



- (51) International Patent Classification:
H04N 19/42 (2014.01) *H04N 19/52* (2014.01)
- (21) International Application Number:
PCT/CN2019/124048
- (22) International Filing Date:
09 December 2019 (09.12.2019)
- (25) Filing Language: English
- (26) Publication Language: English
- (30) Priority Data:
PCT/CN2018/119946
08 December 2018 (08.12.2018) CN
PCT/CN2018/121118
14 December 2018 (14.12.2018) CN
PCT/CN2019/075094
14 February 2019 (14.02.2019) CN
PCT/CN2019/075846
22 February 2019 (22.02.2019) CN

- (71) Applicants: **BEIJING BYTEDANCE NETWORK TECHNOLOGY CO., LTD.** [CN/CN]; Room B-0035, 2/F, No. 3 Building, No. 30, Shixing Road, Shijingshan Dis-

trict, Beijing 100041 (CN). **BYTEDANCE INC.** [US/US]; 12655 West Jefferson Boulevard, Sixth Floor, Suite No. 137, Los Angeles, California 90066 (US).

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

(54) Title: SHIFTING ON AFFINE PARAMETERS

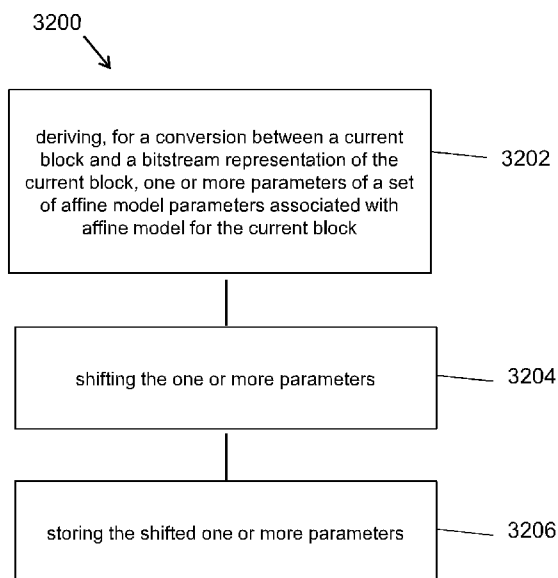


FIG. 32

(57) Abstract: Shifting on affine parameters is described. One example method includes deriving, for a conversion between a current block and a bitstream representation of the current block, one or more parameters of a set of affine model parameters associated with affine model for the current block; shifting the one or more parameters; and storing the shifted one or more parameters.

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, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

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

Declarations under Rule 4.17:

— *of inventorship (Rule 4.17(iv))*

Published:

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

SHIFTING ON AFFINE PARAMETERS

[001] Under the applicable patent law and/or rules pursuant to the Paris Convention, this application is made to timely claim the priority to and benefits of International Patent Application No. PCT/CN2018/119946, filed on December 8, 2018, No. PCT/CN2018/121118, filed on December 14, 2018, PCT/CN2019/075094, filed on February 14, 2019, and No. PCT/CN2019/075846, filed on February 22, 2019. The entire disclosures of International Patent Application No. PCT/CN2018/119946, PCT/CN2018/121118, PCT/CN2019//075094 and No. PCT/CN2019/075846 are incorporated by reference as part of the disclosure of this application.

TECHNICAL FIELD

[002] This patent document relates to image and video coding and decoding.

BACKGROUND

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

SUMMARY

[004] The disclosed techniques may be used by video decoder or encoder embodiments during video decoding or encoding using control point motion vectors and affine coding.

[005] In one example aspect, a method of processing video is disclosed. The method includes associating, with a current video block, a first group of motion vectors (MVs) for determining inherited motion information of other video blocks, a second group of MVs for deriving MVs of sub-blocks of the current video block and a third group of MVs that is included in a bitstream representation of the current video block, and performing a conversion between the current video block and the bitstream representation using the first group of MVs, the second group of MVs or the third group of MVs.

[006] In another example aspect, another method of video processing is disclosed. The method includes performing a conversion between a current block and a bitstream representation of the

current block using affine inherited motion vectors (MVs) for the current block, wherein the affine inherited MVs are derived from (1) MVs stored for an adjacent neighboring basic block, denoted as B_{adj}, or (2) an affine history list.

[007] In yet another example aspect, another method of video processing is disclosed. The method includes performing a conversion between a current block and a bitstream representation of the current block using affine inherited motion vectors (MVs) for the current block, wherein the affine inherited MVs are derived from a first MV stored in a first basic block adjacently neighboring the current block and a second MV stored in a second basic block that is offset from the first building block by an offset.

[008] In yet another example aspect, another method of video processing is disclosed. The method includes associating a first group of control point motion vectors (CPMVs) for determining inherited motion information of blocks coded after a first block, with a second group of CPMVs for deriving MVs of sub-blocks of the first block or a third group of CPMVs that is signaled for the first block, wherein the first group of CPMVs is not identical with the second group of CPMVs or the third group of CPMVs; determining inherited motion information for a second block, which is coded after the first block, based on the first group of CPMVs, and performing a conversation between the second block and a bitstream representation of the second block by using the inherited motion information.

[009] In yet another example aspect, another method of video processing is disclosed. The method includes deriving, for a conversion between a first block of video and a bitstream representation of the first block, affine inherited motion vectors (MVs) for the first block based on stored motion vectors (MVs); and performing the conversion by using the affine inherited MVs.

[0010] In yet another example aspect, another method of video processing is disclosed. The method includes deriving, for a conversion between a current block of video and a bitstream representation of the current block, affine inherited motion vectors (MVs) for the current block based on a first stored motion vector (MV) and a second stored MV different from the first stored MV, wherein the first stored MV is stored in a first basic block neighbouring to the current block, and the second stored MV is stored in a second basic block with an offset to the first basic block; and performing the conversion by using the affine inherited MVs for the current block.

[0011] In yet another example aspect, another method of video processing is disclosed. The method includes deriving, for a conversion between a current block and a bitstream representation of the current block, one or more parameters of a set of affine model parameters associated with affine model for the current block; shifting the one or more parameters; and storing the shifted one or more parameters.

[0012] In yet another example aspect, a video encoder apparatus is disclosed. The video encoder apparatus includes a processor that is configured to implement a method described herein.

[0013] In yet another example aspect, a video decoder apparatus is disclosed. The video decoder apparatus includes a processor that is configured to implement a method described herein.

[0014] In yet another aspect, a computer readable medium having code stored thereupon is disclosed. The code, when executed by a processor, causes the processor to implement a method described in the present document.

[0015] These, and other, aspects are described in the present document.

LISTING OF FIGURES

[0016] FIG. 1 shows an example of derivation process for merge candidate list construction.

[0017] FIG. 2 shows example positions of spatial merge candidates.

[0018] FIG. 3 shows examples of candidate pairs considered for redundancy check of spatial merge candidates.

[0019] FIG. 4A-4B show example positions for the second PU of $N \times 2N$ and $2N \times N$ partitions.

[0020] FIG. 5 is an illustration of motion vector scaling for temporal merge candidate.

[0021] FIG. 6 shows candidate positions for temporal merge candidate, C0 and C1.

[0022] FIG. 7 shows example of combined bi-predictive merge candidate.

[0023] FIG. 8 summarizes derivation process for motion vector prediction candidate.

[0024] FIG. 9 is an example illustration of motion vector scaling for spatial motion vector candidate.

[0025] FIG. 10 shows an example of alternative motion vector predictor (ATMVP) motion prediction for a coding unit CU.

[0026] FIG. 11 shows example of one CU with four sub-blocks (A-D) and its neighbouring blocks (a-d).

[0027] FIG. 12 shows an example flowchart of encoding with different MV precision.

[0028] FIG. 13A-13B show respectively 4 and 6 parameter simplified affine motion models.

[0029] FIG. 14 shows an example of Affine MVF per sub-block.

[0030] FIG. 15A shows an example of a 4-parameter affine model.

[0031] FIG. 15B shows an example of a 6-parameter affine model.

[0032] FIG. 16 shows an example of an MVP for AF_INTER for inherited affine candidates.

[0033] FIG. 17 shows example MVP for AF_INTER for constructed affine candidates.

[0034] FIG. 18A shows an example of candidates for AF_MERGE in a five neighboring block scenario.

[0035] FIG. 18B shows an example flow of a CPMV predictor derivation process.

[0036] FIG. 19 shows example Candidates position for affine merge mode.

[0037] FIG. 20 shows an example of affine inheritance at CTU-row.

[0038] FIG. 21 shows examples of MV stored in adjacent neighbouring basic blocks

[0039] FIG. 22 shows positions in a 4×4 basic block.

[0040] FIG. 23 shows examples of MVs of two adjacent neighbouring blocks.

[0041] FIG. 24 shows an example of MVs used for affine inheritance crossing CTU rows.

[0042] FIG. 25 is a flowchart for an example of a video processing method.

[0043] FIG. 26 is a block diagram of an example of a video processing apparatus.

[0044] FIG. 27 shows an exemplary flowchart to find the first basic block and the second basic block (rectangular block indicates the termination of the whole process).

[0045] FIG. 28 shows another exemplary flowchart to find the first basic block and the second basic block (rectangular block indicates the termination of the whole process).

[0046] FIG. 29 is a flowchart for an example of a video processing method.

[0047] FIG. 30 is a flowchart for an example of a video processing method.

[0048] FIG. 31 is a flowchart for an example of a video processing method.

[0049] FIG. 32 is a flowchart for an example of a video processing method.

DETAILED DESCRIPTION

[0050] The present document provides various techniques that can be used by a decoder of video bitstreams to improve the quality of decompressed or decoded digital video or images. Furthermore, a video encoder may also implement these techniques during the process of encoding in order to reconstruct decoded frames used for further encoding.

[0051] Section headings are used in the present document for ease of understanding and do not limit the embodiments and techniques to the corresponding sections. As such, embodiments from one section can be combined with embodiments from other sections.

1. Summary

[0052] This patent document is related to video coding technologies. Specifically, it is related to motion vector coding 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.

2. Introductory remarks

[0053] 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.

2.1 Inter prediction in HEVC/H.265

[0054] 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.

[0055] When a CU is coded with skip mode, one PU is associated with the CU, and there are no significant residual coefficients, no coded motion vector delta or reference picture index. A

merge mode is specified whereby the motion parameters for the current PU are obtained from neighbouring PUs, including spatial and temporal candidates. The merge mode can be applied to any inter-predicted PU, not only for skip mode. The alternative to merge mode is the explicit transmission of motion parameters, where motion vector (to be more precise, motion vector differences (MVD) 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. 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.

[0056] 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.

[0057] The following text provides the details on the inter prediction modes specified in HEVC. The description will start with the merge mode.

2.1.1 Reference picture list

[0058] In HEVC, the term inter prediction is used to denote prediction derived from data elements (e.g., sample values or motion vectors) of reference pictures other than the current decoded picture. Like in H.264/AVC, a picture can be predicted from multiple reference pictures. The reference pictures that are used for inter prediction are organized in one or more reference picture lists. The reference index identifies which of the reference pictures in the list should be used for creating the prediction signal.

[0059] A single reference picture list, List 0, is used for a P slice and two reference picture lists, List 0 and List 1 are used for B slices. It should be noted reference pictures included in List 0/1 could be from past and future pictures in terms of capturing/display order.

2.1.2 Merge Mode

2.1.2.1 Derivation of candidates for merge mode

[0060] 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

[0061] 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 $2N \times 2N$ prediction unit.

[0062] In the following, the operations associated with the aforementioned steps are detailed.

2.1.2.2 Spatial candidates derivation

[0063] 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. 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 $2N \times 2N$. As an example, FIG. 4A-4B depict the second PU for the case of $N \times 2N$ and $2N \times N$, respectively. When the current PU is partitioned as $N \times 2N$, candidate at position A_1 is not considered for list construction. In 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 $2N \times N$.

[0064] FIG. 2 shows positions of spatial merge candidates.

[0065] FIG. 3 shows candidate pairs considered for redundancy check of spatial merge candidates.

