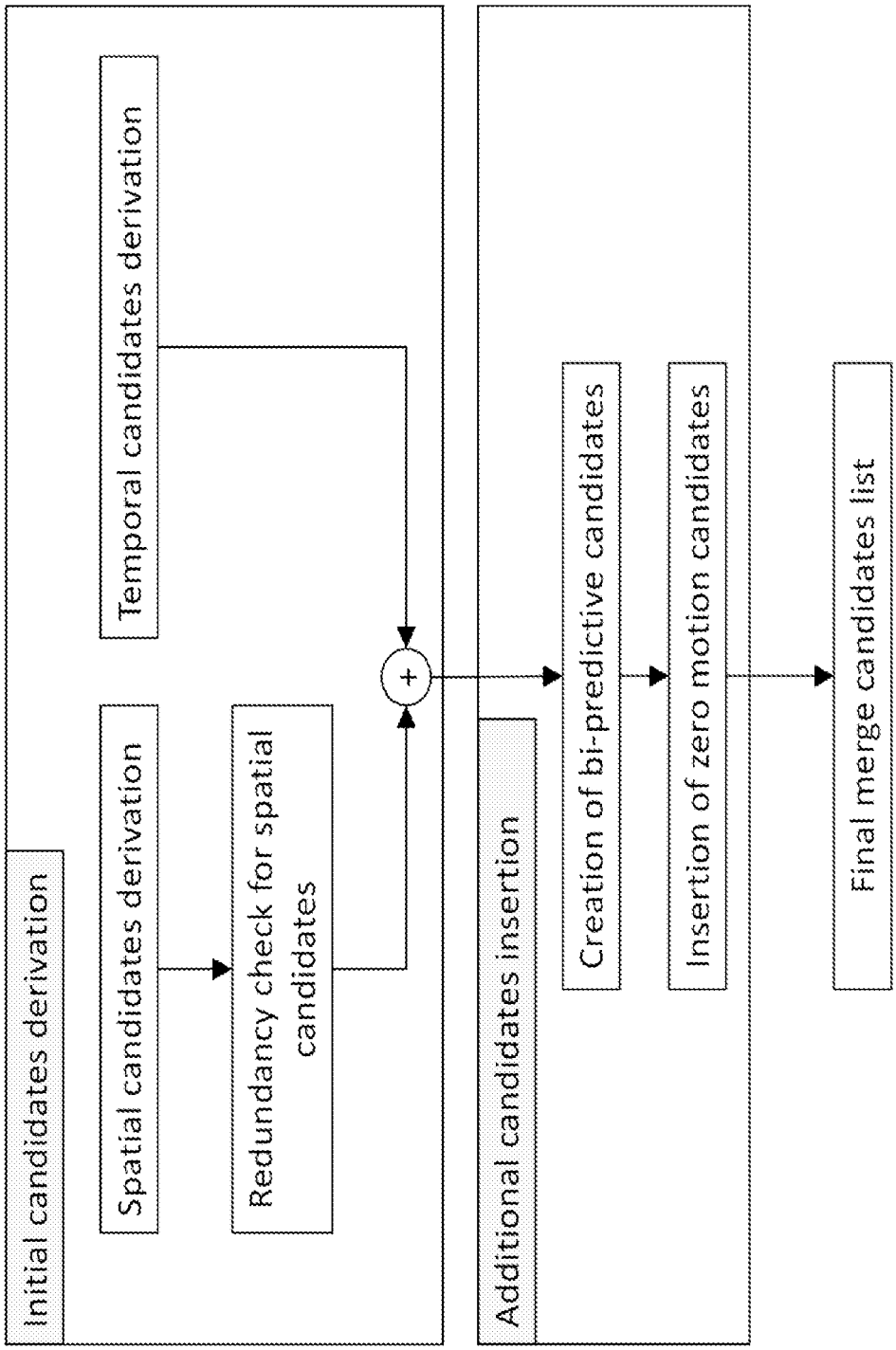

Initial candidates derivation
Spatial candidates derivation
Redundancy check for spatial candidates
Temporal candidates derivation
+
Additional candidates insertion
Creation of bi-predictive candidates
Insertion of zero motion candidates
Final merge candidates list

FIG. 1

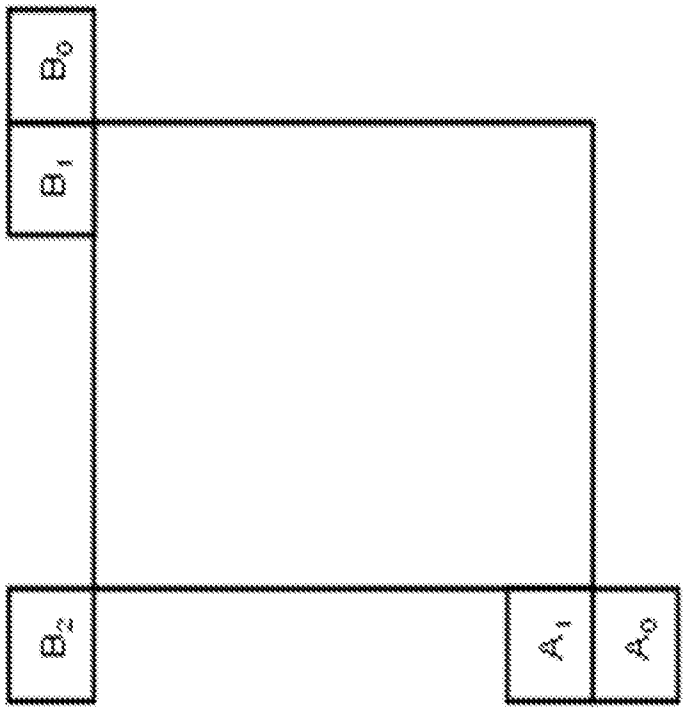


FIG. 2

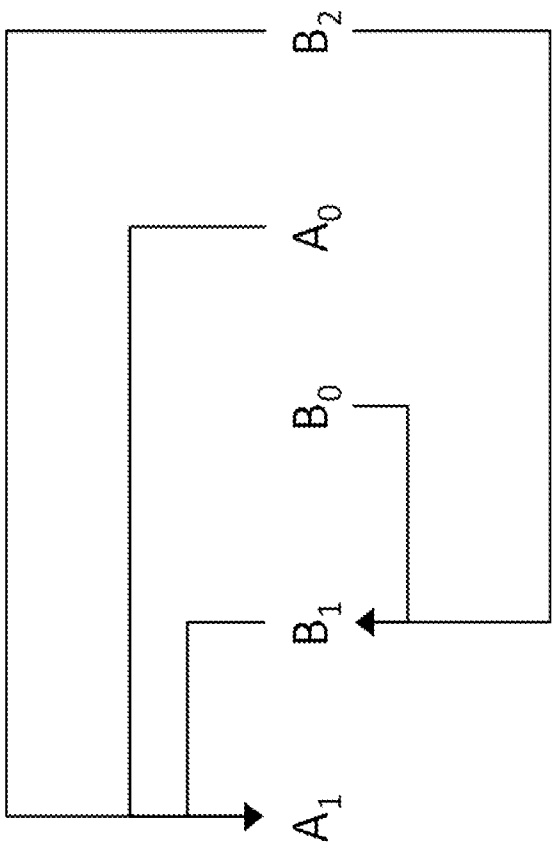


FIG. 3

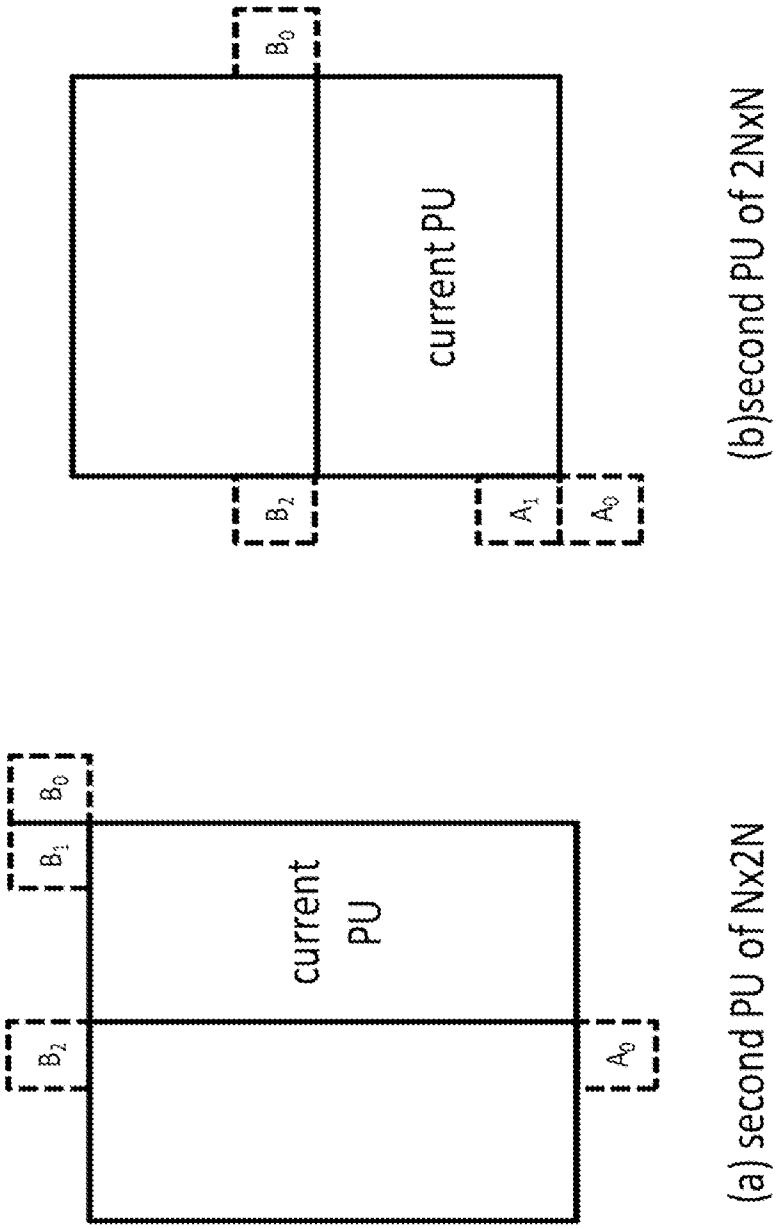

B0
B1
B2
current PU
A0
(a) second PU of Nx2N
B0
B2
current PU
A1
A0
(b)second PU of 2NxN

FIG. 4

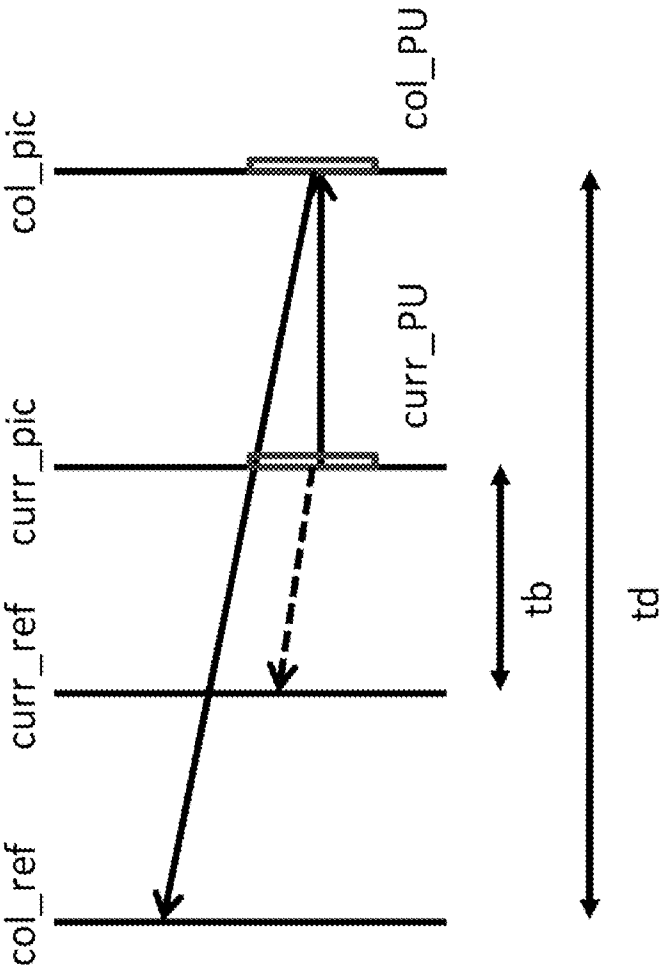


FIG. 5

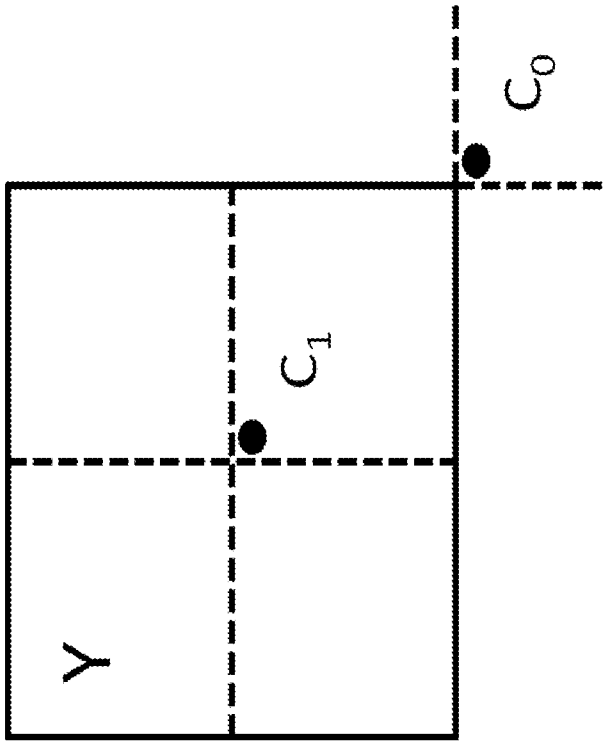


FIG. 6

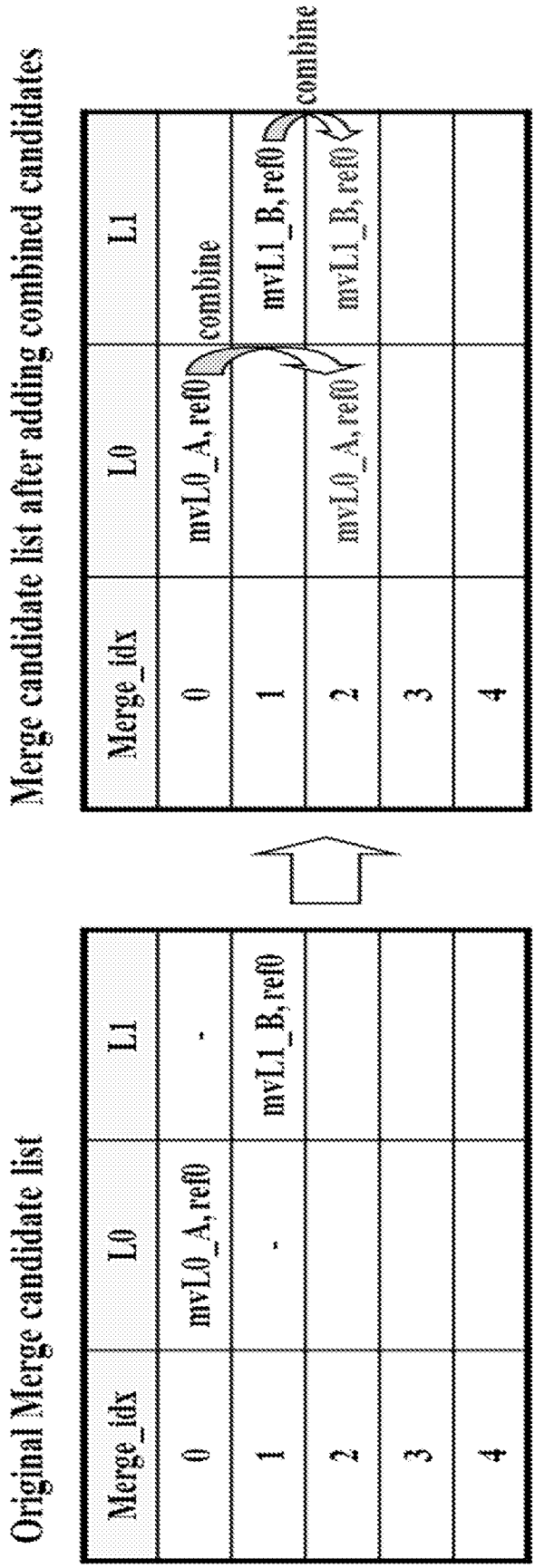

Original Merge candidate list
Merge_idx
L0
L1
0
mvL0_A, ref0
-
1
-
mvL1_B, ref0
2
3
4
Merge candidate list after adding combined candidates
Merge_idx
L0
L1
0
mvL0_A, ref0
combine
1
mvL1_B, ref0
combine
2
mvL0_A, ref0
mvL1_B, ref0
3
4

FIG. 7

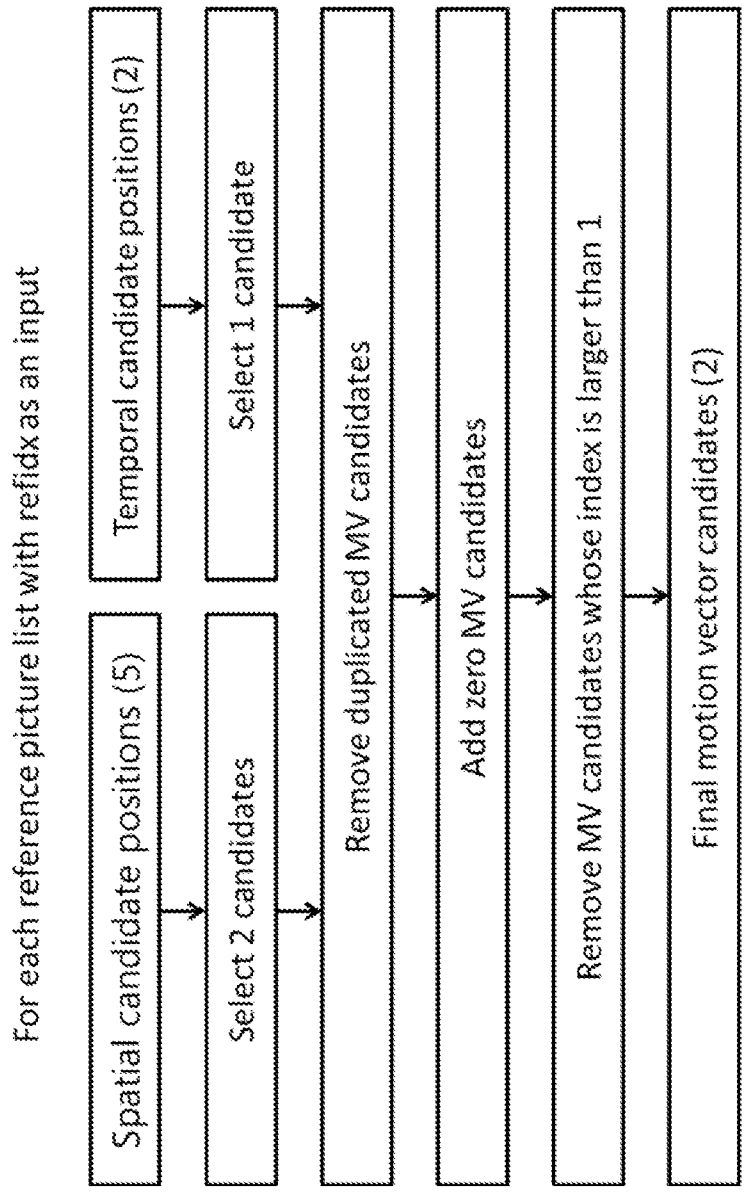

For each reference picture list with refidx as an input
Spatial candidate positions (5)
Select 2 candidates
Temporal candidate positions (2)
Select 1 candidate
Remove duplicated MV candidates
Add zero MV candidates
Remove MV candidates whose index is larger than 1
Final motion vector candidates (2)

FIG. 8

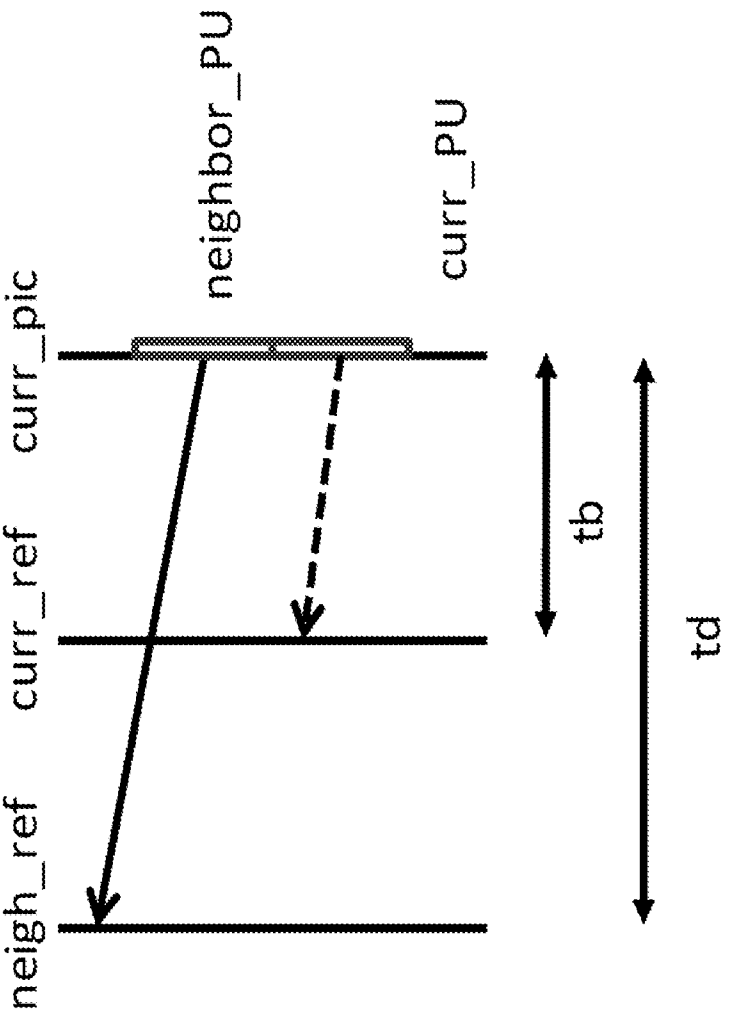


FIG. 9

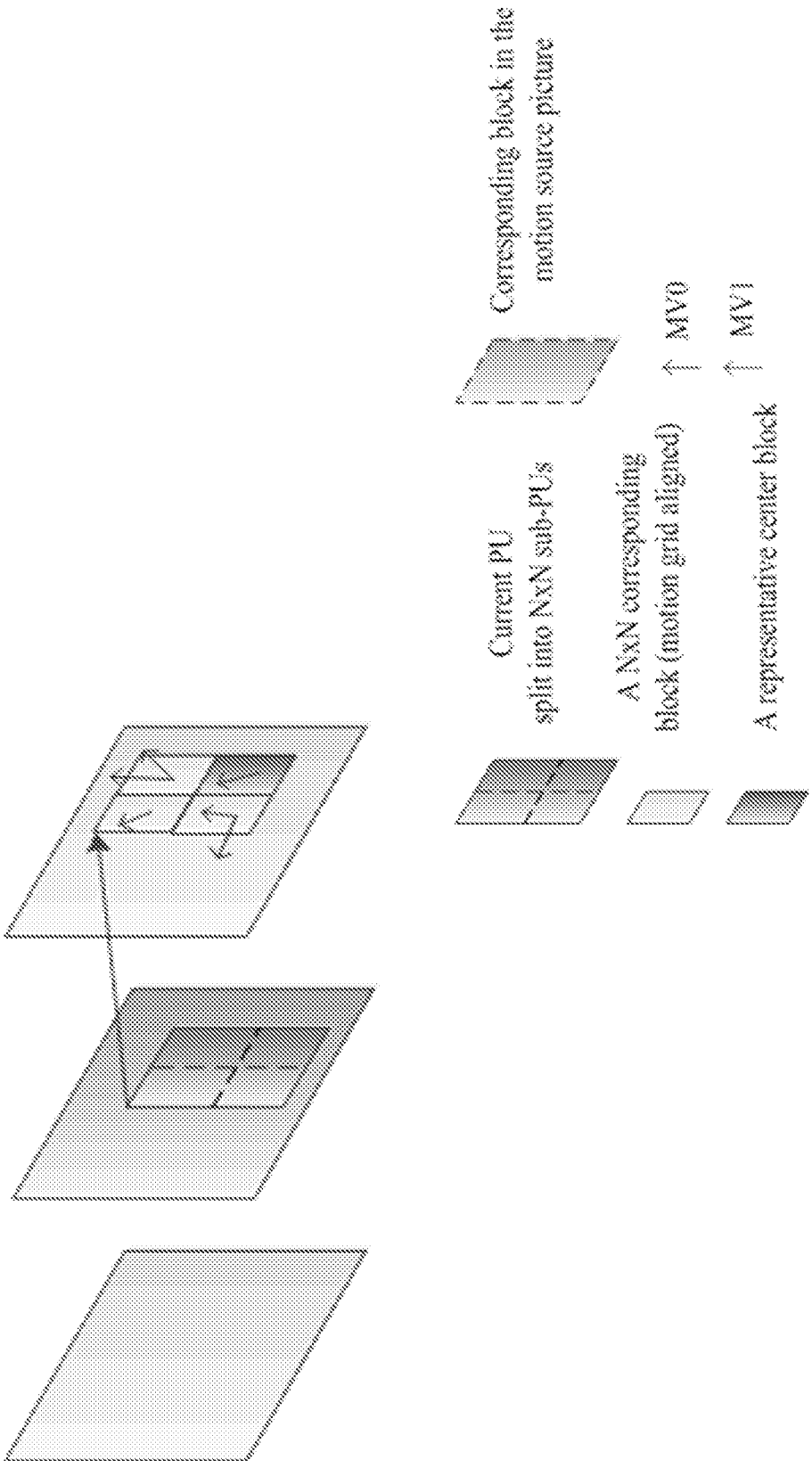

Current PU
split into NxN sub-PUs
A NxN corresponding
block (motion grid aligned)
A representative center block
MV0
MV1
Corresponding block in the
motion source picture

FIG. 10

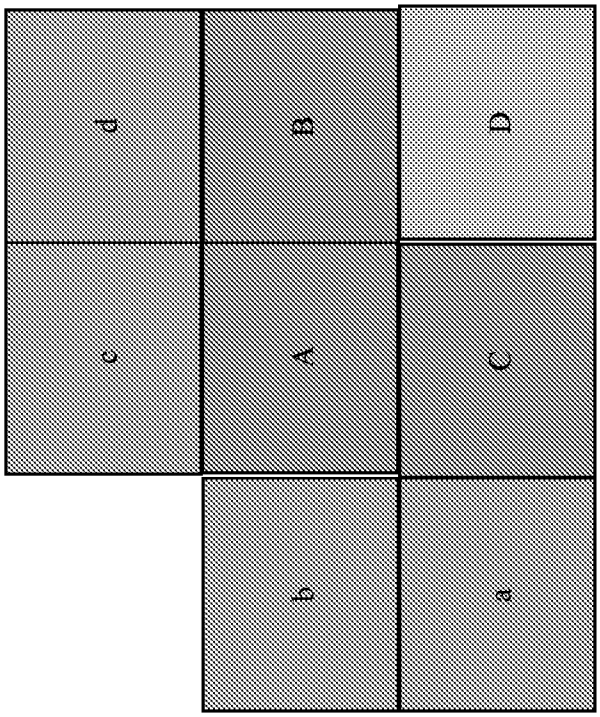


FIG. 11

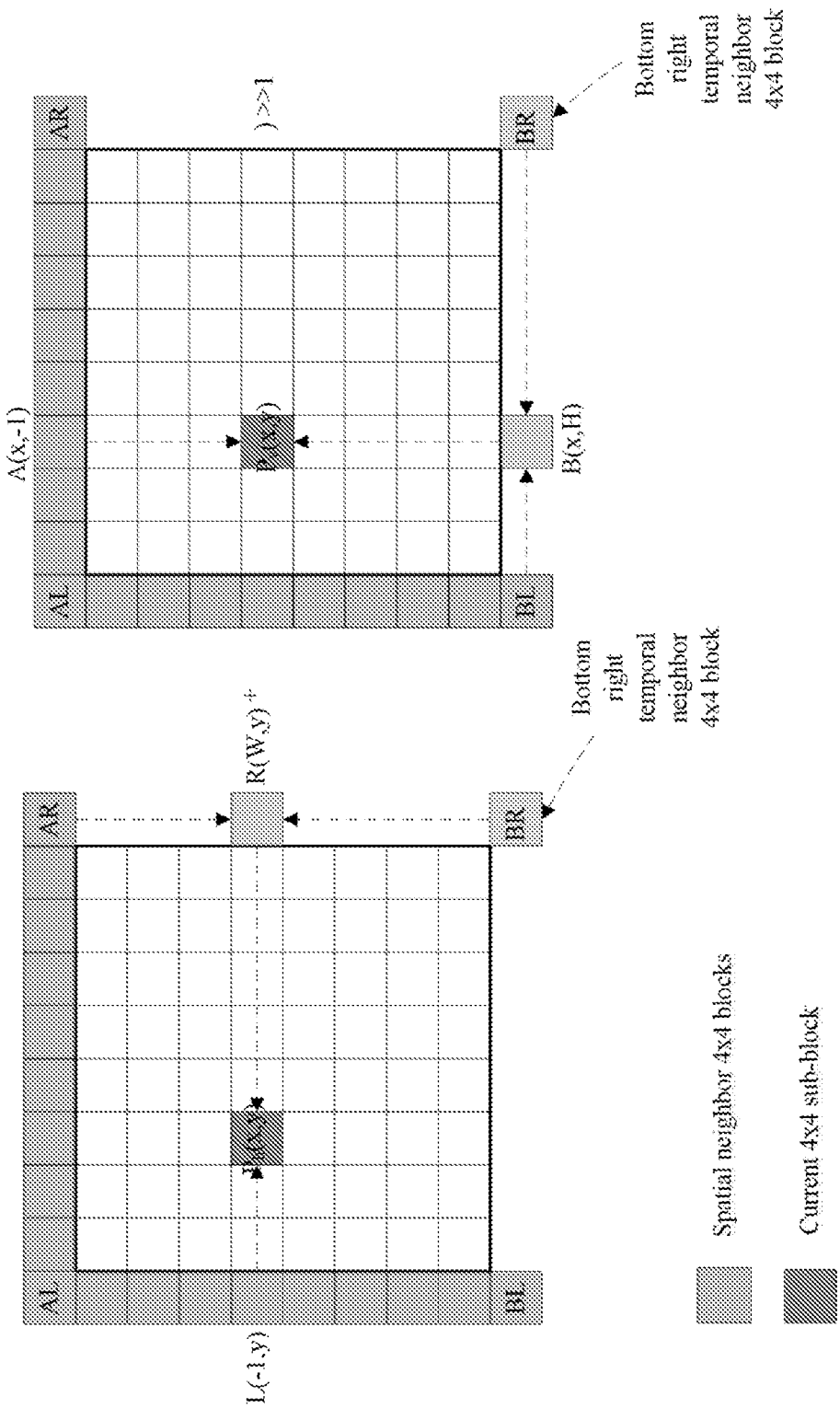

AR
BR
AL
BL
A(x,-1)
B(x,H)
L(-1,y)
R(W,y)
Bottom right temporal neighbor 4x4 block
Bottom right temporal neighbor 4x4 block
Spatial neighbor 4x4 blocks
Current 4x4 sub-block