[0066] FIG. 4A-4B show positions for the second PU of $N \times 2N$ and $2N \times N$ partitions.

2.1.2.3 Temporal candidates derivation

[0067] In this step, only one candidate is added to the list. Particularly, in the derivation of this temporal merge candidate, a scaled motion vector is derived based on co-located PU belonging to the picture which has the smallest POC difference with current picture within the given reference picture list. The reference picture list to be used for derivation of the co-located PU is explicitly signalled in the slice header. The scaled motion vector for temporal merge candidate is obtained as illustrated by the dotted line in FIG. 5, which is scaled from the motion vector of the co-located PU using the POC distances, t_b and t_d , where t_b is defined to be the POC difference between the reference picture of the current picture and the current picture and t_d is defined to be the POC difference between the reference picture of the co-located picture and the co-located picture. The reference picture index of temporal merge candidate is set equal to zero. 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.

[0068] FIG. 5 is an example illustration of motion vector scaling for temporal merge candidate.

[0069] 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 coding tree unit (CTU aka. LCU, largest coding unit) row, position C_1 is used. Otherwise, position C_0 is used in the derivation of the temporal merge candidate.

[0070] FIG. 6 shows an example of candidate positions for temporal merge candidate, C_0 and C_1 .

2.1.2.4 Additional candidates insertion

[0071] 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.

[0072] 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. Finally, no redundancy check is performed on these candidates.

2.1.3 AMVP

[0073] AMVP exploits spatio-temporal correlation of motion vector with neighbouring 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 neighbouring PU positions, removing redundant candidates and adding zero vector to make the candidate list to be constant length. Then, the encoder can select the best predictor from the candidate list and transmit the corresponding index indicating the chosen candidate. Similarly with merge index signalling, the index of the best motion vector candidate is encoded using truncated unary. The maximum value to be encoded in this case is 2 (see FIG. 8). In the following sections, details about derivation process of motion vector prediction candidate are provided.

2.1.3.1 Derivation of AMVP candidates

[0074] FIG. 8 summarizes derivation process for motion vector prediction candidate.

[0075] 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. 2.

[0076] For temporal motion vector candidate derivation, one motion vector candidate is selected from two candidates, which are derived based on two different co-located positions. After the first list of spatio-temporal candidates is made, duplicated motion vector candidates in the list are removed. If the number of potential candidates is larger than two, motion vector candidates whose reference picture index within the associated reference picture list is larger than 1 are removed from the list. If the number of spatio-temporal motion vector candidates is smaller than two, additional zero motion vector candidates is added to the list.

2.1.3.2 Spatial motion vector candidates

[0077] 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)

[0078] The no-spatial-scaling cases are checked first followed by the spatial scaling. Spatial scaling is considered when the POC is different between the reference picture of the neighbouring PU and that of the current PU regardless of reference picture list. If all PUs of left candidates are not available or are intra coded, scaling for the above motion vector is allowed to help parallel derivation of left and above MV candidates. Otherwise, spatial scaling is not allowed for the above motion vector.

[0079] FIG. 9 is an illustration of motion vector scaling for spatial motion vector candidate. In a spatial scaling process, the motion vector of the neighbouring PU is scaled in a similar manner as for temporal scaling, as depicted in 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.

2.1.3.3 Temporal motion vector candidates

[0080] 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.

2.2 Sub-CU based motion vector prediction methods in JEM

[0081] In the JEM with QTBT, each CU can have at most one set of motion parameters for each prediction direction. Two sub-CU level motion vector prediction methods are considered in the

encoder by splitting a large CU into sub-CUs and deriving motion information for all the sub-CUs of the large CU. Alternative temporal motion vector prediction (ATMVP) method allows each CU to fetch multiple sets of motion information from multiple blocks smaller than the current CU in the collocated reference picture. In spatial-temporal motion vector prediction (STMVP) method motion vectors of the sub-CUs are derived recursively by using the temporal motion vector predictor and spatial neighbouring motion vector.

[0082] To preserve more accurate motion field for sub-CU motion prediction, the motion compression for the reference frames is currently disabled.

[0083] FIG. 10 shows an example of ATMVP motion prediction for a CU.

2.2.1 Alternative temporal motion vector prediction

[0084] 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. The sub-CUs are square $N \times N$ blocks (N is set to 4 by default).

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.

[0085] In the first step, a reference picture and the corresponding block is determined by the motion information of the spatial neighbouring blocks of the current CU. To avoid the repetitive scanning process of neighbouring blocks, the first merge candidate in the merge candidate list of the current CU is used. The first available motion vector as well as its associated reference index are set to be the **temporal vector** and the index to the motion source picture. This way, in ATMVP, the corresponding block may be more accurately identified, compared with TMVP, wherein the corresponding block (sometimes called collocated block) is always in a bottom-right or center position relative to the current CU.

[0086] 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 $N \times N$ block is identified, it is converted to the motion vectors and reference indices of the current sub-CU, in the same way as TMVP of HEVC, wherein motion scaling and other procedures apply. For example, the decoder checks whether the low-delay condition (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.

2.2.2 Spatial-temporal motion vector prediction (STMVP)

[0087] 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 neighbouring 4×4 blocks in the current frame are labelled as a, b, c, and d.

[0088] The motion derivation for sub-CU A starts by identifying its two spatial neighbours. The first neighbour is the $N \times N$ block above sub-CU A (block c). If this block c is not available or is intra coded the other $N \times N$ 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, starting at block b). The motion information obtained from the neighbouring blocks for each list is scaled to the first reference frame for a given list. Next, temporal motion vector predictor (TMVP) of sub-block A is derived by following the same procedure of TMVP derivation as specified in HEVC. The motion information of the collocated block at location D is fetched and scaled accordingly. Finally, after retrieving and scaling the motion information, all available motion vectors (up to 3) are averaged separately for each reference list. The averaged motion vector is assigned as the motion vector of the current sub-CU.

[0089] FIG. 11 shows an example of one CU with four sub-blocks (A-D) and its neighbouring blocks (a-d).

2.2.3 Sub-CU motion prediction mode signalling

[0090] 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.

[0091] 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.

2.3 Inter prediction methods in VVC

[0092] There are several new coding tools for inter prediction improvement, such as Adaptive motion vector difference resolution (AMVR) for signaling MVD, affine prediction mode, Triangular prediction mode (TPM), ATMVP, Generalized Bi-Prediction (GBI), Bi-directional Optical flow (BIO).

2.3.1 Coding block structure in VVC

[0093] In VVC, a QuadTree/BinaryTree/MultitpleTree (QT/BT/TT) structure is adopted to divide a picture into square or rectangle blocks.

[0094] Besides QT/BT/TT, separate tree (a.k.a. Dual coding tree) is also adopted in VVC for I-frames. With separate tree, the coding block structure are signaled separately for the luma and chroma components.

2.3.2 Adaptive motion vector difference resolution

[0095] 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 VVC, a locally adaptive motion vector resolution (LAMVR) is introduced. In the VVC, MVD can be coded in units of quarter luma samples, integer luma samples or four luma samples (i.e., 1/4-pel, 1-pel, 4-pel). 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.

[0096] 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.

[0097] 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.

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.

[0098] The encoding process is shown in FIG. 12. 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.

[0099] FIG. 12 is a flowchart of encoding with different MV precision.

2.3.3 Affine motion compensation prediction

[00100] 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 the other irregular motions. In VVC, a simplified affine transform motion compensation prediction is applied with 4-parameter affine model and 6-parameter affine model. As shown in FIGS. 13A-13B, the affine motion field of the block is described by two control point motion vectors (CPMVs) for the 4-parameter affine model and 3 CPMVs for the 6-parameter affine model.

[00101] The motion vector field (MVF) of a block is described by the following equations with the 4-parameter affine model (wherein the 4-parameter are defined as the variables a , b , e and f) in equation (1) and 6-parameter affine model (wherein the 4-parameter are defined as the variables a , b , c , d , e and f) in equation (2) respectively:

$$\begin{cases} mv^h(x, y) = ax - by + e = \frac{(mv_1^h - mv_0^h)}{w}x - \frac{(mv_1^v - mv_0^v)}{w}y + mv_0^h \\ mv^v(x, y) = bx + ay + f = \frac{(mv_1^v - mv_0^v)}{w}x + \frac{(mv_1^h - mv_0^h)}{w}y + mv_0^v \end{cases} \quad (1)$$

$$\begin{cases} mv^h(x, y) = ax + cy + e = \frac{(mv_1^h - mv_0^h)}{w}x + \frac{(mv_2^h - mv_0^h)}{h}y + mv_0^h \\ mv^v(x, y) = bx + dy + f = \frac{(mv_1^v - mv_0^v)}{w}x + \frac{(mv_2^v - mv_0^v)}{h}y + mv_0^v \end{cases} \quad (2)$$

[00102] where (mv_0^h, mv_0^v) is motion vector of the top-left corner control point, and (mv_1^h, mv_1^v) is motion vector of the top-right corner control point and (mv_2^h, mv_2^v) is motion vector of the bottom-left corner control point, all of the three motion vectors are called **control point motion vectors (CPMV)**, (x, y) represents the coordinate of a **representative point** relative to the top-left sample within current block and $(mv^h(x, y), mv^v(x, y))$ is the motion vector derived for a sample located at (x, y) . The CP motion vectors may be signaled (like in the affine AMVP mode) or derived on-the-fly (like in the affine merge mode). w and h are the width and height of the current block. In practice, the division is implemented by right-shift with a rounding operation. In VTM, the **representative point** is defined to be the center position of a sub-block, e.g., when the coordinate of the left-top corner of a sub-block relative to the top-left sample within current

block is (xs, ys), the coordinate of the representative point is defined to be (xs+2, ys+2). For each sub-block (i.e., 4x4 in VTM), the representative point is utilized to derive the motion vector for the whole sub-block.

[00103] In order to further simplify the motion compensation prediction, sub-block based affine transform prediction is applied. To derive motion vector of each $M \times N$ (both M and N are set to 4 in current VVC) sub-block, the motion vector of the center sample of each sub-block, as shown in FIG. 14, is calculated according to Equation (1) and (2), and rounded to 1/16 fraction accuracy. Then the motion compensation interpolation filters for 1/16-pel are applied to generate the prediction of each sub-block with derived motion vector. The interpolation filters for 1/16-pel are introduced by the affine mode.

[00104] FIG. 14 shows example of Affine MVF per sub-block.

[00105] After MCP, the high accuracy motion vector of each sub-block is rounded and saved as the same accuracy as the normal motion vector.

2.3.3.1 Signaling of affine prediction

[00106] Similar to the translational motion model, there are also two modes for signaling the side information due affine prediction. They are AFFINE_INTER and AFFINE_MERGE modes.

2.3.3.2 AF_INTER mode

[00107] 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.

[00108] In this mode, for each reference picture list (List 0 or List 1), an affine AMVP candidate list is constructed with three types of affine motion predictors in the following order, wherein each candidate includes the estimated CPMVs of the current block. The differences of the best CPMVs found at the encoder side (such as mv_0 mv_1 mv_2 in FIG. 17) and the estimated CPMVs are signalled. In addition, the index of affine AMVP candidate from which the estimated CPMVs are derived is further signalled.

1) Inherited affine motion predictors

[00109] The checking order is similar to that of spatial MVPs in HEVC AMVP list construction. First, a left inherited affine motion predictor is derived from the first block in {A1, A0} that is affine coded and has the same reference picture as in current block. Second, an above inherited

affine motion predictor is derived from the first block in $\{B1, B0, B2\}$ that is affine coded and has the same reference picture as in current block. The five blocks A1, A0, B1, B0, B2 are depicted in FIG. 16.

[00110] Once a neighboring block is found to be coded with affine mode, the CPMVs of the coding unit covering the neighboring block are used to derive predictors of CPMVs of current block. For example, if A1 is coded with non-affine mode and A0 is coded with 4-parameter affine mode, the left inherited affine MV predictor will be derived from A0. In this case, the CPMVs of a CU covering A0, as denoted by MV_0^N for the top-left CPMV and MV_1^N for the top-right CPMV in FIG. 18B are utilized to derive the estimated CPMVs of current block, denoted by MV_0^C , MV_1^C , MV_2^C for the top-left (with coordinate (x0, y0)), top-right (with coordinate (x1, y1)) and bottom-right positions (with coordinate (x2, y2)) of current block.

2) Constructed affine motion predictors

[00111] A constructed affine motion predictor consists of control-point motion vectors (CPMV) that are derived from neighboring inter coded blocks, as shown in FIG. 17, that have the same reference picture. If the current affine motion model is 4-parameter affine, the number of CPMVs is 2, otherwise if the current affine motion model is 6-parameter affine, the number of CPMVs is 3. The top-left CPMV $\overline{mv}_0$ is derived by the MV at the first block in the group $\{A, B, C\}$ that is inter coded and has the same reference picture as in current block. The top-right CPMV $\overline{mv}_1$ is derived by the MV at the first block in the group $\{D, E\}$ that is inter coded and has the same reference picture as in current block. The bottom-left CPMV $\overline{mv}_2$ is derived by the MV at the first block in the group $\{F, G\}$ that is inter coded and has the same reference picture as in current block.

- If the current affine motion model is 4-parameter affine, then a constructed affine motion predictor is inserted into the candidate list only if both $\overline{mv}_0$ and $\overline{mv}_1$ are founded, that is, $\overline{mv}_0$ and $\overline{mv}_1$ are used as the estimated CPMVs for top-left (with coordinate (x0, y0)), top-right (with coordinate (x1, y1)) positions of current block.
- If the current affine motion model is 6-parameter affine, then a constructed affine motion predictor is inserted into the candidate list only if $\overline{mv}_0$, $\overline{mv}_1$ and $\overline{mv}_2$ are all founded, that is, $\overline{mv}_0$, $\overline{mv}_1$ and $\overline{mv}_2$ are used as the estimated CPMVs for top-left (with coordinate

(x0, y0)), top-right (with coordinate (x1, y1)) and bottom-right (with coordinate (x2, y2)) positions of current block.

[00112] No pruning process is applied when inserting a constructed affine motion predictor into the candidate list.

3) Normal AMVP motion predictors

[00113] The following applies until the number of affine motion predictors reaches the maximum.

- 1) Derive an affine motion predictor by setting all CPMVs equal to $\overline{mv}_2$ if available.
- 2) Derive an affine motion predictor by setting all CPMVs equal to $\overline{mv}_1$ if available.
- 3) Derive an affine motion predictor by setting all CPMVs equal to $\overline{mv}_0$ if available.
- 4) Derive an affine motion predictor by setting all CPMVs equal to HEVC TMVP if available.
- 5) Derive an affine motion predictor by setting all CPMVs to zero MV.

[00114] Note that $\overline{mv}_i$ is already derived in constructed affine motion predictor.

[00115] FIG. 15A shows an example of a 4-parameter affine model. FIG. 15B shows an example of a 6-parameter affine model.

[00116] FIG. 16 shows an example of an MVP for AF_INTER for inherited affine candidates.

[00117] FIG. 17 shows an example of an MVP for AF_INTER for constructed affine candidates.

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 FIGS. 15A-15B. It is proposed to derive the MV as follows, i.e., mvd_1 and mvd_2 are predicted from mvd_0 .

$$\begin{aligned} mv_0 &= \overline{mv}_0 + mvd_0 \\ mv_1 &= \overline{mv}_1 + mvd_1 + mvd_0 \\ mv_2 &= \overline{mv}_2 + mvd_2 + mvd_0 \end{aligned}$$

[00118] Wherein $\overline{mv}_i$, mvd_i and mv_i 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. 15B. 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,

$\text{newMV} = \text{mvA} + \text{mvB}$ and the two components of newMV is set to $(x_A + x_B)$ and $(y_A + y_B)$, respectively.

2.3.3.3 AF_MERGE mode

[00119] 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. 18A (denoted by A, B, C, D, E in order). For example, if the neighbour left bottom block is coded in affine mode as denoted by A0 in FIG. 18B, the Control Point (CP) motion vectors mv_0^N , mv_1^N and mv_2^N of the top left corner, above right corner and left bottom corner of the neighbouring CU/PU which contains the block A are fetched. And the motion vector mv_0^C , mv_1^C and mv_2^C (which is only used for the 6-parameter affine model) of the top left corner/top right/bottom left on the current CU/PU is calculated based on mv_0^N , mv_1^N and mv_2^N . It should be noted that sub-block (e.g. 4×4 block in VTM) located at the top-left corner stores mv0, the sub-block located at the top-right corner may store mv1 if the current block is affine coded. If the current block is coded with the 6-parameter affine model, the sub-block located at the bottom-left corner stores mv2; otherwise (with the 4-parameter affine model), LB stores mv2'. Other sub-blocks stores the MVs used for MC.

[00120] After the CPMV of the current CU mv_0^C , mv_1^C and mv_2^C are derived, according to the simplified affine motion model Equation (1) and (2), 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.

[00121] FIG. 18A shows example candidates for AF_MERGE in a 5 neighboring blocks case.

[00122] FIG. 18B shows an example of CPMV predictor derivation process.

[00123] It is proposed that an affine merge candidate list is constructed with following steps:

1) Insert inherited affine candidates

[00124] Inherited affine candidate means that the candidate is derived from the affine motion model of its valid neighbor affine coded block. The maximum two inherited affine candidates are derived from affine motion model of the neighboring blocks and inserted into the candidate list. For the left predictor, the scan order is $\{A0, A1\}$; for the above predictor, the scan order is $\{B0, B1, B2\}$.

2) Insert constructed affine candidates

[00125] If the number of candidates in affine merge candidate list is less than MaxNumAffineCand (e.g., 5), 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.

[00126] A) The motion information for the control points is derived firstly from the specified spatial neighbors and temporal neighbor shown in FIG. 19. CP_k (k=1, 2, 3, 4) represents the k-th control point. A₀, A₁, A₂, B₀, B₁, B₂ and B₃ are spatial positions for predicting CP_k (k=1, 2, 3); T is temporal position for predicting CP₄.

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.

[00127] FIG. 19 shows examples of candidate positions for affine merge mode.

[00128] The motion information of each control point is obtained according to the following priority order:

- 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.
- For CP₂, the checking priority is B₁->B₀.
- For CP₃, the checking priority is A₁->A₀.
- For CP₄, T is used.

[00129] B) Secondly, the combinations of controls points are used to construct an affine merge candidate.

- I. 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.
- II. 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 two combinations

({CP1, CP2}, {CP1, CP3}). The two combinations will be converted to a 4-parameter motion model represented by top-left and top-right control points.

III. 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}

- i. For each combination, the reference indices of list X for each CP are checked, if they are all the same, then this combination has valid CPMVs for list X. If the combination does not have valid CPMVs for both list 0 and list 1, then this combination is marked as invalid. Otherwise, it is valid, and the CPMVs are put into the sub-block merge list.

3) Padding with zero motion vectors

[00130] 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.

[00131] More specifically, for the sub-block merge candidate list, a 4-parameter merge candidate with MVs set to (0, 0) and prediction direction set to uni-prediction from list 0 (for P slice) and bi-prediction (for B slice).

2.3.3.4 Storage required by affine model inheritance

[00132] Memory required inside a CTU

[00133] To conduct affine model inheritance as shown in FIG. 18B, additional information is stored in each 8×8 block inside a CTU.

1) Three CPMVs for the two reference lists, requiring $2 \text{ (2 reference lists)} \times 2 \text{ (x and y components)} \times 16 \text{ (16 bits for one component)} \times 3 \text{ (3 CPMVs)} = 192 \text{ bits (or 24 bytes in a software design)}$.

2) The coordinate of the top-left corner of the CU, requiring $2 \times 13 = 26 \text{ bits (or 4 bytes in a software design)}$.

3) The width and height of the CU, requiring $2 \times 7 = 14 \text{ bits (or 2 bytes in a software design)}$.

[00134] So totally 232 bits (or 30 bytes) side information is required to be stored for each 8×8 block. We should notice that in HEVC, the total amount of motion information required to be stored in each 8×8 block is $2 \text{ (2 reference lists)} \times 2 \text{ (x and y components)} \times 16 \text{ (16 bits for one component)} \times 4 \text{ (4 MVs for 4 4×4blocks)} = 256 \text{ bits (or 32 bytes in a software design)}$.

[00135] Therefore, the memory inside a CTU required by motion information is increased by $232/256 = 90\%$ or $30/32 = 94\%$ in a software design. This increases the cache memory requirement dramatically.

[00136] Memory required for line-buffer

[00137] For blocks at CTU-row boundary, it is proposed to reduce the line-buffer memory required by affine inheritance. FIG. 20 shows an example when the current CU is at the CTU row boundary. Suppose the 4×4 block covering (x_{LE1}, y_{LE1}) is selected to be inherited the affine model from, then the neighbouring CU covering (x_{LE0}, y_{LE0}) is found. The MVs of the bottom-left 4×4 block and bottom right block of the neighbouring CU are found (noted as v_{LE0} and v_{LE1} in the figure). Then the CPMVs of the current block are calculated as

[00138] The control point vectors $\vec{v}_0$ and $\vec{v}_1$ of the current CU are derived by using the 4-parameter model, and by

$$\begin{cases} v_{0x} = \frac{(v_{LE1x} - v_{LE0x})}{(x_{LE1} - x_{LE0})}(x_0 - x_{LE0}) - \frac{(v_{LE1y} - v_{LE0y})}{(x_{LE1} - x_{LE0})}(y_0 - y_{LE0}) + v_{LE0x} \\ v_{0y} = \frac{(v_{LE1y} - v_{LE0y})}{(x_{LE1} - x_{LE0})}(x_0 - x_{LE0}) + \frac{(v_{LE1x} - v_{LE0x})}{(x_{LE1} - x_{LE0})}(y_0 - y_{LE0}) + v_{LE0y} \\ v_{1x} = \frac{(v_{LE1x} - v_{LE0x})}{(x_{LE1} - x_{LE0})}(x_1 - x_{LE0}) - \frac{(v_{LE1y} - v_{LE0y})}{(x_{LE1} - x_{LE0})}(y_1 - y_{LE0}) + v_{LE0x} \\ v_{1y} = \frac{(v_{LE1y} - v_{LE0y})}{(x_{LE1} - x_{LE0})}(x_1 - x_{LE0}) + \frac{(v_{LE1x} - v_{LE0x})}{(x_{LE1} - x_{LE0})}(y_1 - y_{LE0}) + v_{LE0y} \end{cases}$$

[00139] And if the current CU uses the 6-parameter affine motion model, the control point vectors $\vec{v}_2$ is derived by

$$\begin{cases} v_{2x} = \frac{(v_{LE1x} - v_{LE0x})}{(x_{LE1} - x_{LE0})}(x_2 - x_{LE0}) - \frac{(v_{LE1y} - v_{LE0y})}{(x_{LE1} - x_{LE0})}(y_2 - y_{LE0}) + v_{LE0x} \\ v_{2y} = \frac{(v_{LE1y} - v_{LE0y})}{(x_{LE1} - x_{LE0})}(x_2 - x_{LE0}) + \frac{(v_{LE1x} - v_{LE0x})}{(x_{LE1} - x_{LE0})}(y_2 - y_{LE0}) + v_{LE0y} \end{cases}$$

$$\begin{cases} x_{LE} = x_{E0} \\ x_{LE1} = x_{E1} \\ y_{LE} = y_0 \\ y_{LE} = y_1 \end{cases}$$

[00140] Since CPMVs of neighbouring blocks out of the current CTU row are not required, CPMVs are not required to be stored in line-buffer. Moreover, the height and y-component of the coordinate of the top-left corner are not required to be stored in line-buffer. However, the width

and the x-component of the coordinate of the top-left corner are still required to be stored in line-buffer.

[00141] FIG. 20 shows an example of affine inheritance at CTU-row.

[00142] For affine inheritance, it is proposed that the affine parameters a , b , c and d are stored for affine inheritance instead of storing CPMVs. For history motion vector prediction (HMVP), it is proposed that the affine parameters a , b , c and d can be stored to generate a history motion vector prediction (HMVP) for affine merge or affine inter coding. The buffer/table/list to store the history-based affine models is known as affine HMVP buffer/table/list.