FIG. 12

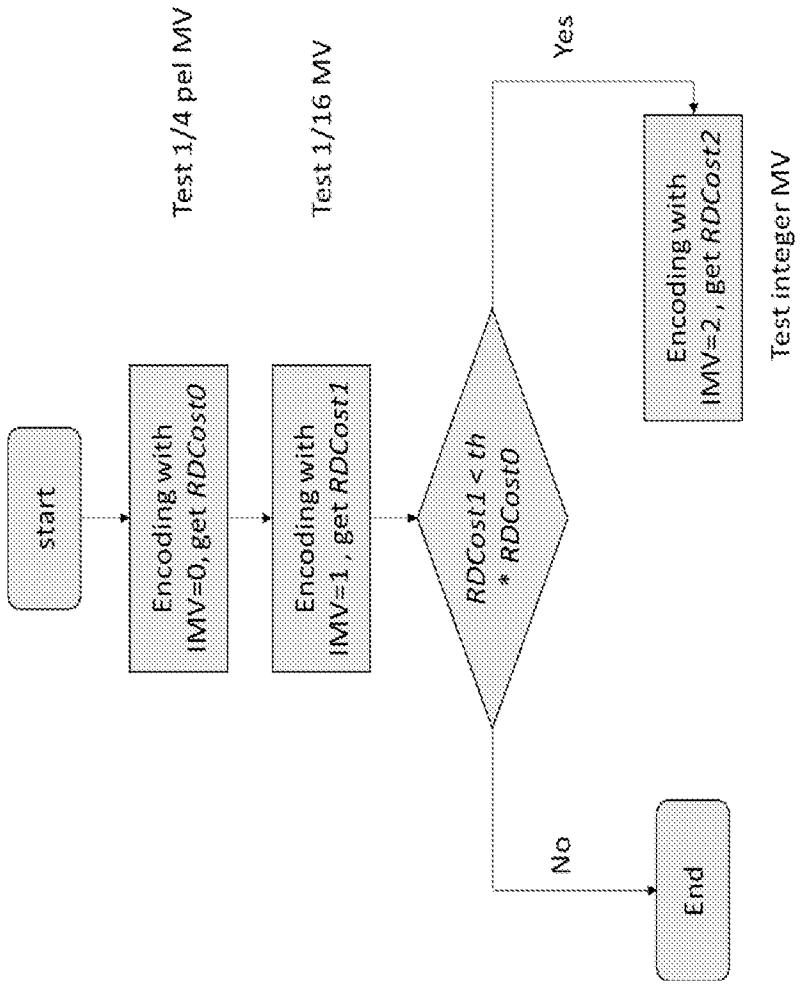

start
Encoding with IMV=0, get RDCost0
Test 1/4 pel MV
Encoding with IMV=1 , get RDCost1
Test 1/16 MV
RDCost1 < th * RDCost0
Yes
Encoding with IMV=2 , get RDCost2
Test integer MV
No
End

FIG. 13

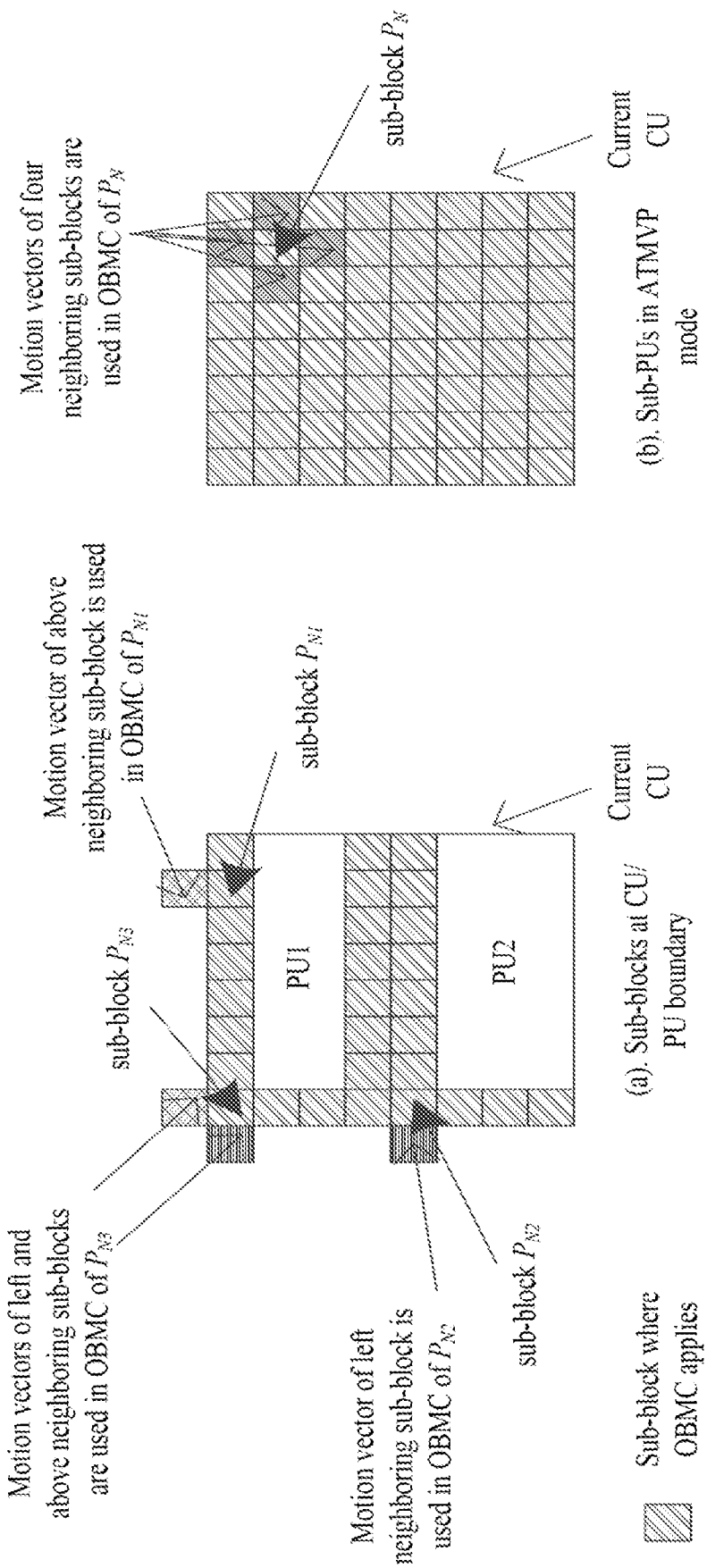

Motion vectors of four neighboring sub-blocks are used in OBMC of P_N
sub-block P_N
Current CU
(b). Sub-PUs in ATMVP mode
Motion vector of above neighboring sub-block is used in OBMC of P_{N1}
sub-block P_{N1}
Motion vectors of left and above neighboring sub-blocks are used in OBMC of P_{N3}
sub-block P_{N3}
PU1
PU2
Current CU
(a). Sub-blocks at CU/PU boundary
Motion vector of left neighboring sub-block is used in OBMC of P_{N2}
sub-block P_{N2}
Sub-block where OBMC applies

FIG. 14

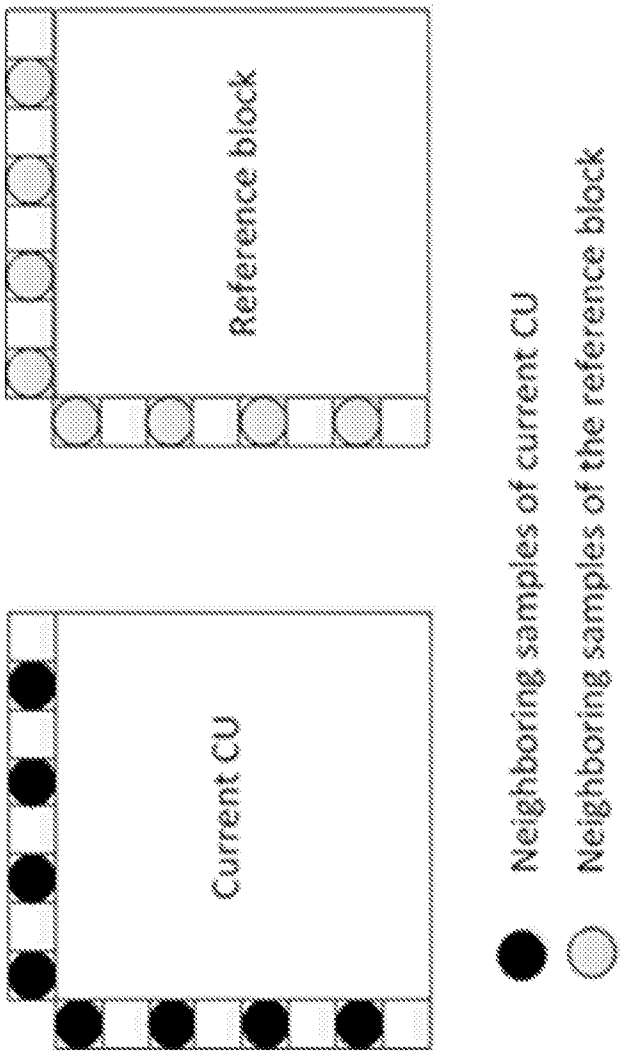

Reference block
Current CU
Neighboring samples of current CU
Neighboring samples of the reference block

FIG. 15

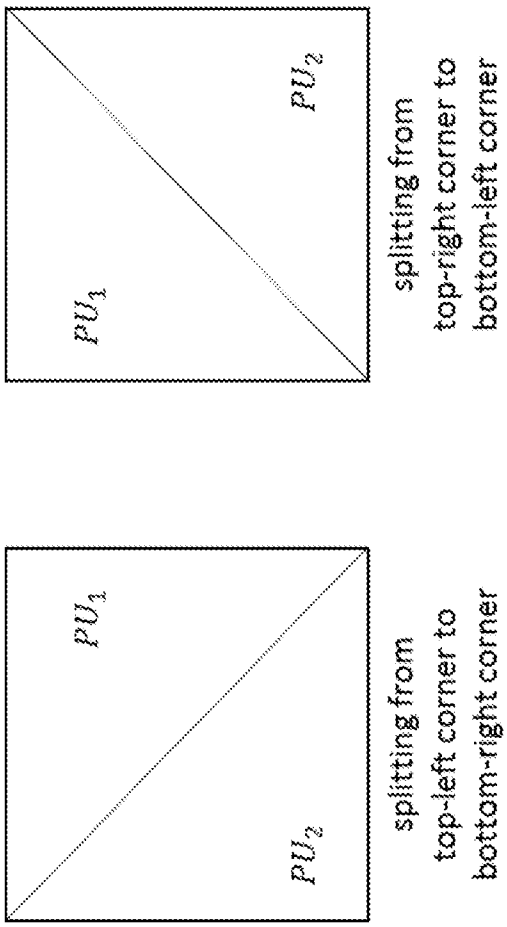

PU_1
PU_2
splitting from top-left corner to bottom-right corner
PU_1
PU_2
splitting from top-right corner to bottom-left corner

FIG. 16

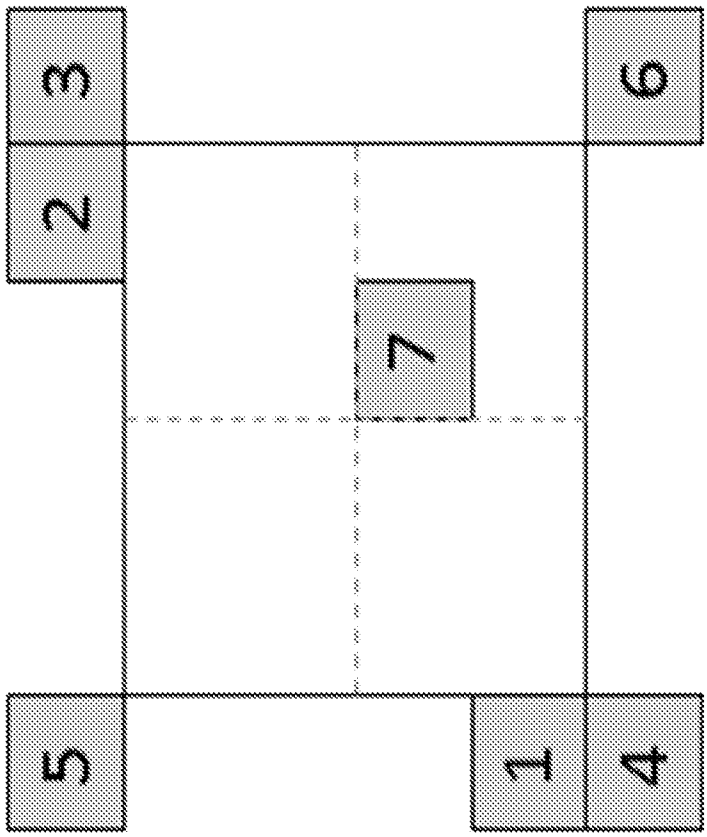


FIG. 17

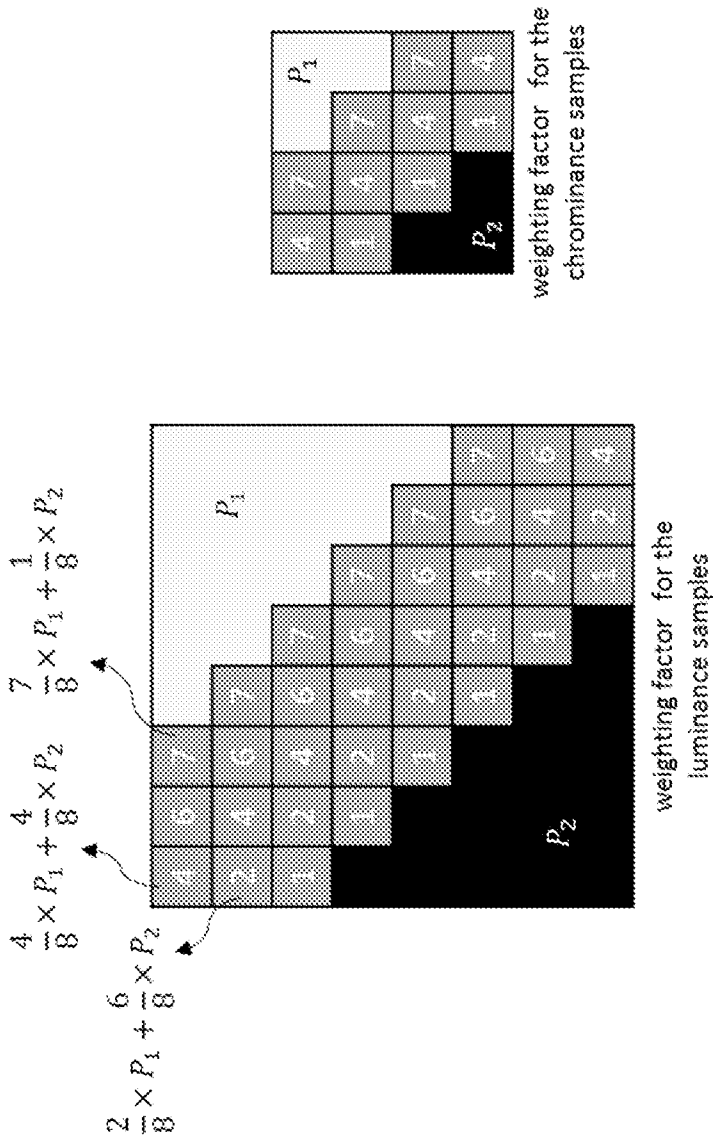

P_1
P_2
weighting factor for the chrominance samples
P_1
P_2
weighting factor for the luminance samples
$\frac{2}{8} \times P_1 + \frac{6}{8} \times P_2$
$\frac{4}{8} \times P_1 + \frac{4}{8} \times P_2$
$\frac{7}{8} \times P_1 + \frac{1}{8} \times P_2$