3. Examples of problems solved by the embodiments and techniques described herein

[00143] In the current design, the extra storage required by affine inheritance is still a big issue.

[00144] Besides, there is a misalignment in terms of the MV utilized as being associated with one position but assumed to be associated with another position in the current design of affine inheritance crossing CTU rows.

[00145] As shown in FIG. 20, the control point vectors $\vec{v}_0$ and $\vec{v}_1$ of the current CU are derived by using the 4-parameter model, and by

$$\begin{cases} v_{0x} = \frac{(v_{LE1x} - v_{LE0x})}{(x_{LE1} - x_{LE0})}(x_0 - x_{LE0}) - \frac{(v_{LE1y} - v_{LE0y})}{(x_{LE1} - x_{LE0})}(y_0 - y_{LE0}) + v_{LE0x} \\ v_{0y} = \frac{(v_{LE1x} - v_{LE0x})}{(x_{LE1} - x_{LE0})}(x_0 - x_{LE0}) + \frac{(v_{LE1y} - v_{LE0y})}{(x_{LE1} - x_{LE0})}(y_0 - y_{LE0}) + v_{LE0y} \\ v_{1x} = \frac{(v_{LE1x} - v_{LE0x})}{(x_{LE1} - x_{LE0})}(x_1 - x_{LE0}) - \frac{(v_{LE1y} - v_{LE0y})}{(x_{LE1} - x_{LE0})}(y_1 - y_{LE0}) + v_{LE0x} \\ v_{1y} = \frac{(v_{LE1x} - v_{LE0x})}{(x_{LE1} - x_{LE0})}(x_1 - x_{LE0}) + \frac{(v_{LE1y} - v_{LE0y})}{(x_{LE1} - x_{LE0})}(y_1 - y_{LE0}) + v_{LE0y} \end{cases}$$

And if the current CU uses the 6-parameter affine motion model, the control point vectors $\vec{v}_2$ is derived by

$$\begin{cases} v_{2x} = \frac{(v_{LE1x} - v_{LE0x})}{(x_{LE1} - x_{LE})}(x_2 - x_{LE}) - \frac{(v_{LE1y} - v_{LE0y})}{(x_{LE1} - x_{LE0})}(y_2 - y_{LE0}) + v_{LE0x} \\ v_{2y} = \frac{(v_{LE1x} - v_{LE0x})}{(x_{LE1} - x_{LE0})}(x_2 - x_{LE}) + \frac{(v_{LE1y} - v_{LE0y})}{(x_{LE1} - x_{LE})}(y_2 - y_{LE}) + v_{LE0y} \end{cases}$$

$$\begin{cases} x_{LE} = x_{E0} \\ x_{LE1} = x_{E1} \\ y_{LE0} = y_0 \\ y_{LE} = y_1 \end{cases}$$

[00146] In the above equations, (x_{LE1}, y_{LE1}) and (x_{LE0}, y_{LE0}) are used as a representative position of the bottom-right and bottom-left sample coordinate of a coding unit with affine mode, respectively.

[00147] It is assumed that v_{LE0} is assigned to the left-bottom position of the neighbouring CU denoted as (x_{E0}, y_0) . However, v_{LE0} is assigned to the center position of the left-bottom sub-block of the neighbouring CU.

[00148] Additionally, it is required to know the coordinate of the affine coded CU as well as the width of the CU since the **two representative sub-blocks** covering the bottom-left and bottom-right samples of the CU need to be identified. In above examples, the sub-block covering (x_{LE0}, y_{LE0}) is named as the first representative sub-block and the sub-block covering (x_{LE1}, y_{LE1}) is named as the second representative sub-block. Therefore, additional line buffer is required to store the CU width, coordinate, etc.

4. Examples of embodiments and techniques

[00149] The detailed listing of items below should be considered as examples to explain general concepts. These embodiments should not be interpreted in a narrow way. Furthermore, these inventions can be combined in any manner.

[00150] In the following discussion, $SatShift(x, n)$ is defined as

$$SatShift(x, n) = \begin{cases} (x + offset0) \gg n & \text{if } x \geq 0 \\ -((-x + offset1) \gg n) & \text{if } x < 0 \end{cases}$$

$Shift(x, n)$ is defined as $Shift(x, n) = (x + offset0) \gg n$.

[00151] In one example, $offset0$ and/or $offset1$ are set to $(1 \ll n) \gg 1$ or $(1 \ll (n-1))$. In another example, $offset0$ and/or $offset1$ are set to 0.

$Clip3(\min, \max, x)$ is defined as

$$Clip3(\min, \max, x) = \begin{cases} \min & \text{if } x < \min \\ \max & \text{if } x > \max \\ x & \text{Otherwise} \end{cases}$$

[00152] Reducing the in-CTU storage required by affine inheritance.

1. A first group of CPMVs (denoted as MV^F0 , MV^F1 and MV^F2 , at represented points (x^F0, y^F0) , (x^F1, y^F1) and (x^F2, y^F2) , respectively) of a block used to conduct affine inheritance for following blocks may be different from a second group of CPMVs (denoted as MV^S0 , MV^S1 and MV^S2 , at represented points (x^S0, y^S0) , (x^S1, y^S1) and (x^S2, y^S2) , respectively) of a block used to derive the MVs for each sub-block, or a third group of CPMVs (denoted as MV^T0 , MV^T1 and MV^T2 , at represented points (x^T0, y^T0) , (x^T1, y^T1) and (x^T2, y^T2) , respectively) signaled from the encoder to the decoder.
 - a) In one example, the second group CPMVs are the same to the third group of CPMVs.
 - b) In one example, the first group of CPMVs are derived from the second or the third group of CPMVs.
 - c) In one example, the first group of CPMVs are stored after coding/decoder a block.
 - d) In one example, the represented points' coordinator (such as (x^Fi, y^Fi) , (x^Si, y^Si) , (x^Ti, y^Ti)) are defined as coordinators relative to one sub-block which is used in the affine motion compensation process.
2. The relative offset between the representative points of two CPMVs in the first group may not depend on the width or height of the block.
 - a) In one example, CPMVs of a block B denoted as MV^F0 , MV^F1 and MV^F2 at representative points (x^F0, y^F0) , (x^F1, y^F1) and (x^F2, y^F2) , respectively, are stored. In another example, CPMVs of a block B denoted as MV^F0 and MV^F1 at positions (x^F0, y^F0) and (x^F1, y^F1) are stored. (x^F0, y^F0) , (x^F1, y^F1) and (x^F2, y^F2) may be inside block B, or they be outside of it.
 - b) In one example, $y^F1 = y^F0$, $x^F1 = x^F0 + PW$.
 - c) In one example, $x^F1 = x^F0$, $y^F1 = y^F0 + PH$.
 - d) In one example, $y^F2 = y^F0$, $x^F2 = x^F0 + PW$.
 - e) In one example, $x^F2 = x^F0$, $y^F2 = y^F0 + PH$.
 - f) In one example, $y^F2 = y^F1$, $x^F2 = x^F1 + PW$.
 - g) In one example, $x^F2 = x^F1$, $y^F2 = y^F1 + PH$.
 - h) PW and PH are integers.
 - i. In one example, $PW = 2^M$. For example, PW may be equal to 4, 8 16, 32, 64 or 128.

- ii. In one example, $PW = -2^M$. For example, PW may be equal to -4, -8 -16, -32, -64 or -128.
- iii. In one example, $PH = 2^M$. For example, PH may be equal to 4, 8 16, 32, 64 or 128.
- iv. In one example, $PH = -2^M$. For example, PH may be equal to -4, -8 -16, -32, -64 or -128.
- v. PW and PH are not stored.
 - 1. In one example, they are fixed.
 - 2. In another example, they are signaled in VPS/SPS/PPS/Slice header/tile group header/tile/CTU.
 - 3. In another example, they may be different in different standard profiles/levels/tiers.
 - 4. In another example, they may depend on the maximum CU size or/and minimum CU size of the slice/picture.
- i) In one example, $MV^{F0} = MV^{S0}$, $(x^{F0}, y^{F0}) = (x^{S0}, y^{S0})$, or $MV^{F0} = MV^{T0}$, $(x^{F0}, y^{F0}) = (x^{T0}, y^{T0})$;
- j) In one example, $MV^{F0} = MV^{S1}$, $(x^{F0}, y^{F0}) = (x^{S1}, y^{S1})$, or $MV^{F0} = MV^{T1}$, $(x^{F0}, y^{F0}) = (x^{T1}, y^{T1})$;
- k) In one example, $MV^{F0} = MV^{S2}$, $(x^{F0}, y^{F0}) = (x^{S2}, y^{S2})$, or $MV^{F0} = MV^{T2}$, $(x^{F0}, y^{F0}) = (x^{T2}, y^{T2})$;
- l) In one example, MV^{F0} , MV^{F1} and MV^{F2} are derived from MV^{S0} and MV^{S1} by Eq. (1) with (x^{F0}, y^{F0}) , (x^{F1}, y^{F1}) and (x^{F2}, y^{F2}) as the input coordinates.
- m) In one example, MV^{F0} , MV^{F1} and MV^{F2} are derived from MV^{S0} , MV^{S1} and MV^{S2} by Eq. (2) with (x^{F0}, y^{F0}) , (x^{F1}, y^{F1}) and (x^{F2}, y^{F2}) as the input coordinates.
- n) In one example, MV^{F0} , MV^{F1} and MV^{F2} are derived from MV^{T0} and MV^{T1} by Eq. (1) with (x^{F0}, y^{F0}) , (x^{F1}, y^{F1}) and (x^{F2}, y^{F2}) as the input coordinates.
- o) In one example, MV^{F0} , MV^{F1} and MV^{F2} are derived from MV^{T0} , MV^{T1} and MV^{T2} by Eq. (2) with (x^{F0}, y^{F0}) , (x^{F1}, y^{F1}) and (x^{F2}, y^{F2}) as the input coordinates.
- p) In one example, MV^{F2} is only calculated if the current block is coded with the 6-parameter affine model.

- i. Alternatively, MV^F2 is calculated no matter the current block is coded with the 6-parameter affine model or the 6-parameter affine model.
3. The difference between CPMVs are stored, instead of storing the CPMVs themselves.
 - a) In one example, $D1 = MV^F1 - MV^F0$ is stored;
 - b) In one example, $D2 = MV^F2 - MV^F0$ is stored;
 - c) In one example, both D1 and D2 are stored;
 - i. In one example, D2 is stored only if the current block is coded with the 6-parameter affine model.
 - ii. Alternatively, D2 is stored no matter the current block is coded with the 6-parameter affine model or the 6-parameter affine model.
 - d) In one example, the CPMVs and the differences between CPMVs may be stored together. For example, MV^F0 , D1 and D2 are stored.
4. The stored CPMVs or differences between CPMVs may be shifted before being stored. Suppose MV is a CPMV or the difference between CPMVs to be stored, then
 - a) For example, $MV'_x = SatShift(MV_x, n)$ and $MV'_y = SatShift(MV_y, n)$. $MV' = (MV'_x, MV'_y)$ is stored instead of MV.
 - b) For example, $MV'_x = Shift(MV_x, n)$ and $MV'_y = Shift(MV_y, n)$. $MV' = (MV'_x, MV'_y)$ is stored instead of MV.
 - c) For example, n is an integer such as 2 or 4;
 - i. In one example, n depends on the motion precision.
 - ii. n may be different when CPMV is stored or the difference between CPMVs is stored.
 - d) In one example, the stored MV' is left shift first before it is used in the affine inheritance.
5. The CPMVs or differences between CPMVs to be stored may be clipped before being stored. Suppose MV is a CPMV or the difference between CPMVs to be stored, then
 - a) $MV'_x = Clip3(MinV, MaxV, MV_x)$ and $MV'_y = Clip3(MinV, MaxV, MV_y)$. $MV' = (MV'_x, MV'_y)$ is stored instead of MV.
 - b) In one example, MV is stored with K bits, then $MinV = -2^{K-1}$, $MaxV = 2^{K-1} - 1$. For example, $MinV = -128$, $MaxV = 127$ when $K = 8$.

- i. K may be different depending on whether MV is a CPMV or a difference between CPMVs.
 - c) In one example, the stored MV' is first shifted, then clipped before it is used in the affine inheritance.
6. The MV stored in an adjacent neighbouring basic block denoted as Badj is used to derive the affine inherited MVs of the current block.
- a) FIG. 21 shows examples of MV stored in adjacent neighbouring basic blocks: L, A, LB, AR and AL. In VTM, a basic block is a 4×4 block.
 - b) The MV stored in an adjacent neighbouring basic block Badj, is denoted as MVa = (mv_a^h, mv_a^v) , then an affine inherited MV of the current block at position (x, y) denoted as $(mv^h(x, y), mv^v(x, y))$ is derived as

$$\begin{cases} mv^h(x, y) = a(x - x_0) - b(y - y_0) + mv_a^h \\ mv^v(x, y) = b(x - x_0) + a(y - y_0) + mv_a^v \end{cases} \quad (3)$$

or

$$\begin{cases} mv^h(x, y) = a(x - x_0) + c(y - y_0) + mv_a^h \\ mv^v(x, y) = b(x - x_0) + d(y - y_0) + mv_a^v \end{cases} \quad (4)$$

where (x_0, y_0) is the representative point of MVa.

- i. (x_0, y_0) may be any position inside the basic block Badj. FIG. 22 shows an example. (x_0, y_0) may be any one of Pij with $i=0 \dots Wb-1, j=0 \dots Hb-1$, where Wb and Hb are the width and height of the basic block. In the example, $Wb=Hb=4$. In an example, suppose coordinate of the top-left corner sample in Badj is (x_{TL}, y_{TL}) , then (x_0, y_0) may be any one of $(x_{TL}+i, y_{TL}+j)$ with $i=0 \dots Wb-1, j=0 \dots Hb-1$.
 - 1. For example, (x_0, y_0) may be P22 in FIG. 22.
 - 2. Suppose the coordinate of top-left sample of the current block is (x_{Pos00}, y_{Pos00}) , the coordinate of top-right sample of the current block is (x_{Pos10}, y_{Pos00}) , the coordinate of top-right sample of the current block is (x_{Pos00}, y_{Pos01}) , then in FIG. 21:
 - a. (x_0, y_0) for adjacent neighbouring basic block L is $(x_{Pos00}-2, y_{Pos01}-1)$;
 - b. (x_0, y_0) for adjacent neighbouring basic block LB is $(x_{Pos00}-2, y_{Pos01}+3)$;

- c. (x_0, y_0) for adjacent neighbouring basic block A is $(xPos10-1, yPos00-2)$;
 - d. (x_0, y_0) for adjacent neighbouring basic block AR is $(xPos10+3, yPos00-2)$;
 - e. (x_0, y_0) for adjacent neighbouring basic block AL is $(xPos00-2, yPos00-2)$.
- ii. (x_0, y_0) may be any position outside or at the boundary of the basic block Badj.
 - 1. For example, suppose coordinate of the top-left corner sample in Badj is (x_{TL}, y_{TL}) , then (x_0, y_0) may be any one of $(x_{TL}+i, y_{TL}+j)$ with i may be $-1, 0, W_b-1$ or W_b ; j may be $-1, 0, H_b-1$ or H_b .
- iii. Suppose coordinate of the top-left corner sample in Badj is (x_{TL}, y_{TL}) and $(x_0, y_0) = (x_{TL}+i, y_{TL}+j)$.
 - 1. i and j may depend on the position of Badj. For example, $i = 0, j = H_b-1$ if Badj is block L in FIG. 21, but $i = W_b-1, j = 0$ if Badj is block A in FIG. 21.
 - 2. i and j may depend on the width and height of the current block.
 - 3. i and j may be signaled in VPS/SPS/PPS/Slice header/tile group header/tile/CTU/CU.
 - 4. In another example, i and j may be different in different standard profiles/levels/tiers.
- iv. The position (x, y) may be in a sub-block of the current block, then the MV of a sub-block is inherited depending on MVa.
- v. The position (x, y) may be a corner of the current block, then a CPMV of the current block is inherited depending on MVa.
 - 1. In one example, the inherited CPMVs can be used to predict the signaled CPMVs of the affine inter-coded current block.
 - 2. In one example, the inherited CPMVs can be directly used as CPMVs of the affine merge-coded current block.
- vi. In one example, Eq. (3) is applied if the current block uses the 4-paramter affine model. Eq. (4) is applied if the current block uses the 6-paramter affine model.

- vii. Alternatively, Eq. (4) is applied no matter the current block uses the 4-paramter affine model or the 6-paramter affine model.
- c) a , b , c and d in Eq. (3) and Eq. (4) are calculated as

$$\begin{cases} a = \frac{(mv_{t1}^h - mv_{t0}^h)}{w_t} \\ b = \frac{(mv_{t1}^v - mv_{t0}^v)}{w_t} \\ c = \frac{(mv_{t2}^h - mv_{t0}^h)}{h_t} \\ d = \frac{(mv_{t2}^v - mv_{t0}^v)}{h_t} \end{cases} \quad (5)$$

- i. In one example, a , b , c and d are derived from stored CPMVs in the second or third group, as declared in bullet 1, of the CU covering the adjacent neighbouring basic block Badj. Suppose the CU covering the adjacent neighbouring basic block Badj is block Z Then $mv_{t0} = (mv_{t0}^h, mv_{t0}^v)$, $mv_{t1} = (mv_{t1}^h, mv_{t1}^v)$ $mv_{t2} = (mv_{t2}^h, mv_{t2}^v)$ are the CPMVs in the second or third group of block Z. w_t and h_t are the width and height of block Z.
- ii. In one example, a , b , c and d are derived from stored CPMVs in the first group, as declared in bullet 1, of the CU covering the adjacent neighbouring basic block Badj. Suppose the CU covering the adjacent neighbouring basic block Badj is block Z Then $mv_{t0} = (mv_{t0}^h, mv_{t0}^v)$, $mv_{t1} = (mv_{t1}^h, mv_{t1}^v)$ $mv_{t2} = (mv_{t2}^h, mv_{t2}^v)$ are the CPMVs in the first group of block Z. w_t and h_t are PW and PH declared in bullet 2.
- iii. In one example, $mv_{t1}^h - mv_{t0}^h$, $mv_{t1}^v - mv_{t0}^v$, $mv_{t2}^h - mv_{t0}^h$, $mv_{t2}^v - mv_{t0}^v$, are fetched from the storage directly as claimed in bullet 3.

[00153] FIG. 21 shows examples of MV stored in adjacent neighbouring basic blocks.

[00154] FIG. 22 shows positions in a 4×4 basic block.

[00155] Reducing the line-buffer storage required by affine inheritance.

7. A first MV stored in a first basic block adjacently neighbouring to the current block, and a second MV stored in a second basic block with a known offset to the first basic block, are used to derive the CPMVs of the current block.

- a) In one example, the MV stored in the first adjacent neighbouring basic block is denoted as $MV_a = (mv_a^h, mv_a^v)$, then an affine inherited MV of the current block at position (x, y) denoted as $(mv^h(x, y), mv^v(x, y))$ is derived by Eq. (3), and a, b are derived by Eq. (5).

- i. In one example, mv_{t0} and mv_{t1} in Eq. (5) are set equal to the MV stored in the first basic block and the MV stored in the second basic block, respectively. w_t is set to be the horizontal offset between the two basic blocks.

- b) Alternatively, a, b are derived as

$$\begin{cases} a = \frac{(mv_{t1}^v - mv_{t0}^v)}{h_t} \\ b = -\frac{(mv_{t1}^h - mv_{t0}^h)}{h_t} \end{cases} \quad (6)$$

where h_t is set to be the horizontal offset between the two basic blocks.

- c) For example, w_t and h_t must be in a form of 2^N , such as 4, 8, 16..
- d) Suppose (x_{LT0}, y_{LT0}) and (x_{LT1}, y_{LT1}) represent the coordinates of the top-left sample of the first basic block and the second basic block, respectively, then
- i. The horizontal offset between the first basic block and the second basic block is defined as $x_{LT1} - x_{LT0}$;
- ii. The vertical offset between the first basic block and the second basic block is defined as $y_{LT1} - y_{LT0}$;
- e) In one example, $y_{LT1} - y_{LT0}$ must be equal to 0 when the first basic block is above the current block (such as block A, AL and AR in FIG. 23).
- f) In one example, $x_{LT1} - x_{LT0}$ must be equal to 0. when the first basic block is left to the current block (such as block L, LB and AL in FIG. 23).
- g) How to choose the second basic block may depend on the position of the first block.
- i. For example, $y_{LT1} = y_{LT0}$ and $x_{LT1} = x_{LT0} - offset$, where *offset* is a positive number such as 4, 8 or 16, if $x_{LT0} \% M$ is not equal to 0, e.g., $M = 8$ and *offset* = 4.

- ii. For example, $y_{LT1} = y_{LT0}$ and $y_{LT1} = y_{LT0} + offset$, where *offset* is a positive number such as 4, 8 or 16, if $x_{LT0} \% M$ is equal to 0, e.g., $M = 8$ and *offset* = 4.
- iii. For example, $y_{LT1} = y_{LT0}$ and $x_{LT1} = x_{LT0} + offset$, where *offset* is a positive number such as 4, 8 or 16, if $x_{LT0} \% M$ is equal to 0, e.g., $M = 8$ and *offset* = 4.
- iv. For example, $x_{LT1} = x_{LT0}$ and $y_{LT1} = y_{LT0} - offset$, where *offset* is a positive number such as 4, 8 or 16, if $x_{LT0} \% M$ is not equal to 0, e.g., $M = 8$ and *offset* = 4.
- v. For example, $x_{LT1} = x_{LT0}$ and $y_{LT1} = y_{LT0} + offset$, where *offset* is a positive number such as 4, 8 or 16, if $x_{LT0} \% M$ is equal to 0, e.g., $M = 8$ and *offset* = 4.
- vi. FIG. 23 shows examples of pairs of the first and second basic blocks: AL and AL', A and A', AR and AR', L and L', LB and LB'.
- vii. For example, $y_{LT1} = y_{LT0}$ and $x_{LT1} = x_{LT0} + offset$, where *offset* is a positive number such as 4, 8 or 16, if the first block is above-left to the current block (such as block AL in FIG. 23).
- viii. For example, $y_{LT1} = y_{LT0}$ and $x_{LT1} = x_{LT0} - offset$, where *offset* is a positive number such as 4, 8 or 16, if the first block is above-right to the current block (such as block AR in FIG. 23).
- ix. For example, $y_{LT1} = y_{LT0}$ and $x_{LT1} = x_{LT0} + offset$, where *offset* is a positive number such as 4, 8 or 16, if the first block is above-left to the current block (such as block AL in FIG. 23) and left boundary of the current block is also the left boundary of a CTU.
- x. For example, $y_{LT1} = y_{LT0}$ and $x_{LT1} = x_{LT0} - offset$, where *offset* is a positive number such as 4, 8 or 16, if the first block is above-right to the current block (such as block AR in FIG. 23) and the right boundary of the current block is also the right boundary of a CTU.
- xi. For example, $y_{LT1} = y_{LT0}$ and $x_{LT1} = x_{LT0} + offset$, where *offset* is a positive number such as 4, 8 or 16, if $x_{LT0} - offset < x_{LT_AL}$, where

x_{LT_AL} is the top-left coordinate of the neighbouring basic block above-left to the current block (such as block AL in FIG. 23).