FIG. 18

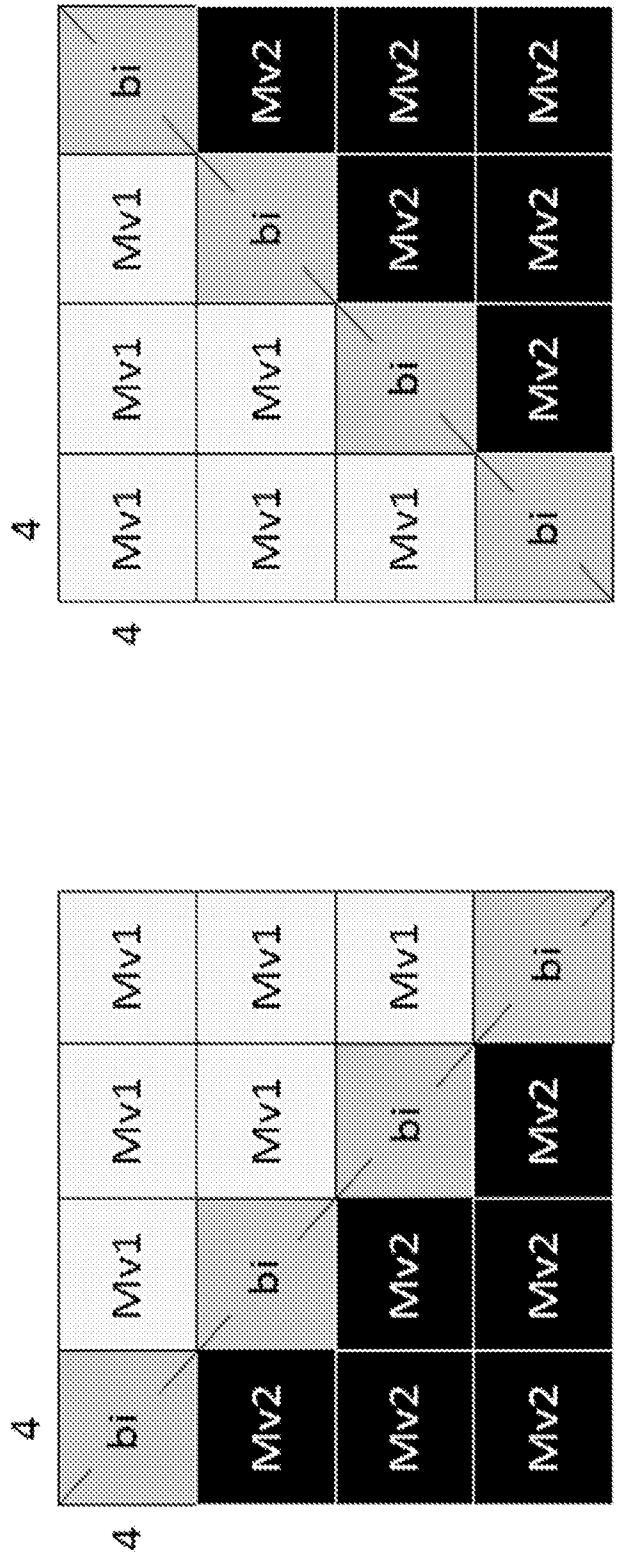

4
4
Mv1
Mv1
Mv1
bi
Mv1
bi
Mv2
Mv2
Mv1
Mv1
bi
Mv2
Mv2
Mv1
Mv1
bi
Mv2
Mv2
4
4
bi
Mv2
Mv2
Mv2
Mv1
bi
Mv2
Mv2
Mv1
Mv1
bi
Mv2
Mv2
Mv1
Mv1
bi