- xii. For example, $y_{LT1} = y_{LT0}$ and $x_{LT1} = x_{LT0} - offset$, where *offset* is a positive number such as 4, 8 or 16, if $x_{LT0} + offset > x_{RT_AR}$, where x_{RT_AR} is the top-right coordinate of the neighbouring basic block above-right to the current block (such as block AR in FIG. 23).
 - xiii. For example, $y_{LT1} = y_{LT0}$ and $x_{LT1} = x_{LT0} + offset$, where *offset* is a positive number such as 4, 8 or 16, if $x_{LT0} - offset < x_{LT_AL}$, where x_{LT_AL} is the top-left coordinate of the neighbouring basic block above-left to the current block (such as block AL in FIG. 23) and left boundary of the current block is also the left boundary of a CTU.
 - xiv. For example, $y_{LT1} = y_{LT0}$ and $x_{LT1} = x_{LT0} - offset$, where *offset* is a positive number such as 4, 8 or 16, if $x_{LT0} + offset > x_{RT_AR}$, where x_{RT_AR} is the top-right coordinate of the neighbouring basic block above-right to the current block (such as block AR in FIG. 23) and the right boundary of the current block is also the right boundary of a CTU.
- h) The second basic block may be selected from several candidate basic blocks.
- i. For example, the top-left positions of the M candidate basic block are denoted as (x_{C0}, y_{C0}) , (x_{C1}, y_{C1}) , ... $(x_{C_{M-1}}, y_{C_{M-1}})$. The M candidate basic block are checked in order, to find the one that is inter-coded, and has a MV referring to the same reference picture as the MV of the first basic block referring to. The found candidate is selected as the second basic block.
 - 1. In one example, $M = 2$. $y_{C0} = y_{LT0}$, $x_{C0} = x_{LT0} - offset$, $y_{C1} = y_{LT0}$, $x_{C1} = x_{LT0} - offset$, where *offset* is a positive number such as 4, 8 or 16.
 - 2. In one example, $M = 2$. $x_{C0} = x_{LT0}$, $y_{C0} = y_{LT0} - offset$, $x_{C1} = x_{LT0}$, $y_{C1} = y_{LT0} - offset$, where *offset* is a positive number such as 4, 8 or 16.
 - 3. In one example, $M = 2$. $y_{C0} = y_{LT0}$, $x_{C0} = x_{LT0} - offset$, $y_{C1} = y_{LT0}$, $x_{C1} = x_{LT0} + offset$, where *offset* is a positive number such as 4, 8 or 16.

4. 4) In one example, $M = 2$. $x_{C0} = x_{LT0}$, $y_{C0} = y_{LT0} - \text{offset}$, $x_{C1} = x_{LT0}$, $y_{C1} = y_{LT0} + \text{offset}$, where *offset* is a positive number such as 4, 8 or 16.
- ii. Whether to and/or how to select the second basic block from candidate basic blocks may depend on the position of the first basic block and/or the position of the current block.
 1. For example, $M = 1$. $y_{C0} = y_{LT0}$, $x_{C0} = x_{LT0} + \text{offset}$, where *offset* is a positive number such as 4, 8 or 16, if the first block is above-left to the current block (such as block AL in FIG. 23).
 - a. For example, in other cases, $M = 2$. $y_{C0} = y_{LT0}$, $x_{C0} = x_{LT0} - \text{offset}$, $y_{C1} = y_{LT0}$, $x_{C1} = x_{LT0} + \text{offset}$, where *offset* is a positive number such as 4, 8 or 16.
 2. For example, $M = 1$. $y_{C0} = y_{LT0}$, $x_{C0} = x_{LT0} - \text{offset}$, where *offset* is a positive number such as 4, 8 or 16, if the first block is above-right to the current block (such as block AR in FIG. 23).
 - a. For example, in other cases, $M = 2$. $y_{C0} = y_{LT0}$, $x_{C0} = x_{LT0} - \text{offset}$, $y_{C1} = y_{LT0}$, $x_{C1} = x_{LT0} + \text{offset}$, where *offset* is a positive number such as 4, 8 or 16.
 3. For example, $M = 1$. $y_{C0} = y_{LT0}$, $x_{C0} = x_{LT0} + \text{offset}$, where *offset* is a positive number such as 4, 8 or 16, if the first block is above-left to the current block (such as block AL in FIG. 23) and left boundary of the current block is also the left boundary of a CTU.
 - a. For example, in other cases, $M = 2$. $y_{C0} = y_{LT0}$, $x_{C0} = x_{LT0} - \text{offset}$, $y_{C1} = y_{LT0}$, $x_{C1} = x_{LT0} + \text{offset}$, where *offset* is a positive number such as 4, 8 or 16.
 4. For example, $M = 1$. $y_{C0} = y_{LT0}$, $x_{C0} = x_{LT0} - \text{offset}$, where *offset* is a positive number such as 4, 8 or 16, if the first block is above-right to the current block (such as block AR in FIG. 23) and the right boundary of the current block is also the right boundary of a CTU.
 - a. For example, in other cases, $M = 2$. $y_{C0} = y_{LT0}$, $x_{C0} = x_{LT0} - \text{offset}$, $y_{C1} = y_{LT0}$, $x_{C1} = x_{LT0} + \text{offset}$, where *offset* is a positive number such as 4, 8 or 16.
 5. For example, $M = 1$. $y_{C0} = y_{LT0}$, $x_{C0} = x_{LT0} + \text{offset}$, where *offset* is a positive number such as 4, 8 or 16, if $x_{LT0} - \text{offset} < x_{LT_AL}$, where x_{LT_AL} is the top-left coordinate of the neighbouring basic block above-left to the current block (such as block AL in FIG. 23).
 - a. For example, in other cases, $M = 2$. $y_{C0} = y_{LT0}$, $x_{C0} = x_{LT0} - \text{offset}$, $y_{C1} = y_{LT0}$, $x_{C1} = x_{LT0} + \text{offset}$, where *offset* is a positive number such as 4, 8 or 16.
 6. For example, $M = 1$. $y_{C0} = y_{LT0}$, $x_{C0} = x_{LT0} - \text{offset}$, where *offset* is a positive number such as 4, 8 or 16, if $x_{LT0} + \text{offset} > x_{RT_AR}$, where x_{RT_AR} is the top-right coordinate of the neighbouring basic block above-right to the current block (such as block AR in FIG. 23).

- a. For example, in other cases, $M = 2$. $yC_0 = yLT_0$, $xC_0 = xLT_0 - offset$, $yC_1 = yLT_0$, $xC_1 = xLT_0 + offset$, where *offset* is a positive number such as 4, 8 or 16.
- 7. For example, $M = 1$. $yC_0 = yLT_0$, $xC_0 = xLT_0 + offset$, where *offset* is a positive number such as 4, 8 or 16, if $xLT_0 - offset < xLT_AL$, where xLT_AL is the top-left coordinate of the neighbouring basic block above-left to the current block (such as block AL in FIG. 23) and left boundary of the current block is also the left boundary of a CTU.
 - a. For example, in other cases, $M = 2$. $yC_0 = yLT_0$, $xC_0 = xLT_0 - offset$, $yC_1 = yLT_0$, $xC_1 = xLT_0 + offset$, where *offset* is a positive number such as 4, 8 or 16.
- 8. For example, $M = 1$. $yC_0 = yLT_0$, $xC_0 = xLT_0 - offset$, where *offset* is a positive number such as 4, 8 or 16, if $xLT_0 + offset > xRT_AR$, where xRT_AR is the top-right coordinate of the neighbouring basic block above-right to the current block (such as block AR in FIG. 23) and the right boundary of the current block is also the right boundary of a CTU.
 - a. For example, in other cases, $M = 2$. $yC_0 = yLT_0$, $xC_0 = xLT_0 - offset$, $yC_1 = yLT_0$, $xC_1 = xLT_0 + offset$, where *offset* is a positive number such as 4, 8 or 16.
- i) If a second basic block, which is inter-coded and has a MV referring to the same reference picture as the MV of the first basic block referring to, cannot be found, then affine inheritance cannot be conducted from the first basic block.
- j) In one example, whether and how to apply the methods in this bullet may depend on the position of the current block.
 - i. For example, the methods in this bullet are applied only when the affine model is inherited from an above neighbouring block and it is not in the current CTU or CTU row.
 - ii. For example, the methods in this bullet are applied only when the affine model is inherited from an above or left neighbouring block and it is not in the current CTU.
- k) It is proposed to identify the first representative sub-block by the sub-block where the affine model inheritance is derived from. In this case, the motion vector associated with one block where the affine model inheritance is derived from is utilized as (mv_a^h, mv_a^v) in Eq. (3).
- l) An embodiment is proposed as shown in FIG. 24.

- i. If the current block inherits affine model from a basic block BB (BB may be A, AR or AL, the top-left position of BB is (xBB, yBB), BB must be affine coded), then BB is treated as the first adjacent neighboring basic block. And the following process applies to find the second adjacent neighboring basic block.
 If $x_{BB} \% 8 = 0$, then the basic block BBR right to BB is first checked (If BB is A, BBR is AR; If BB is AR, BBR is AR'; If BB is AL, BBR is AL'');
 Otherwise ($x_{BB} \% 8 \neq 0$), the basic block BBL left to BB (If BB is A, BBL is A'; If BB is AR, BBL is A; If BB is AL, BBL is AL') is first checked.
 - ii. When BBR is first checked, if BBR is affine coded, and it has the same reference index as BB for a given reference list, then BBR is treated as the second adjacent neighboring basic block. Otherwise, BBL is treated as the second adjacent neighboring basic block.
 - iii. When BBL is first checked, if BBL is affine coded, and it has the same reference index as BB for a given reference list, then BBL is treated as the second adjacent neighboring basic block. Otherwise, BBR is treated as the second adjacent neighboring basic block.
- m) For example, whether to find the second block from multiple candidates or from a predefined offset may depend on the position of the first block and/or the position of the current block.
- i. An embodiment is proposed as shown in FIG. 24.
 1. If the current block inherits affine model from a basic block BB (BB may be A, AR or AL, the top-left position of BB is (xBB, yBB), BB must be affine coded), then BB is treated as the first adjacent neighbouring basic block. And the following process applies to find the second adjacent neighbouring basic block.
 - a. If BB is AL and the left boundary of the current block is the left boundary of a CTU, then only the basic block BBR right to BB (AL'') is checked. if BBR is affine coded, and it has the same reference index as BB for a given reference list, then BBR is treated as the second adjacent neighbouring basic block. Otherwise, the affine model inherited from BB is unavailable.
 - b. Otherwise, if $x_{BB} \% S = 0$, then the basic block BBR right to BB is first checked (If BB is A, BBR is AR; If BB is AR, BBR is AR'; If BB is AL, BBR is AL''); Otherwise ($x_{BB} \% S \neq 0$), the basic block BBL left to BB (If BB is A,

BBL is A'; If BB is AR, BBL is A; If BB is AL, BBL is AL') is first checked. For example, S is equal to 8.