FIG. 19

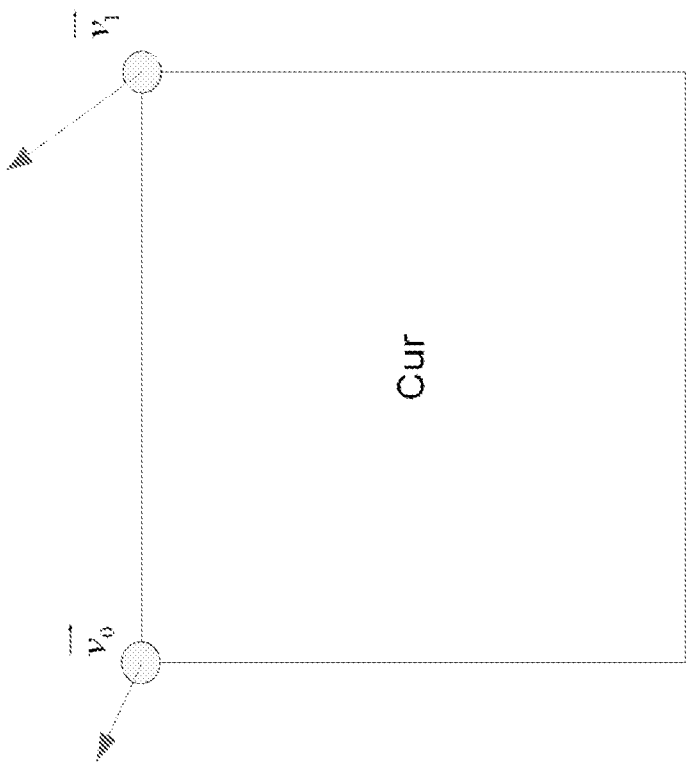


FIG. 20

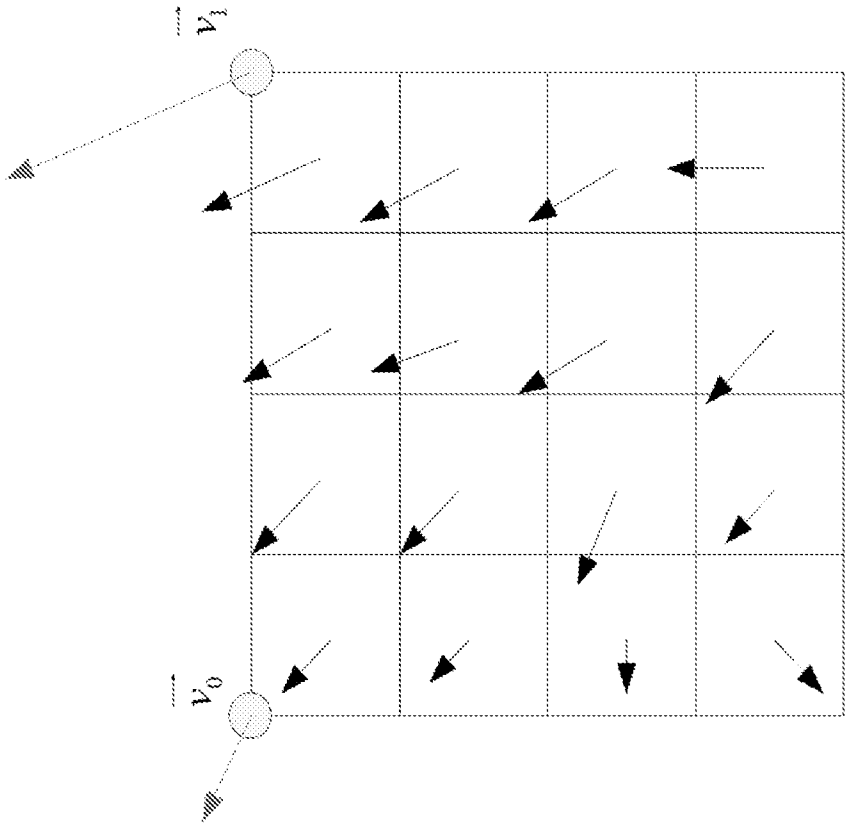


FIG. 21

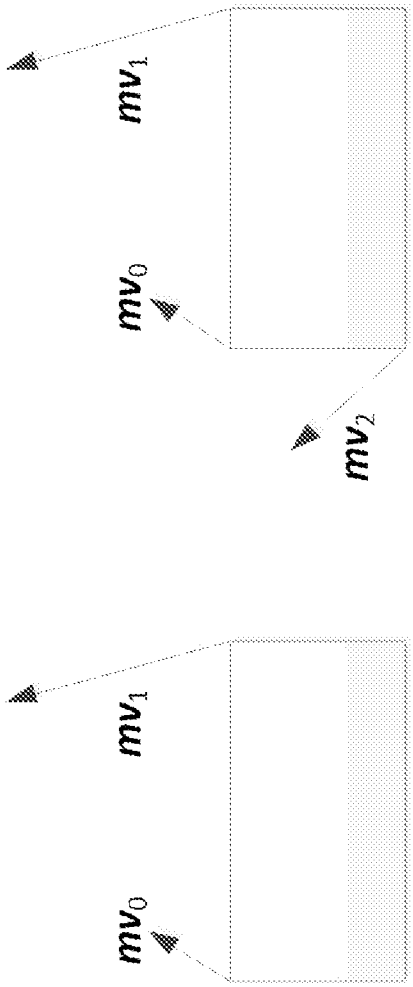


FIG. 22

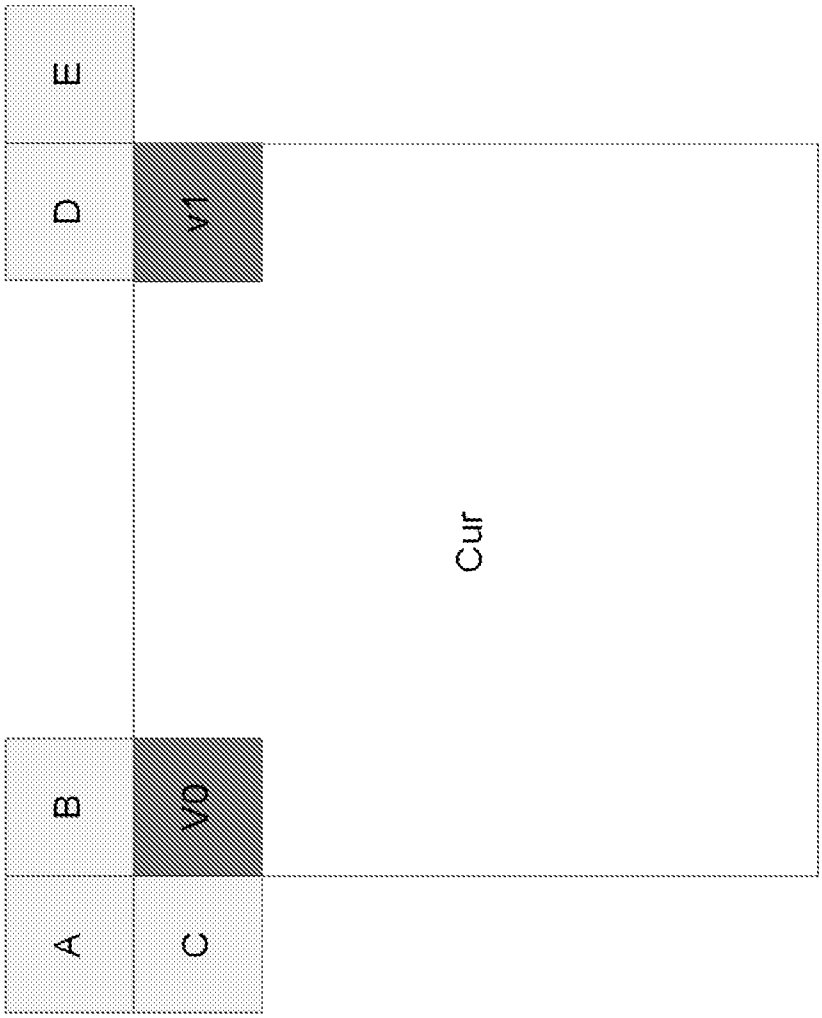


FIG. 23

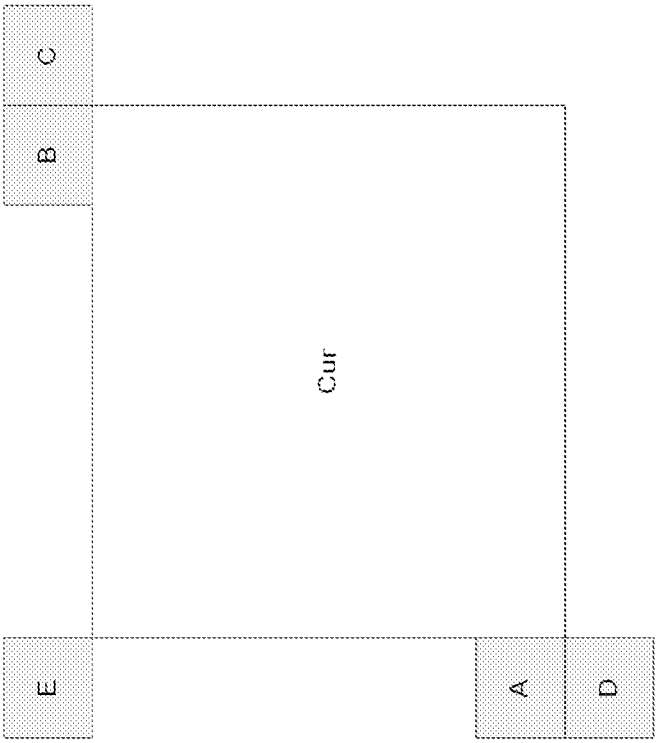


FIG. 24A

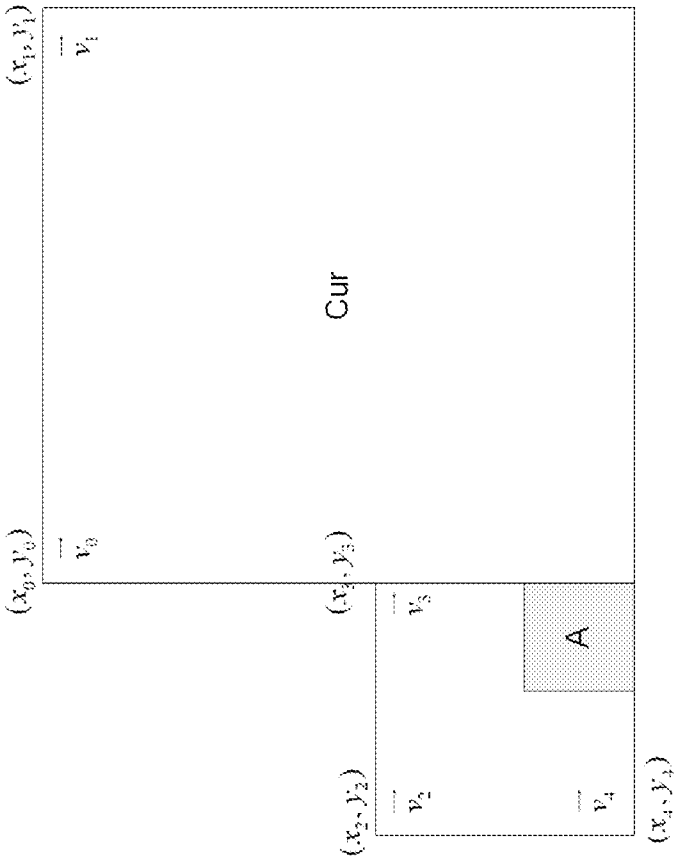


FIG. 24B

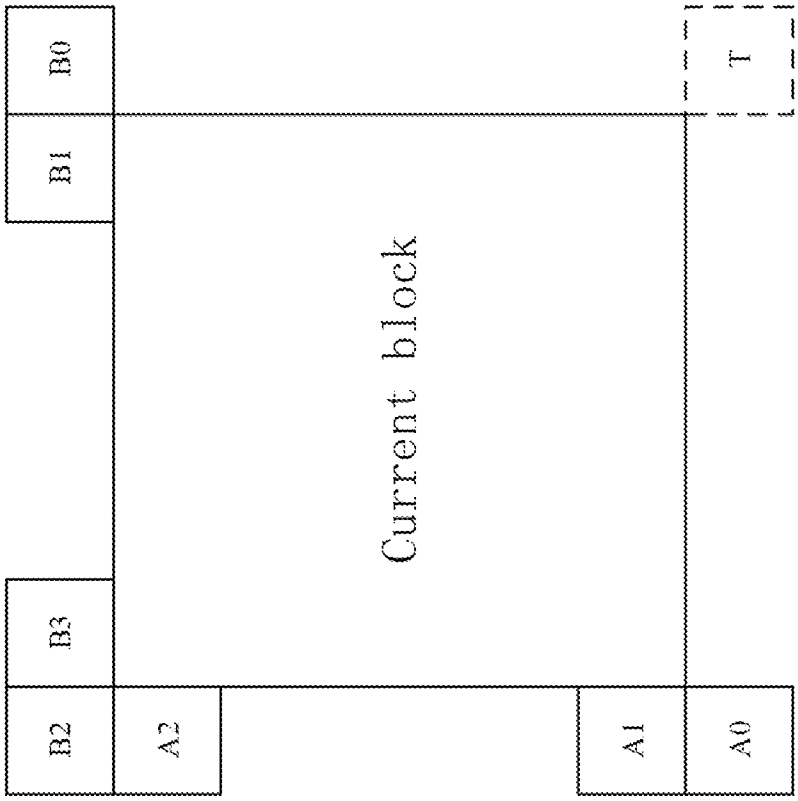


FIG. 25

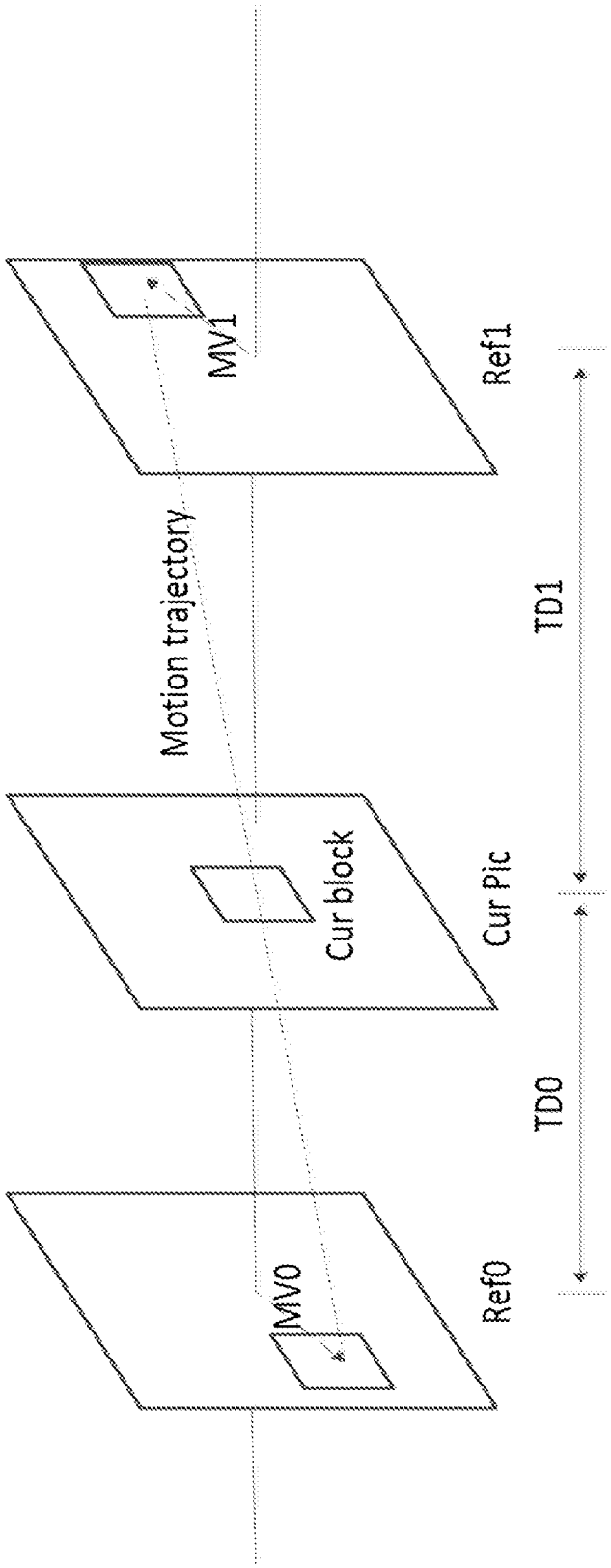

MV1
Ref1
Motion trajectory
TD1
Cur block
Cur Pic
TD0
MV0
Ref0

FIG. 26

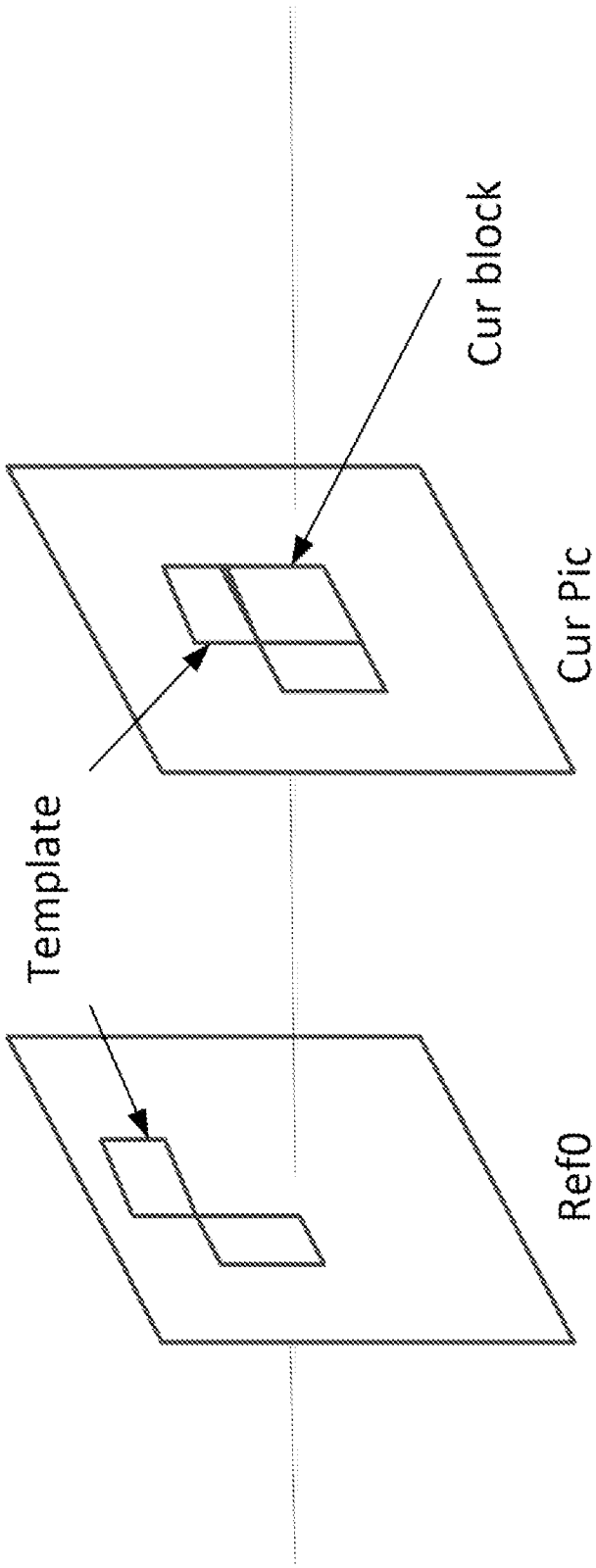


FIG. 27

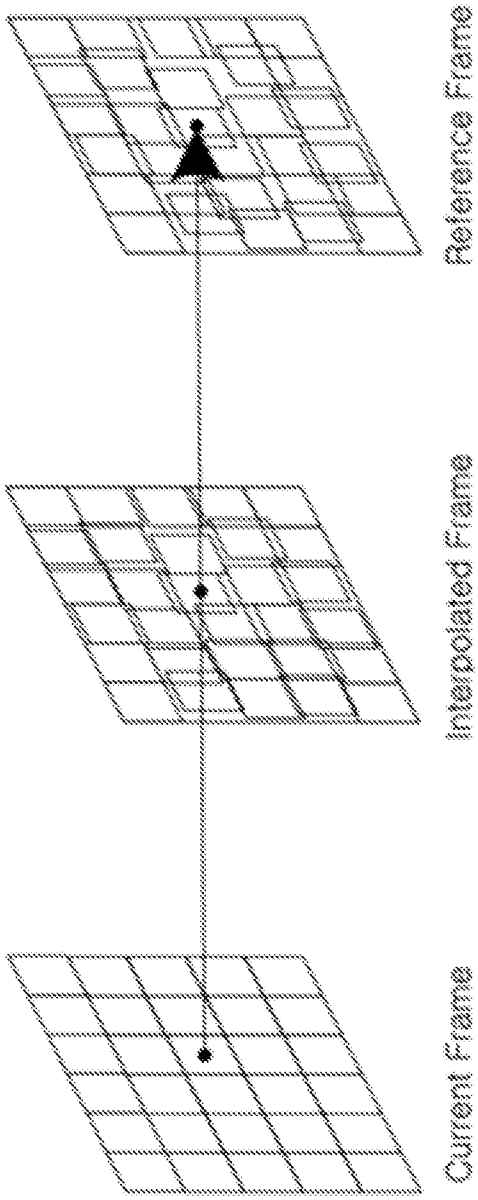


FIG. 28

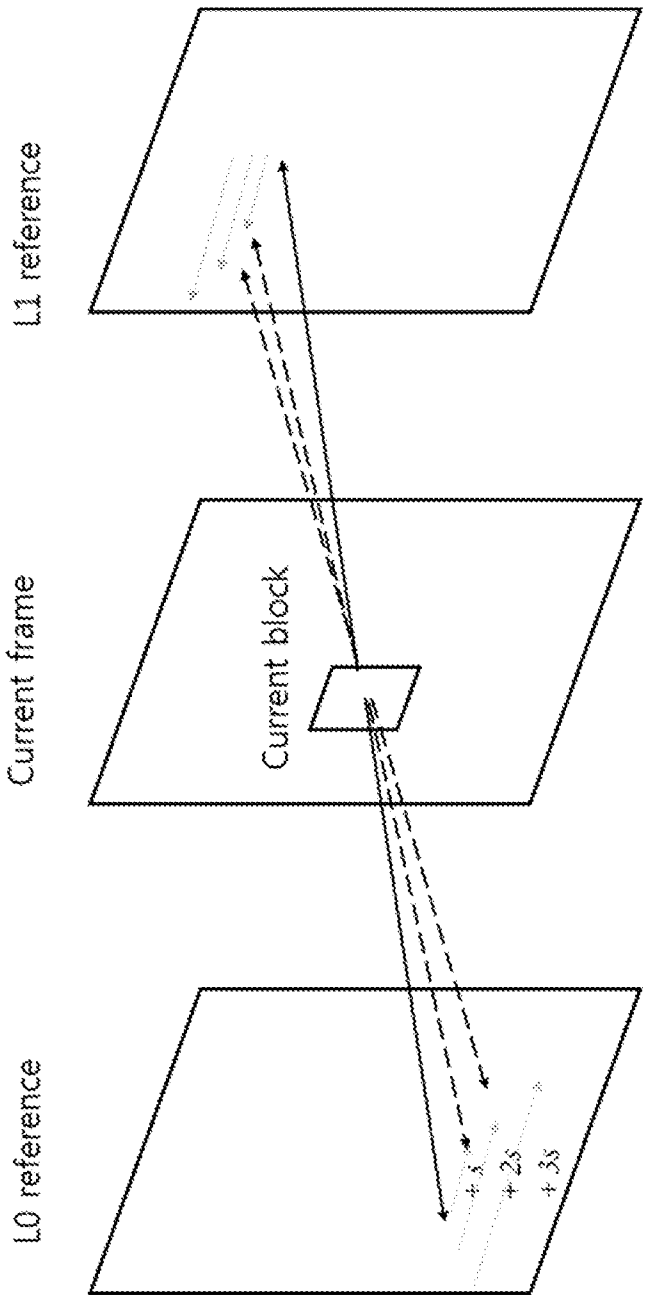

L1 reference
Current frame
Current block
L0 reference
+ 1s
+ 2s
+ 3s

FIG. 29

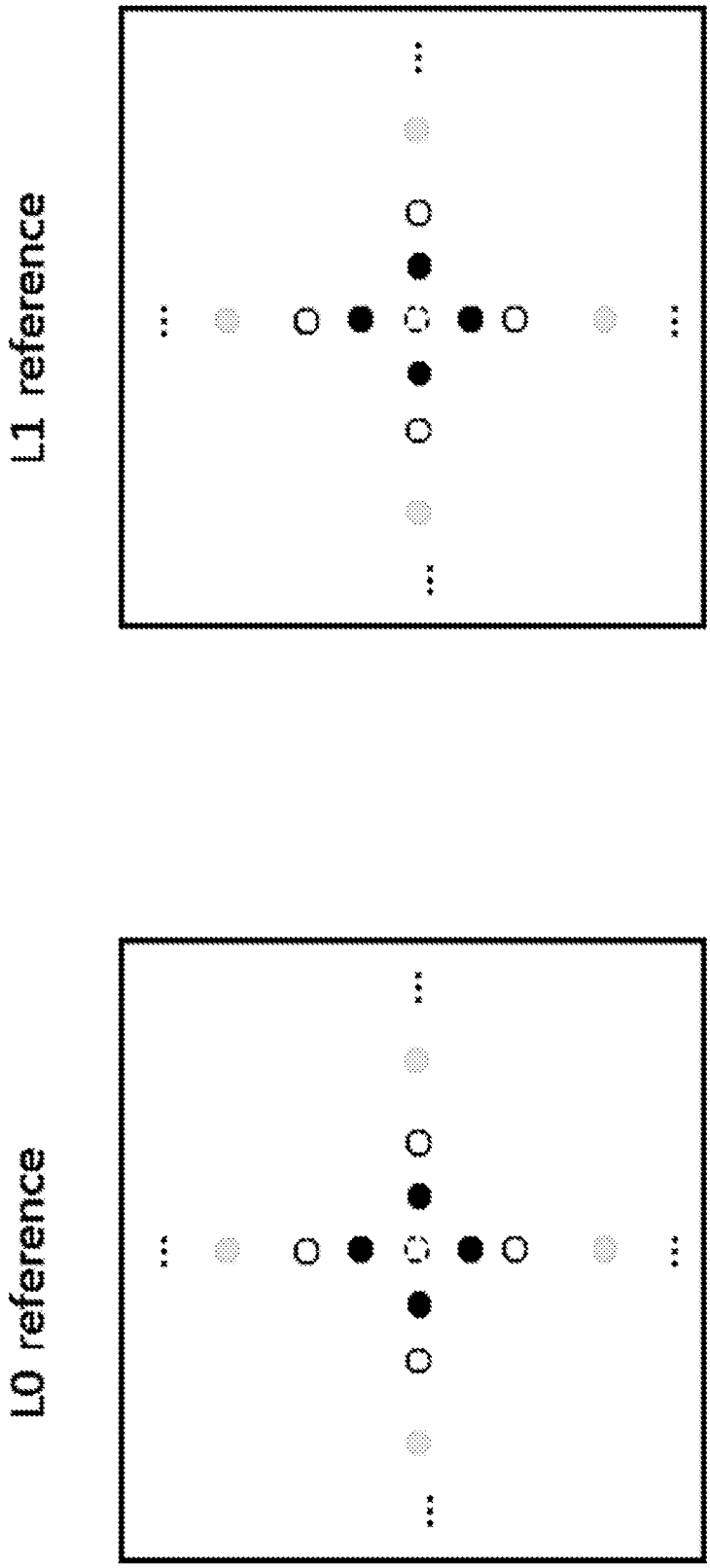


FIG. 30

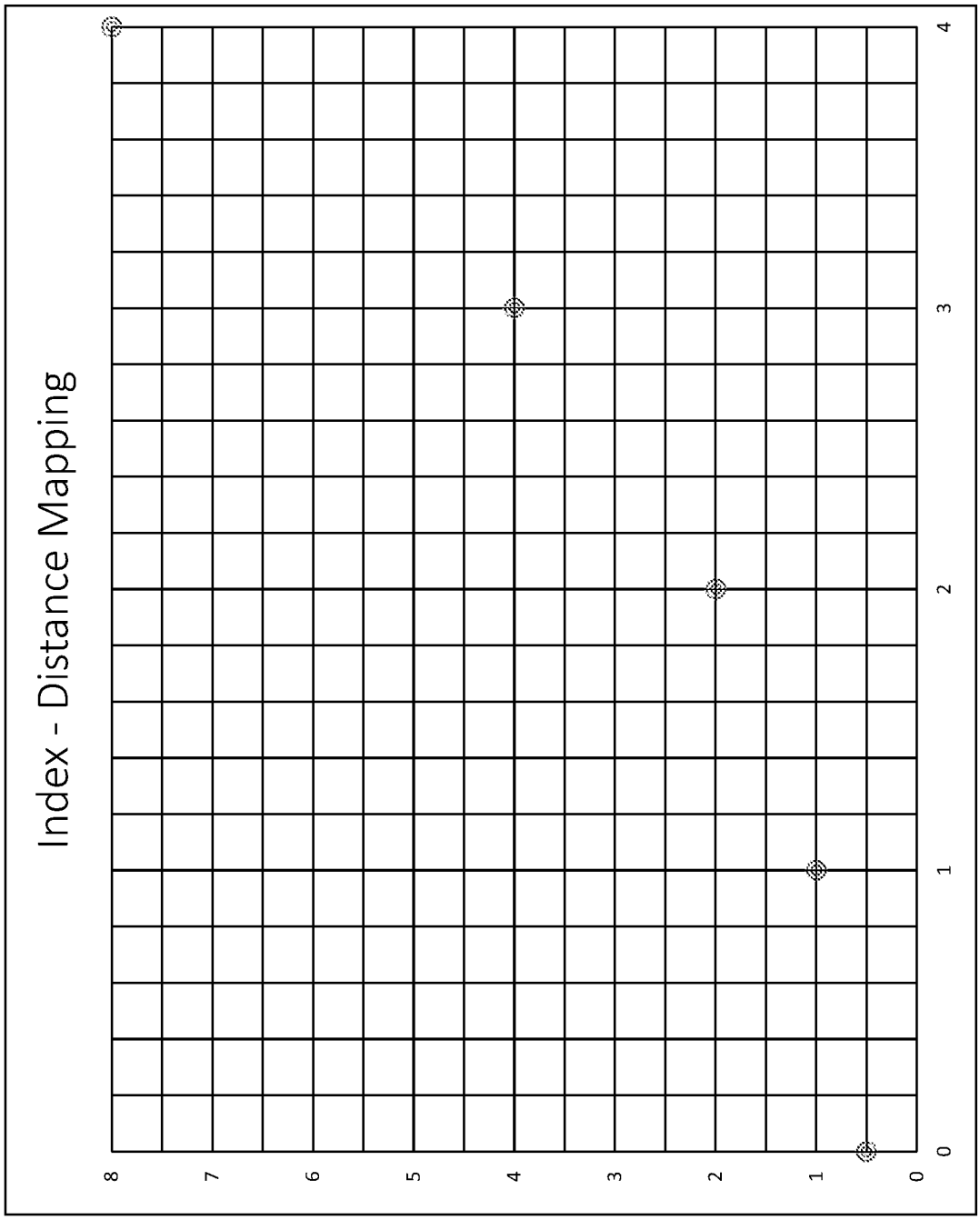


FIG. 31

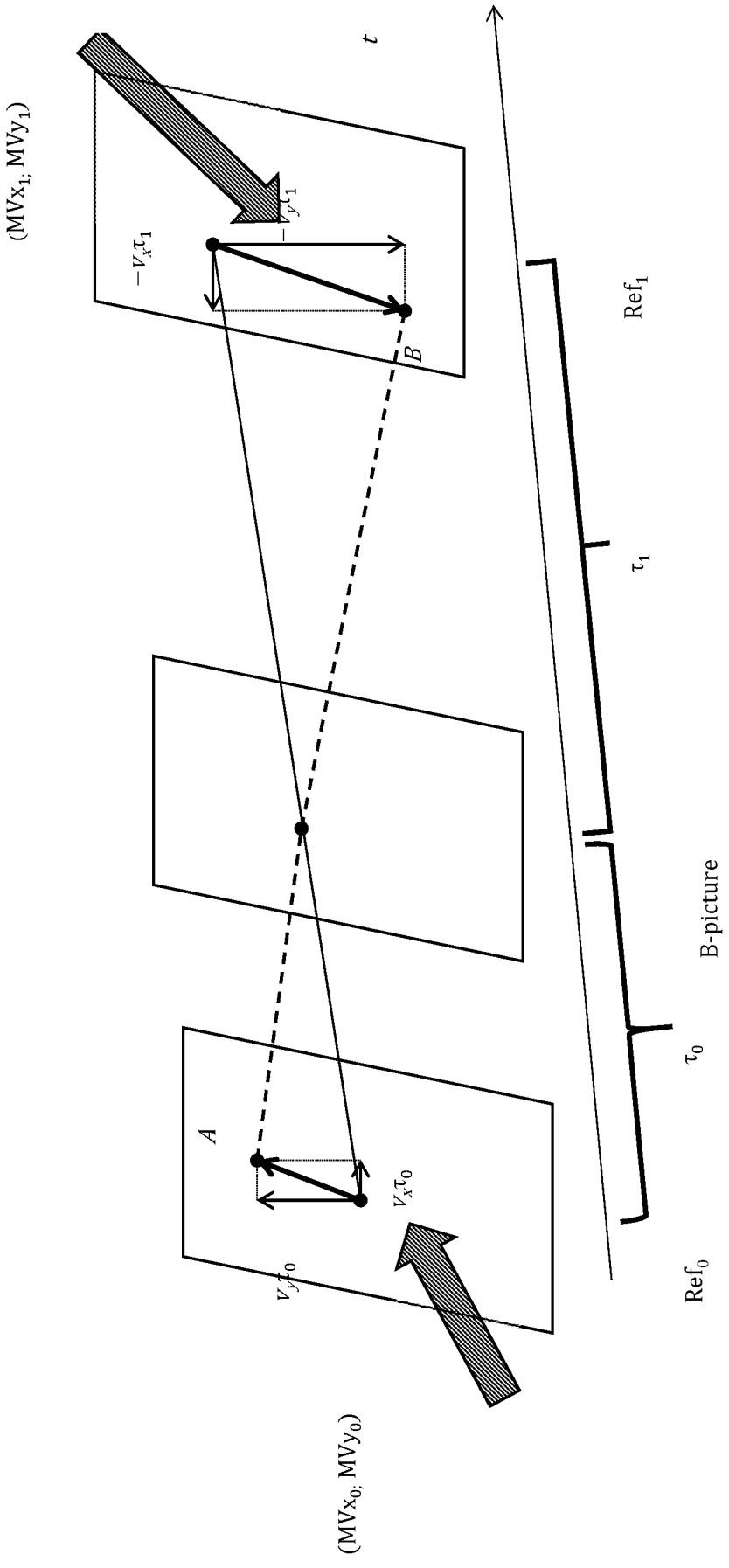

t
(MVx1, MVy1)
−vxτ1
−vyτ1
B
Ref1
τ1
B-picture
τ0
Ref0
A
vxτ0
vyτ0
(MVx0, MVy0)