- i. When BBR is first checked, if BBR is affine coded, and it has the same reference index as BB for a given reference list, then BBR is treated as the second adjacent neighbouring basic block. Otherwise, BBL is treated as the second adjacent neighbouring basic block.
 - ii. When BBL is first checked, if BBL is affine coded, and it has the same reference index as BB for a given reference list, then BBL is treated as the second adjacent neighbouring basic block. Otherwise, BBR is treated as the second adjacent neighbouring basic block.
- n) The *offset* between the first and second basic unit disclosed in this document is a positive integer.
- i. In one example, *offset* must be in a form of 2^K .
 - ii. In one example, *offset* may depend on the minimum allowed CU width.
 - iii. In one example, *offset* may depend on the minimum allowed CU height.
 - iv. In one example, *offset* may depend on the basic block width.
 - v. In one example, *offset* may depend on the basic block height.
 - vi. In one example, *offset* may depend on the minimum allowed width of a CU that affine coding is applicable.
 - vii. In one example, *offset* may depend on the minimum allowed height of a CU that affine coding is applicable.
 - viii. In one example, *offset* may be signaled from the encoder to the decoder.
- o) If a basic block P is chosen as the second block when a basic block Q is the first block, then Q is not allowed to be chosen as the second block when P is the first block.
- i. In one example, P is not allowed to be the first block.
 - ii. In one example, when P is the first block, the second block can only be chosen from a basic unit left to P if Q is right to P.
 - iii. In one example, when P is the first block, the second block can only be chosen from a basic unit right to P if Q is left to P.
 - iv. In one example, when P is the first block, the second block can only be chosen from a basic unit above to P if Q is below to P.
 - v. In one example, when P is the first block, the second block can only be chosen from a basic unit below to P if Q is above to P.
- p) In one example, an adjacent neighbouring basic block may be on a row or column adjacent to the current block. For example, in FIG. 23, AL' AR', LB' may also be regarded as adjacent neighbouring blocks.
- q) In one example, a first basic block is considered as "valid" if it satisfies one, several or all of the following conditions:
- i. It is inter-coded;
 - ii. It is not intra-block-copy coded;

- iii. It is affine-coded;
 - iv. It is affine-merge coded;
 - v. It is affine-inter coded;
- r) In one example, whether a second basic block is considered as “valid” or not may depend on the information of the first basic block.
- s) In one example, a second basic block is considered as “valid” if it satisfies one, several or all of the following conditions:
- i. It is inter-coded;
 - ii. It is not intra-block-copy coded;
 - iii. It is affine-coded;
 - iv. It is affine-merge coded;
 - v. It is affine-inter coded;
 - vi. It has the same inter-prediction direction as the first basic block;
 - vii. It has the same reference index for reference list 0 as the first basic block;
 - viii. It has the same reference index for reference list 1 as the first basic block;
 - ix. It has the same inter-prediction direction and same reference indices as the first basic block;
 - x. It has the same picture-order-count (POC) value of the reference picture in reference list X (X being 0 and/or 1) as that for the first basic block;
- t) In one example, each candidate above neighbouring basic block such as AR, A and AL1 in FIG. 24, is checked in order to determine whether it is a valid first basic block. For example, the order may be AR, A, AL1 or A, AR AL1.
- i. In one example, if one basic block BB is checked (BB may be AR, A or AL1) and BB is a valid first basic block, then the basic block left to it and/or the basic block right to it are checked in order to find the corresponding second basic block.
 - 1. In one example, the basic block BBR right to BB is checked first. An example of the detailed steps for the determination of first and second basic blocks are given as follows:
 - If BBR is a valid second basic block, then BB and BBR are output as the first basic block and second basic block;
 - Otherwise (BBR is not valid), the basic block BBL left to BB is checked;
 - i. If BBL is a valid second basic block, then BB and BBL are output as the first basic block and second basic block;
 - ii. Otherwise (BBL is not valid), then next basic block in order is checked to be the first basic block. In an alternative example, no valid first basic block and second basic block can be output.
 - 2. Alternatively, the basic block BBL left to BB is checked first. An example of the detailed steps for the determination of first and second basic blocks are given as follows:
 - If BBL is a valid second basic block, then BB and BBL are output as the first basic block and second basic block;

- Otherwise (BBL is not valid), the basic block BBR right to BB is checked;
 - i. If BBR is a valid second basic block, then BB and BBR are output as the first basic block and second basic block;
 - ii. Otherwise (BBR is not valid), then next basic block in order is checked determine whether it is a valid first basic block.
 - 1. Alternative, no valid first basic block and second basic block can be output from above neighbouring basic blocks.
 - ii. In one example, if one basic block BB is checked (BB may be AR, A or AL1) and BB is a valid first basic block, then only the basic block BBL left to it is checked to find the corresponding second basic block. An example of the detailed steps for the determination of first and second basic blocks are given as follows:
 - If BBL is a valid second basic block, then BB and BBL are output as the first basic block and second basic block;
 - Otherwise (BBL is not valid), then BB and the basic block BBR right to BB are output as the first basic block and second basic block;
 - a. Alternatively, BB is not used as the first basic block and the next basic block in order is checked to determine whether it is a valid first basic block.
 - b. Alternative, no valid first basic block and second basic block can be output from above neighbouring basic blocks.
 - iii. Alternatively, if basic block BB is checked (BB may be AR, A or AL1) and BB is a valid first basic block, then only the basic block BBR right to it is checked to find the corresponding second basic block. An example of the detailed steps for the determination of first and second basic blocks are given as follows:
 - If BBR is a valid second basic block, then BB and BBR are output as the first basic block and second basic block;
 - Otherwise (BBR is not valid), then BB and the basic block BBL left to BB are output as the first basic block and second basic block;
 - a. Alternatively, BB is not used as the first basic block and the next basic block in order is checked to determine whether it is a valid first basic block.
 - b. Alternative, no valid first basic block and second basic block can be output from above neighbouring basic blocks.
- FIG. 27 and FIG. 28 show two exemplary flowcharts of how to choose the first basic block and the second basic block.
- iv. In one example, which above neighbouring basic blocks may be checked for the determination of first basic blocks may depend on the position of the current block and/or sub-block sizes of affine motion compensation.

1. For example, if the current block is at the left boundary of a CTU, the candidate basic blocks are AR, A and AL'' in FIG. 24. For example, if $xPos00 \% CTU_W == 0$ ($xPos00$ is the top-left coordinate of the current block and CTU_W is the width of a CTU) the current block is at the left boundary of a CTU.
 - a. For example, if the current block is at the left boundary of a CTU, the candidate basic blocks are AR, A and AL''; Otherwise, the candidates are AR, A and AL1.
- v. Whether a basic block can be used as the first basic block may depend on the position of the current block.
 1. For example, if the current block is at the left boundary of a CTU, the basic block AL1 in FIG. 24 cannot be used as the first basic block.
 2. For example, if the current block is at the left boundary of a CTU, the basic block AL' in FIG. 24 cannot be used as the first basic block.
 3. For example, if the current block is at the right boundary of a CTU, the basic block AR in FIG. 24 cannot be used as the first basic block. For example, if $(xPos00+W) \% CTU_W == 0$ ($xPos00$ is the top-left coordinate of the current block, W is the width of the current block and CTU_W is the width of a CTU), the current block is at the right boundary of a CTU.
 4. For example, if the current block is at the right boundary of a CTU, the basic block AR' in FIG. 24 cannot be used as the first basic block.
- vi. Whether a basic block can be used as the second basic block may depend on the position of the current block.
 1. For example, if the current block is at the left boundary of a CTU, the basic block AL1 in FIG. 24 cannot be used as the second basic block.
 2. For example, if the current block is at the left boundary of a CTU, the basic block AL' in FIG. 24 cannot be used as the second basic block.
 3. For example, if the current block is at the right boundary of a CTU, the basic block AR in FIG. 24 cannot be used as the second basic block.
 4. For example, if the current block is at the right boundary of a CTU, the basic block AR' in FIG. 24 cannot be used as the second basic block.
- u) In one example, the first basic block and the second basic block may be exchanged.
 - i. In one example, the output first and second basic blocks are firstly exchanged and then utilized for decoding one block.
 - ii. Alternatively, the determination process of first and second basic blocks mentioned above may be exchanged.

[00156] FIG. 23 shows examples of MVs of two adjacent neighbouring blocks.

[00157] Extension on stored affine parameters

8. The stored affine parameters may be shifted before being stored. Suppose m (m may be a , b , c or d) is to be stored, then
- a) For example, $m' = \text{SatShift}(m, n)$. m' is stored instead of m .
 - b) For example, $m' = \text{Shift}(m, n)$. m' is stored instead of m .
 - c) For example, n is an integer such as 2 or 4;
 - i. In one example, n depends on the motion precision.
 - ii. In one example, n may be different for different affine parameters.
 - i. In one example, n may be signaled in VPS/SPS/PPS/Slice header/tile group header/tile/CTU/CU.
 - ii. In another example, n may be different in different standard profiles/levels/tiers.
 - d) In one example, the stored affine parameter is left shift first before it is used in the affine inheritance.
 - e) In one example, the stored m' is first shifted, then clipped before it is used in the affine inheritance.

[00158] Extension on affine HMVP

- 9. In one example, the CPMVs in the first group as disclosed in bullet 1 and bullet 2 may be stored into the affine HMVP buffer/table/list to represent one history-based candidate affine model.
- 10. In one example, the differences between CPMVs as disclosed in bullet 3 may be stored into the affine HMVP buffer/table/list to represent one history-based candidate affine model.
- 11. All methods proposed in this document such as bullet 4 and bullet 5 can be applied to the CPMVs or differences between CPMVs stored in the affine HMVP buffer/table/list.
- 12. All the methods disclosed in bullet 6 can be applied when CPMVs or differences between CPMVs do not come from a neighbouring block but from the affine history buffer/table/list.

[00159] More Accurate MV Position

- 13. It is proposed that the position of a MV used to conduct affine inheritance from one block should be aligned to the position that used to derive the MV for the block.

- a) In one example as shown in FIG. 20, the control point vectors v_0' and v_1' of the current CU are derived by using the 4-parameter model, and by

$$\begin{cases} v_{0x} = \frac{(v_{LE1x} - v_{LE0x})}{(x_{LE1} - x_{LE0})} (x_0 - (x_{LE0} + \text{off}x)) - \frac{(v_{LE1y} - v_{LE0y})}{(x_{LE1} - x_{LE0})} (y_0 - (y_{LE0} + \text{off}y)) + v_{LE0x} \\ v_{0y} = \frac{(v_{LE1y} - v_{LE0y})}{(x_{LE1} - x_{LE0})} (x_0 - (x_{LE0} + \text{off}x)) + \frac{(v_{LE1x} - v_{LE0x})}{(x_{LE1} - x_{LE0})} (y_0 - (y_{LE0} + \text{off}y)) + v_{LE0y} \\ v_{1x} = \frac{(v_{LE1x} - v_{LE0x})}{(x_{LE1} - x_{LE0})} (x_1 - (x_{LE0} + \text{off}x)) - \frac{(v_{LE1y} - v_{LE0y})}{(x_{LE1} - x_{LE0})} (y_1 - (y_{LE0} + \text{off}y)) + v_{LE0x} \\ v_{1y} = \frac{(v_{LE1y} - v_{LE0y})}{(x_{LE1} - x_{LE0})} (x_1 - (x_{LE0} + \text{off}x)) + \frac{(v_{LE1x} - v_{LE0x})}{(x_{LE1} - x_{LE0})} (y_1 - (y_{LE0} + \text{off}y)) + v_{LE0y} \end{cases}$$

And if the current CU uses the 6-parameter affine motion model, the control point vectors $\vec{v}_2$ is derived by

$$\begin{cases} v_{2x} = \frac{(v_{LE1x} - v_{LE0x})}{(x_{LE1} - x_{LE0})} (x_2 - (x_{LE0} + \text{off}x)) - \frac{(v_{LE1y} - v_{LE0y})}{(x_{LE1} - x_{LE0})} (y_2 - (y_{LE0} + \text{off}y)) + v_{LE0x} \\ v_{2y} = \frac{(v_{LE1y} - v_{LE0y})}{(x_{LE1} - x_{LE0})} (x_2 - (x_{LE0} + \text{off}x)) + \frac{(v_{LE1x} - v_{LE0x})}{(x_{LE1} - x_{LE0})} (y_2 - (y_{LE0} + \text{off}y)) + v_{LE0y} \end{cases}$$

$$\begin{cases} x_{LE0} = x_{E0} \\ x_{LE1} = x_{E1} \\ y_{LE0} = y_0 \\ y_{LE1} = y_1 \end{cases}$$

- i. For example, $\text{off}x=1$ and $\text{off}y=-1$;
- ii. For example, $\text{off}x=2$ and $\text{off}y=-2$;
- iii. For example, $\text{off}x=1$ and $\text{off}y=-2$;
- iv. For example, $\text{off}x=2$ and $\text{off}y=-1$;
- v. For example, $\text{off}x=1$ and $\text{off}y=-3$;
- vi. For example, $\text{off}x=2$ and $\text{off}y=-3$;