FIG. 32

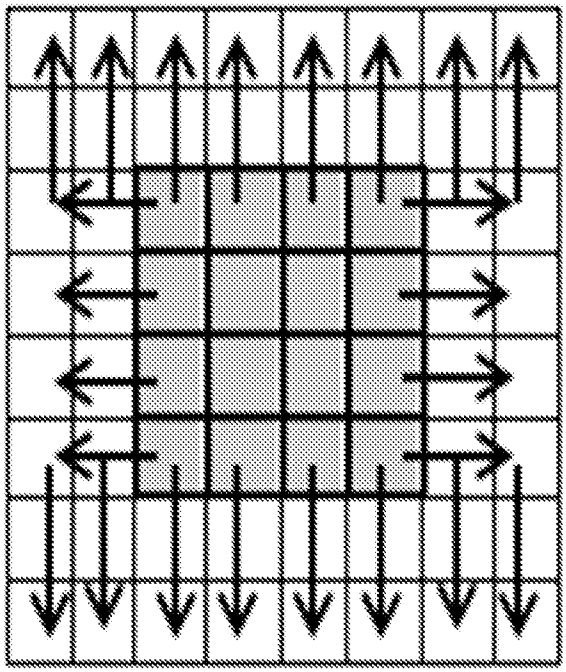


FIG. 33B

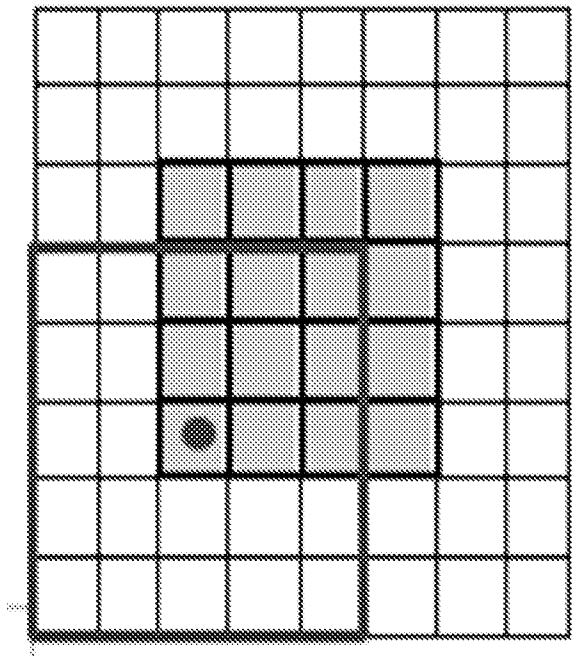


FIG. 33A

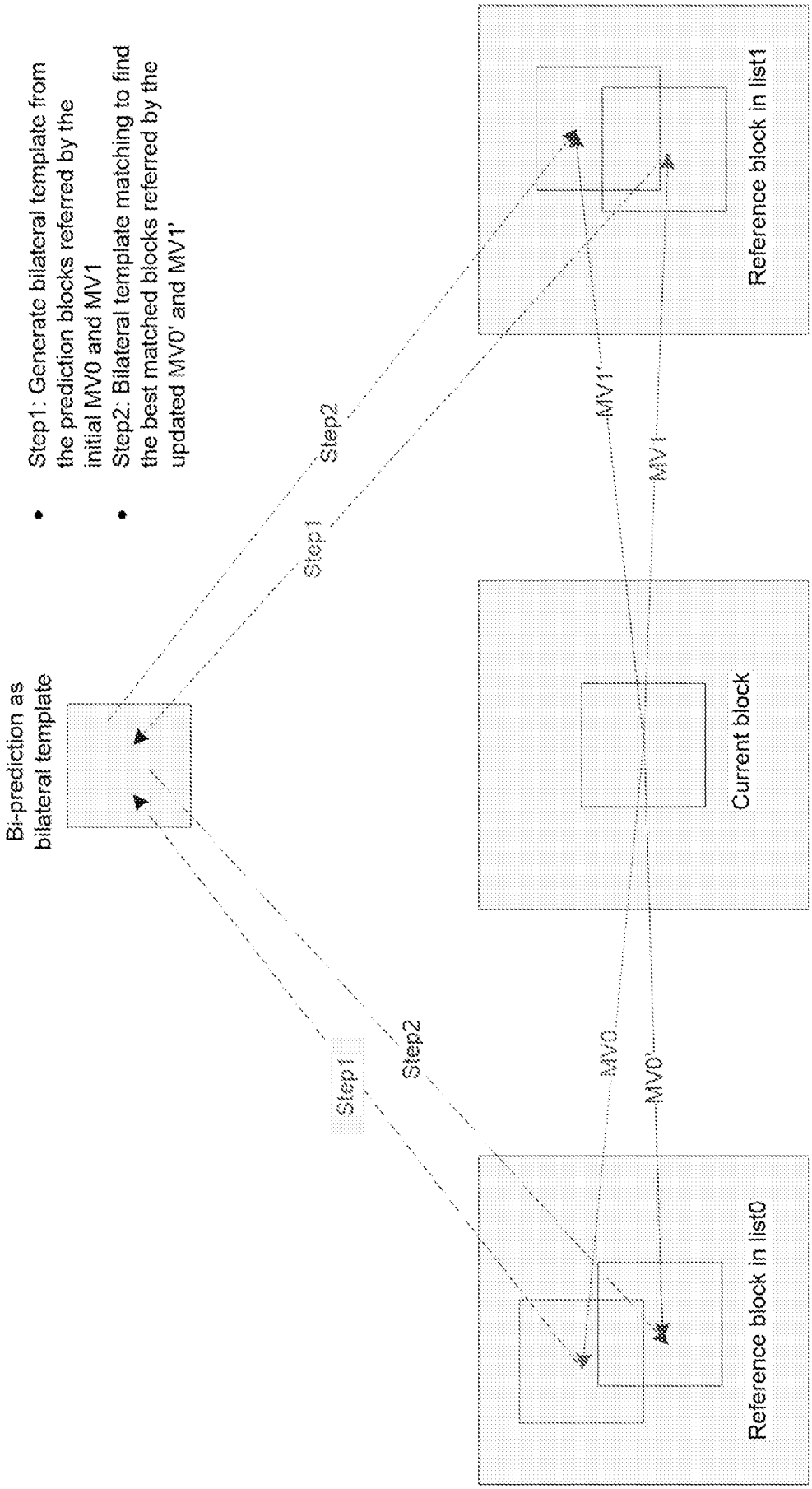

Step1: Generate bilateral template from the prediction blocks referred by the initial MV0 and MV1
Step2: Bilateral template matching to find the best matched blocks referred by the updated MV0' and MV1'
Bi-prediction as bilateral template
Step1
Step2
Step1
Step2
MV0
MV0'
MV1'
MV1
Reference block in list0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

FIG. 34

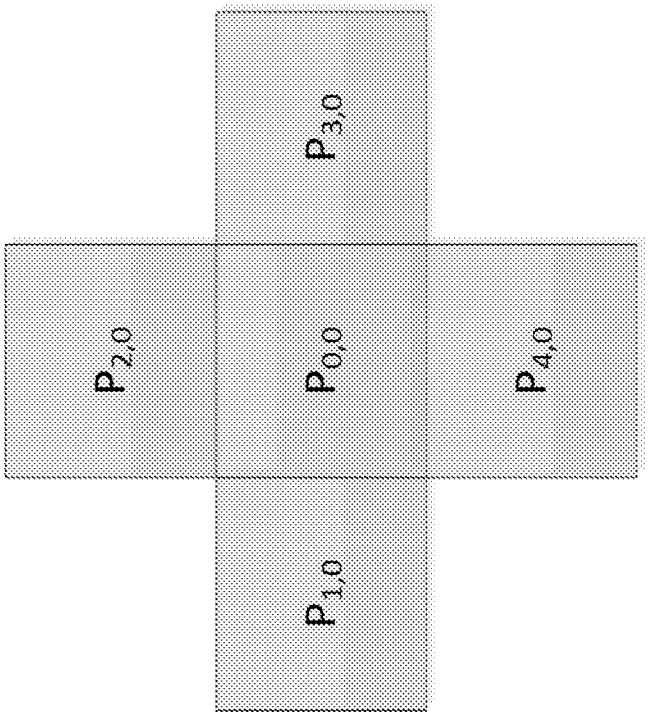


FIG. 35

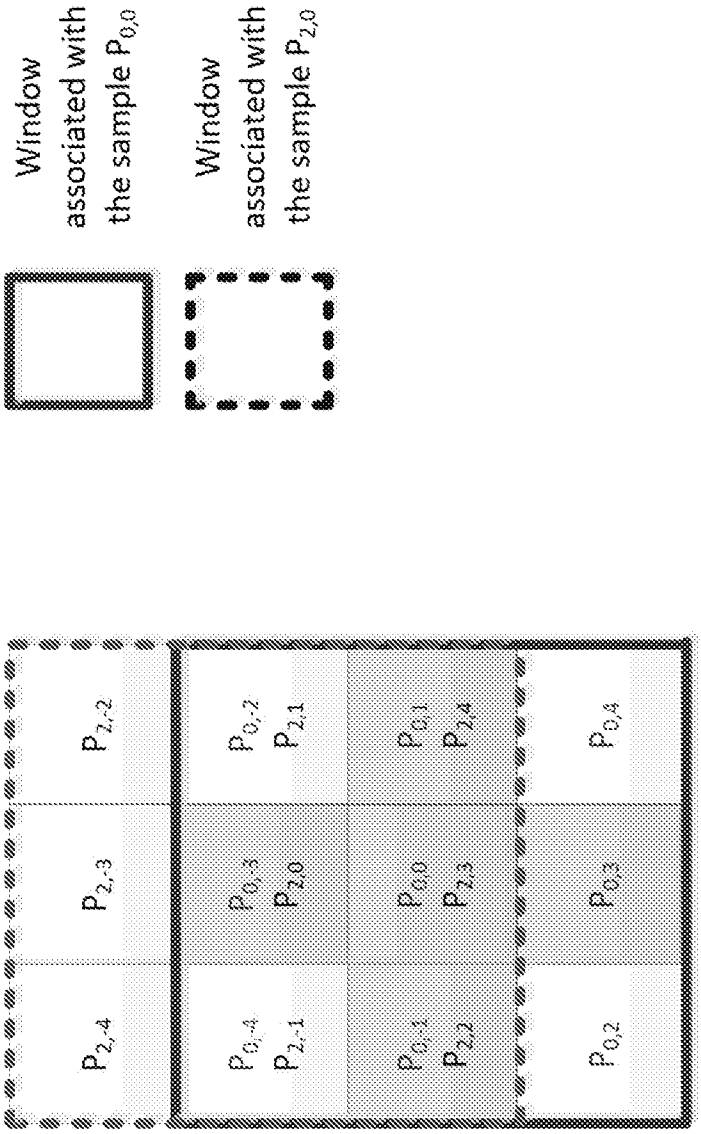

Window associated with the sample $P_{0,0}$
Window associated with the sample $P_{2,0}$
$P_{2,-2}$
$P_{2,-3}$
$P_{2,-4}$
$P_{0,-2}$ $P_{2,1}$
$P_{0,-3}$ $P_{2,0}$
$P_{0,-4}$ $P_{2,-1}$
$P_{0,1}$ $P_{2,4}$
$P_{0,0}$ $P_{2,3}$
$P_{0,-1}$ $P_{2,2}$
$P_{0,4}$
$P_{0,3}$
$P_{0,2}$

FIG. 36

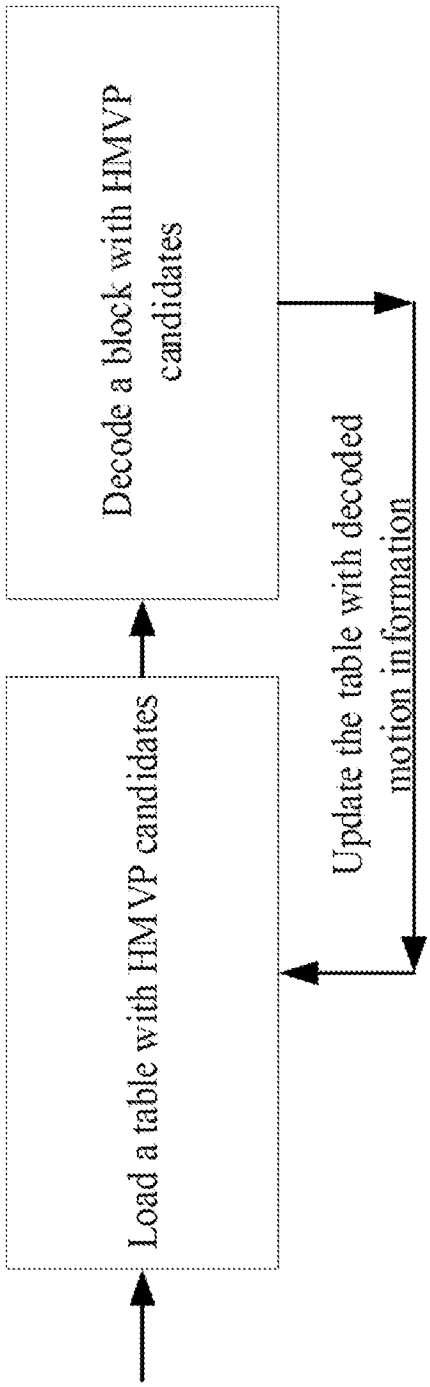

Load a table with HMVP candidates
Decode a block with HMVP candidates
Update the table with decoded motion information

FIG. 37

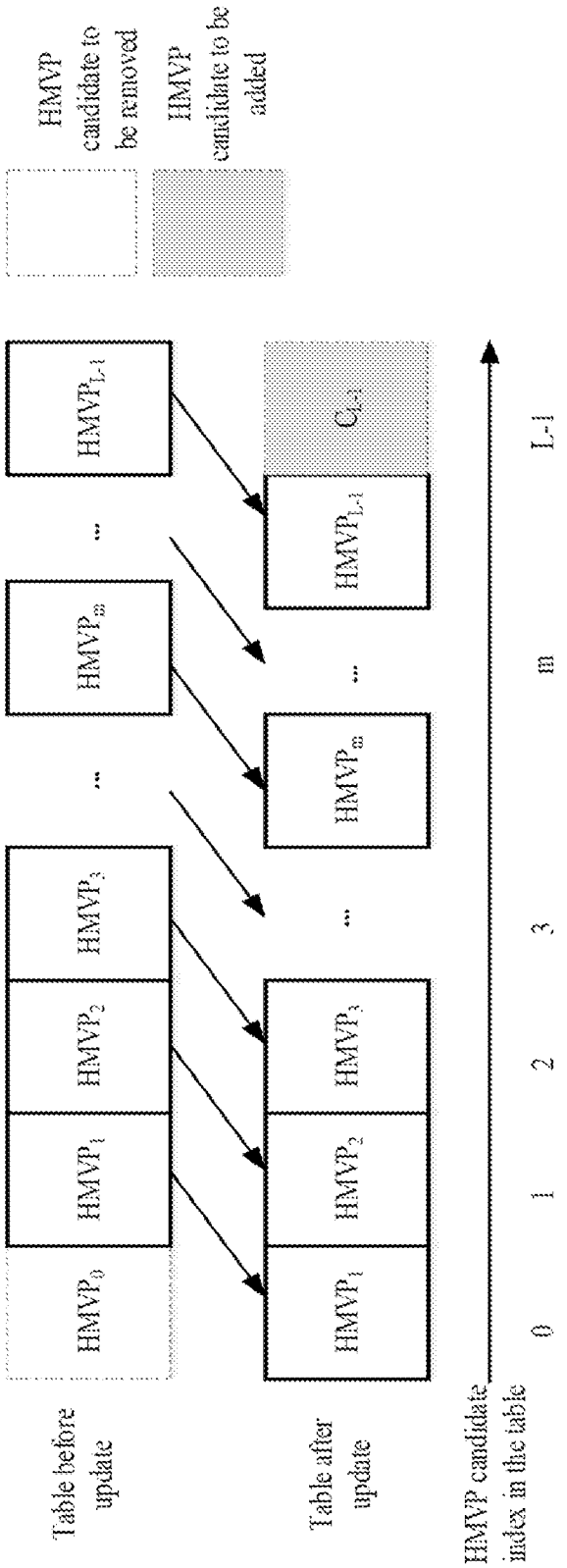

HMVP candidate to be removed
HMVP candidate to be added
HMVP0
HMVP1
HMVP2
HMVP3
...
HMVPm
...
HMVPL-1
Table before update
HMVP1
HMVP2
HMVP3
...
HMVPm
...
HMVPL-1
CL-1
Table after update
0
1
2
3
m
L-1
HMVP candidate index in the table

FIG. 38

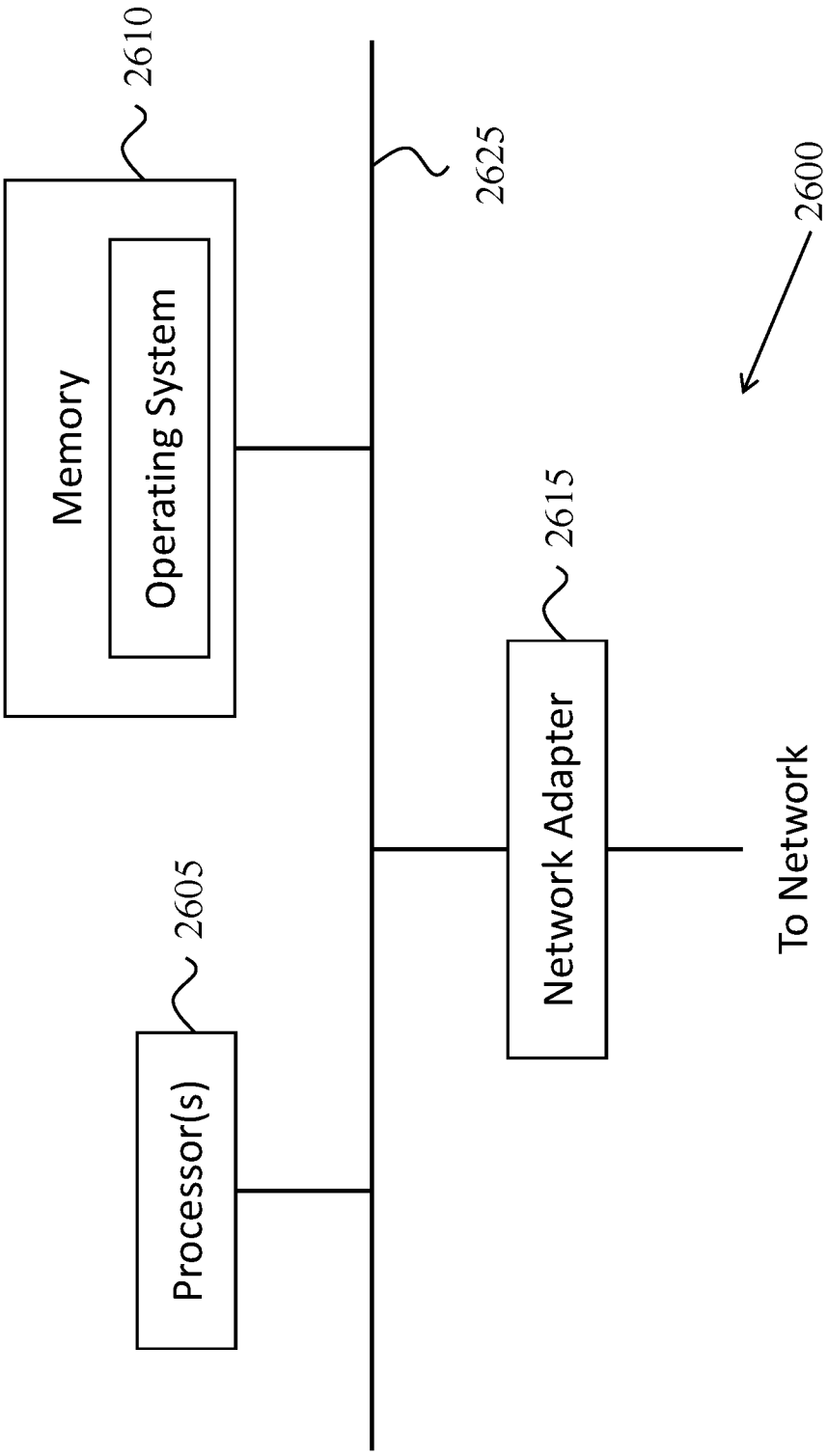

2610
Memory
Operating System
2625
2600
2615
Network Adapter
To Network
2605
Processor(s)

FIG. 39

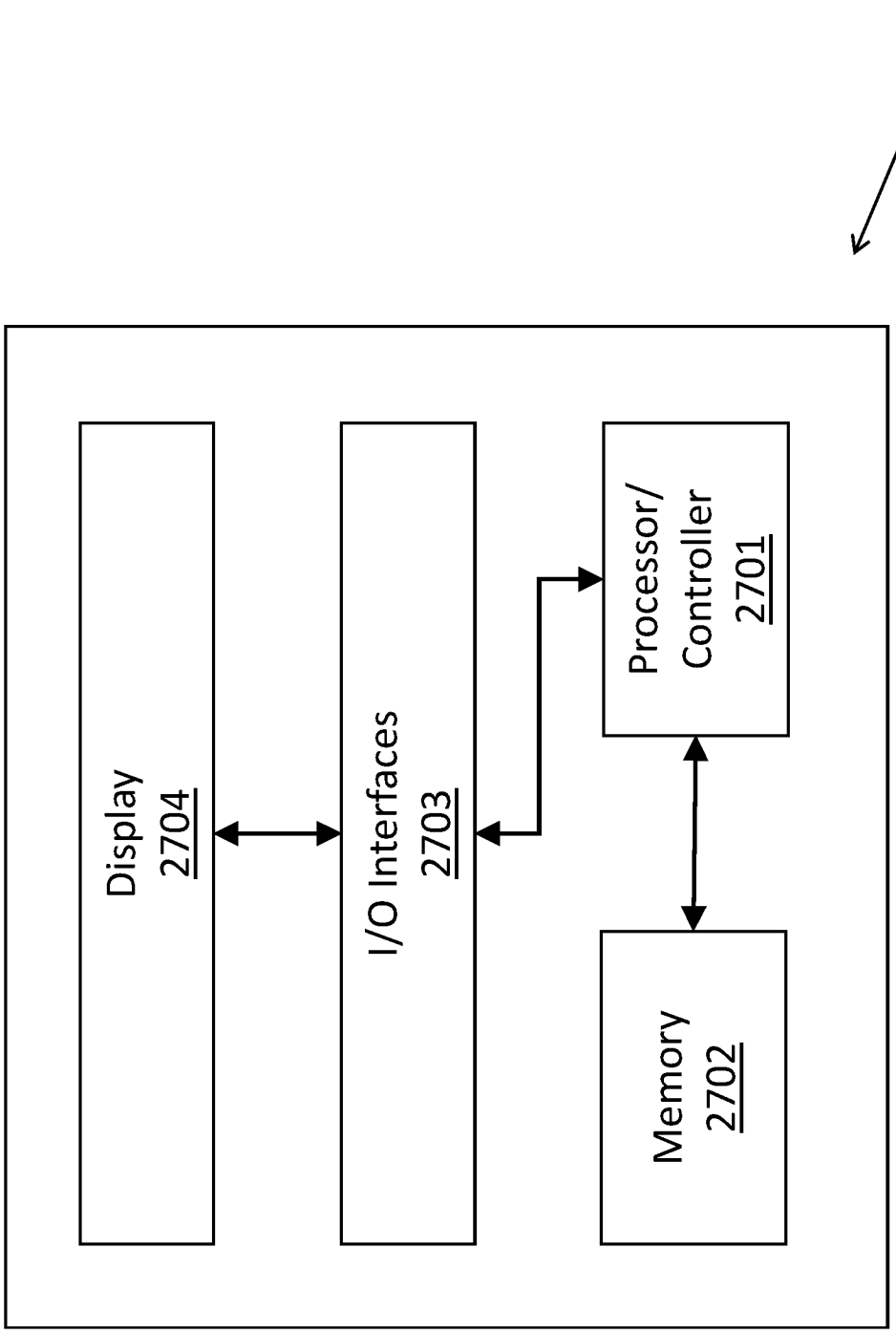

2700
Display
2704
I/O Interfaces
2703
Processor/
Controller
2701
Memory
2702

FIG. 40

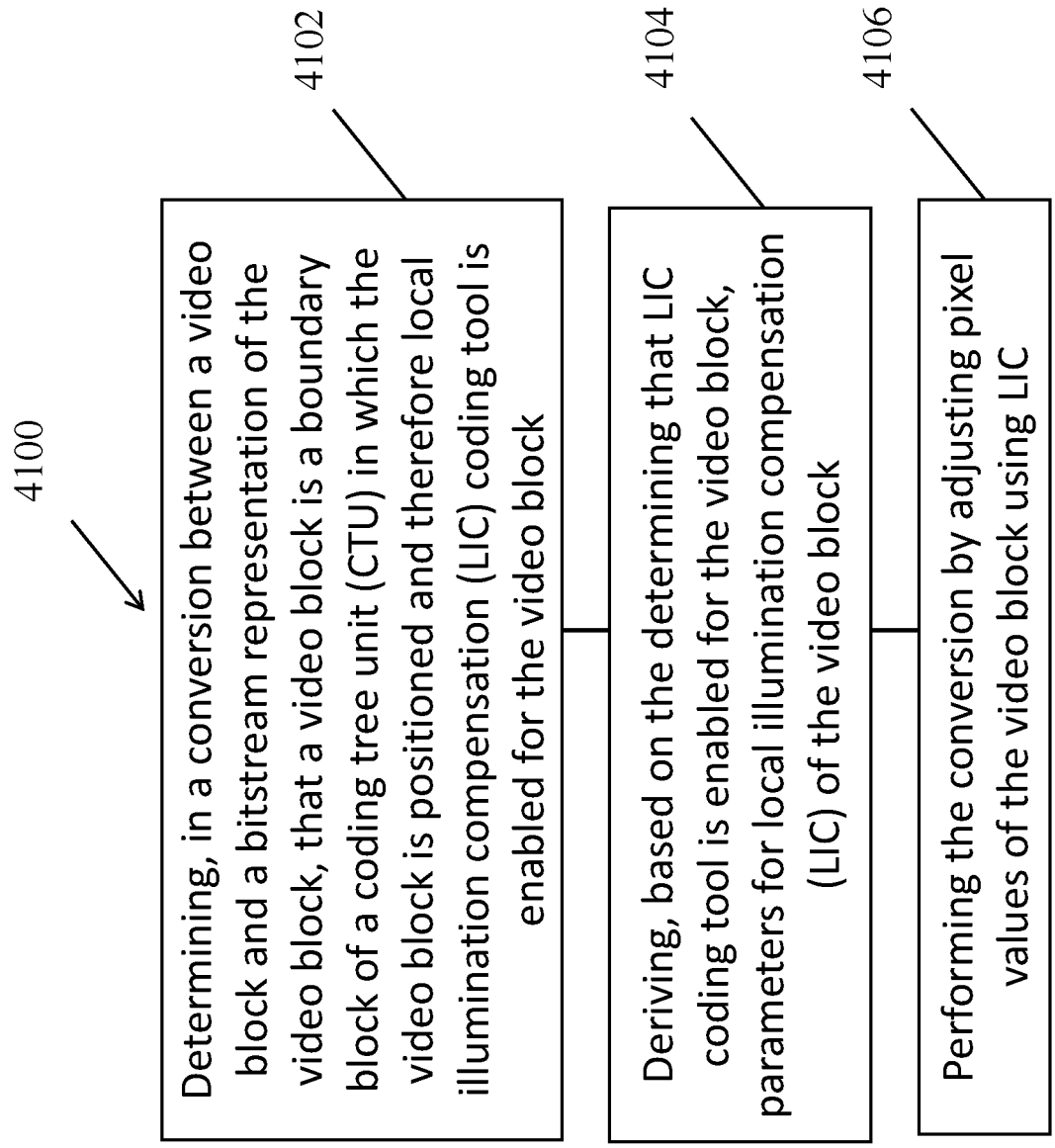

4100
4102
Determining, in a conversion between a video block and a bitstream representation of the video block, that a video block is a boundary block of a coding tree unit (CTU) in which the video block is positioned and therefore local illumination compensation (LIC) coding tool is enabled for the video block
4104
Deriving, based on the determining that LIC coding tool is enabled for the video block, parameters for local illumination compensation (LIC) of the video block
4106
Performing the conversion by adjusting pixel values of the video block using LIC

FIG. 41

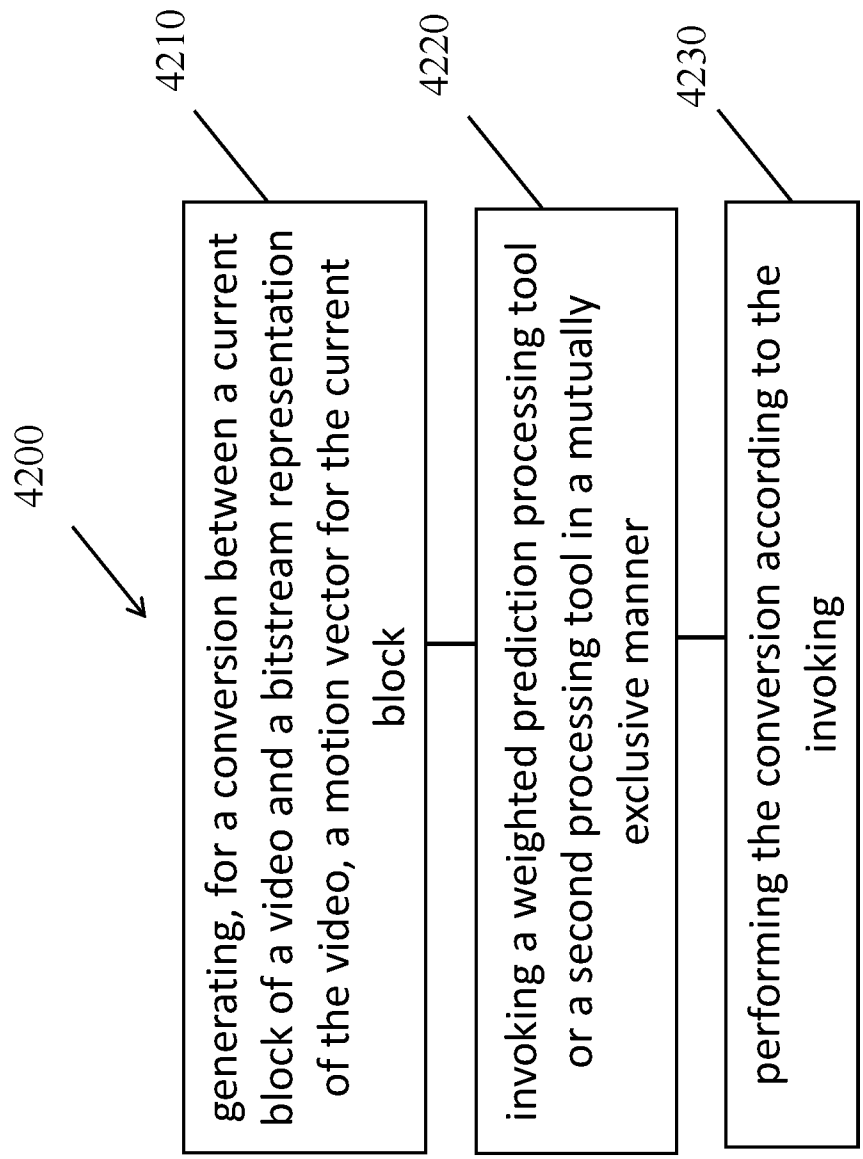

4200
4210
generating, for a conversion between a current block of a video and a bitstream representation of the video, a motion vector for the current block
4220
invoking a weighted prediction processing tool or a second processing tool in a mutually exclusive manner
4230
performing the conversion according to the invoking

FIG. 42

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2020/072241

A. CLASSIFICATION OF SUBJECT MATTER

H04N 19/105(2014.01)i; H04N 19/573(2014.01)i

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

H04N

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

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

VEN;CNABS;CNTXT;CNKI;JVET;JCTVC: video, weighted , prediction, invoke, exclusive, motion 1w vector

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	Yan Ye et al. " 《Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11》 " *AHG13: On bi-prediction with weighted averaging and weighted prediction,* 11 January 2019 (2019-01-11), sections 1 and 2	1-18
A	CN 107113425 A (SAMSUNG ELECTRONICS CO LTD) 29 August 2017 (2017-08-29) the whole document	1-18
A	WO 2019002171 A1 (INTERDIGITAL VC HOLDINGS INC.) 03 January 2019 (2019-01-03) the whole doucment	1-18
A	WO 2019010267 A1 (ARRIS ENTPR LLC) 10 January 2019 (2019-01-10) the whole doucment	1-18
A	US 2018288425 A1 (ARRIS ENTPR LLC) 04 October 2018 (2018-10-04) the whole document	1-18
A	WO 2019001685 A1 (ERICSSON TELEFON AB L M) 03 January 2019 (2019-01-03) the whole document	1-18

☐ Further documents are listed in the continuation of Box C. ☑ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
16 March 2020	**27 March 2020**
Name and mailing address of the ISA/CN **National Intellectual Property Administration, PRC** **6, Xitucheng Rd., Jimen Bridge, Haidian District, Beijing 100088** **China** Facsimile No. **(86-10)62019451**	Authorized officer **LV,Wei** Telephone No. **86-010-62089580**

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2020/072241

Patent document cited in search report			Publication date (day/month/year)	Patent family member(s)			Publication date (day/month/year)
CN	107113425	A	29 August 2017	US	2018288410	A1	04 October 2018
				EP	3217663	A1	13 September 2017
				WO	2016072775	A1	12 May 2016
				KR	20170084055	A	19 July 2017
				EP	3217663	A4	14 February 2018
WO	2019002171	A1	03 January 2019	CN	110832864	A	21 February 2020
WO	2019010267	A1	10 January 2019	CA	3068849	A1	10 January 2019
				US	2019014316	A1	10 January 2019
US	2018288425	A1	04 October 2018	US	10542264	B2	21 January 2020
WO	2019001685	A1	03 January 2019		None		