- b) In one example as shown in FIG. 20, the control point vectors $\vec{v}_0$ and $\vec{v}_1$ of the current CU are derived by using the 4-parameter model, and by

$$\begin{cases} v_{0x} = \frac{(v_{LE1x} - v_{LE0x})}{(x_{LE1} - x_{LE0})} (x_0 - (x_{LE1} + \text{off}x)) - \frac{(v_{LE1y} - v_{LE0y})}{(x_{LE1} - x_{LE0})} (y_0 - (y_{LE1} + \text{off}y)) + v_{LE1x} \\ v_{0y} = \frac{(v_{LE1y} - v_{LE0y})}{(x_{LE1} - x_{LE0})} (x_0 - (x_{LE1} + \text{off}x)) + \frac{(v_{LE1x} - v_{LE0x})}{(x_{LE1} - x_{LE0})} (y_0 - (y_{LE1} + \text{off}y)) + v_{LE1y} \end{cases}$$

$$\begin{cases} v_{1x} = \frac{(v_{LE1x} - v_{LE0x})}{(x_{LE1} - x_{LE0})} (x_1 - (x_{LE1} + \text{offx})) - \frac{(v_{LE1y} - v_{LE0y})}{(x_{LE1} - x_{LE0})} (y_1 - (y_{LE1} + \text{offy})) + v_{LE1x} \\ v_{1y} = \frac{(v_{LE1y} - v_{LE0y})}{(x_{LE1} - x_{LE0})} (x_1 - (x_{LE1} + \text{offx})) + \frac{(v_{LE1x} - v_{LE0x})}{(x_{LE1} - x_{LE0})} (y_1 - (y_{LE1} + \text{offy})) + v_{LE1y} \end{cases}$$

And if the current CU uses the 6-parameter affine motion model, the control point vectors $\vec{v}_2$ is derived by

$$\begin{cases} v_{2x} = \frac{(v_{LE1x} - v_{LE0x})}{(x_{LE1} - x_{LE0})} (x_2 - (x_{LE1} + \text{offx})) - \frac{(v_{LE1y} - v_{LE0y})}{(x_{LE1} - x_{LE0})} (y_2 - (y_{LE1} + \text{offy})) + v_{LE1x} \\ v_{2y} = \frac{(v_{LE1y} - v_{LE0y})}{(x_{LE1} - x_{LE0})} (x_2 - (x_{LE1} + \text{offx})) + \frac{(v_{LE1x} - v_{LE0x})}{(x_{LE1} - x_{LE0})} (y_2 - (y_{LE1} + \text{offy})) + v_{LE1y} \end{cases}$$

$$\begin{cases} x_{LE} = x_{E0} \\ x_{LE1} = x_{E1} \\ y_{LE0} = y_0 \\ y_{LE1} = y_1 \end{cases}$$

- i. For example, $\text{offx}=-1$ and $\text{offy}=-1$;
- ii. For example, $\text{offx}=-2$ and $\text{offy}=-2$;
- iii. For example, $\text{offx}=-1$ and $\text{offy}=-2$;
- iv. For example, $\text{offx}=-2$ and $\text{offy}=-1$;
- v. For example, $\text{offx}=-1$ and $\text{offy}=-3$;
- vi. For example, $\text{offx}=-2$ and $\text{offy}=-3$;

5. Embodiments

[00160] In one embodiment, bullet 6 is applied to conduct affine inheritance not at the CTU row boundary. An exemplary decoding process is specified as (the section numbers here refer to the current release of the VVC standard):

8.3.3.2 Derivation process for motion vectors and reference indices in subblock merge mode

...

2. When `sps_affine_enabled_flag` is equal to 1, the sample locations (x_{NbA0}, y_{NbA0}) , (x_{NbA1}, y_{NbA1}) , (x_{NbA2}, y_{NbA2}) , (x_{NbB0}, y_{NbB0}) , (x_{NbB1}, y_{NbB1}) , (x_{NbB2}, y_{NbB2}) , (x_{NbB3}, y_{NbB3}) , and the variables `numSbX` and `numSbY` are derived as follows:

$$(x_{A_0}, y_{A_0}) = (x_{Cb} - 2, y_{Cb} + cbHeight + 2) \quad (8-309)$$

$$(x_{A_1}, y_{A_1}) = (x_{Cb} - 2, y_{Cb} + cbHeight - 2) \quad (8-310)$$

$$(x_{A_2}, y_{A_2}) = (x_{Cb} - 1, y_{Cb}) \quad (8-311)$$

$$(x_{B_0}, y_{B_0}) = (x_{Cb} + cbWidth + 2, y_{Cb} - 2) \quad (8-312)$$

$$(x_{B_1}, y_{B_1}) = (x_{Cb} + cbWidth - 2, y_{Cb} - 2) \quad (8-313)$$

$$(x_{B_2}, y_{B_2}) = (x_{Cb} - 2, y_{Cb} - 2) \quad (8-314)$$

$$(x_{B_3}, y_{B_3}) = (x_{Cb}, y_{Cb} - 1) \quad (8-315)$$

$$numSbX = cbWidth >> 2 \quad (8-316)$$

$$numSbY = cbHeight >> 2 \quad (8-317)$$

3. When `sps_affine_enabled_flag` is equal to 1, the variable `availableFlagA` is set equal to FALSE and the following applies for (x_{NbA_k}, y_{NbA_k}) from (x_{NbA_0}, y_{NbA_0}) to (x_{NbA_1}, y_{NbA_1}) :
 - The availability derivation process for a block as specified in clause 6.4.X is invoked with the current luma location (x_{Curr}, y_{Curr}) set equal to (x_{Cb}, y_{Cb}) and the neighbouring luma location (x_{NbA_k}, y_{NbA_k}) as inputs, and the output is assigned to the block availability flag `availableAk`.
 - When `availableAk` is equal to TRUE and `MotionModelIdc[ xNbAk ][ yNbAk ]` is larger than 0 and `availableFlagA` is equal to FALSE, the following applies:
 - The variable `availableFlagA` is set equal to TRUE, `motionModelIdcA` is set equal to `MotionModelIdc[ xNbAk ][ yNbAk ]`, (x_{Nb}, y_{Nb}) is set equal to $(CbPosX[xNbA_k][yNbA_k], CbPosY[xNbA_k][yNbA_k])$, `nbW` is set equal to

$CbWidth[xNbA_k][yNbA_k]$, nbH is set equal to $CbHeight[xNbA_k][yNbA_k]$, and $numCpMv$ is set equal to $MotionModelIdc[xNbA_k][yNbA_k] + 1$.

- For X being replaced by either 0 or 1, the following applies:
 - When $PredFlagLX[xNbA_k][yNbA_k]$ is equal to 1, the derivation process for luma affine control point motion vectors from a neighbouring block as specified in clause 8.3.3.5 is invoked with the luma coding block location (xCb, yCb) , the luma coding block width and height $(cbWidth, cbHeight)$, the neighbouring luma coding block location (xNb, yNb) , the neighbouring sub-block center location $(xNbA_k, yNbA_k)$, the neighbouring luma coding block width and height (nbW, nbH) , and the number of control point motion vectors $numCpMv$ as input, the control point motion vector predictor candidates $cpMvLXA[cpIdx]$ with $cpIdx = 0 \dots numCpMv - 1$ as output.
 - The following assignments are made:

$$predFlagLXA = PredFlagLX[xNbA_k][yNbA_k] \quad (8-318)$$

$$refIdxLXA = RefIdxLX[xNbA_k][yNbA_k] \quad (8-319)$$

4. When $sps_affine_enabled_flag$ is equal to 1, the variable $availableFlagB$ is set equal to FALSE and the following applies for $(xNbB_k, yNbB_k)$ from $(xNbB_0, yNbB_0)$ to $(xNbB_2, yNbB_2)$:
 - The availability derivation process for a block as specified in clause 6.4.X is invoked with the current luma location $(xCurr, yCurr)$ set equal to (xCb, yCb) and the neighbouring luma location $(xNbB_k, yNbB_k)$ as inputs, and the output is assigned to the block availability flag $availableB_k$.
 - When $availableB_k$ is equal to TRUE and $MotionModelIdc[xNbB_k][yNbB_k]$ is larger than 0 and $availableFlagB$ is equal to FALSE, the following applies:
 - The variable $availableFlagB$ is set equal to TRUE, $motionModelIdcB$ is set equal to $MotionModelIdc[xNbB_k][yNbB_k]$, (xNb, yNb) is set equal to $(CbPosX[xNbAB][yNbB_k], CbPosY[xNbB_k][yNbB_k])$, nbW is set equal to

$CbWidth[xNbB_k][yNbB_k]$, nbH is set equal to $CbHeight[xNbB_k][yNbB_k]$, and $numCpMv$ is set equal to $MotionModelIdc[xNbB_k][yNbB_k] + 1$.

- For X being replaced by either 0 or 1, the following applies:
 - When $PredFlagLX[xNbB_k][yNbB_k]$ is equal to TRUE, the derivation process for luma affine control point motion vectors from a neighbouring block as specified in clause 8.3.3.5 is invoked with the luma coding block location (xCb, yCb) , the luma coding block width and height $(cbWidth, cbHeight)$, the neighbouring luma coding block location (xNb, yNb) , the neighbouring sub-block center location $(xNbB_k, yNbB_k)$, the neighbouring luma coding block width and height (nbW, nbH) , and the number of control point motion vectors $numCpMv$ as input, the control point motion vector predictor candidates $cpMvLXB[cpIdx]$ with $cpIdx = 0 \dots numCpMv - 1$ as output.
 - The following assignments are made:

8.3.3.5 Derivation process for luma affine control point motion vectors from a neighbouring block

Inputs to this process are:

- a luma location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
- two variables $cbWidth$ and $cbHeight$ specifying the width and the height of the current luma coding block,
- a luma location (xNb, yNb) specifying the top-left sample of the neighbouring luma coding block relative to the top-left luma sample of the current picture,
- a luma location $(xNbC, yNbC)$ specifying the center sample of the neighbouring luma coding sub-block relative to the top-left luma sample of the current picture,

- two variables nNbW and nNbH specifying the width and the height of the neighbouring luma coding block,
- the number of control point motion vectors numCpMv.

Output of this process are the luma affine control point vectors cpMvLX[cpIdx] with cpIdx = 0 .. numCpMv – 1 and X being 0 or 1.

The variable isCTUboundary is derived as follows:

- If all the following conditions are true, isCTUboundary is set equal to TRUE:
 - ((yNb + nNbH) % CtbSizeY) is equal to 0
 - yNb + nNbH is equal to yCb
- Otherwise, isCTUboundary is set equal to FALSE.

The variables log2NbW and log2NbH are derived as follows:

$$\log2NbW = \text{Log2}(nNbW) \quad (8-369)$$

$$\log2NbH = \text{Log2}(nNbH) \quad (8-370)$$

The variables mvScaleHor, mvScaleVer, dHorX and dVerX are derived as follows:

- If isCTUboundary is equal to TRUE, the following applies:

$$mvScaleHor = MvLX[xNb][yNb + nNbH - 1][0] << 7 \quad (8-371)$$

$$mvScaleVer = MvLX[xNb][yNb + nNbH - 1][1] << 7 \quad (8-372)$$

$$\begin{aligned} dHorX = & (MvLX[xNb + nNbW - 1][yNb + nNbH - 1][0] - MvLX[xNb][yNb + nNbH - 1][0]) \\ & << (7 - \log2NbW) \end{aligned} \quad (8-373)$$

$$\begin{aligned} dVerX = & (MvLX[xNb + nNbW - 1][yNb + nNbH - 1][1] - MvLX[xNb][yNb + nNbH - 1][1]) \\ & << (7 - \log2NbW) \end{aligned} \quad (8-374)$$

- Otherwise (isCTUboundary is equal to FALSE), the following applies:

$$mvScaleHor = MvLX[xNbC][yNbC][0][0] << 7 \quad (8-375)$$

$$mvScaleVer = MvLX[xNbC][yNbC][0][1] << 7 \quad (8-376)$$

$$\begin{aligned} dHorX = & (CpMvLX[xNb + nNbW - 1][yNb][1][0] - CpMvLX[xNb][yNb][0][\\ & 0]) \\ & << (7 - \log_2 NbW) \end{aligned} \quad (8-377)$$

$$\begin{aligned} dVerX = & (CpMvLX[xNb + nNbW - 1][yNb][1][1] - CpMvLX[xNb][yNb][0][\\ & 1]) \\ & << (7 - \log_2 NbW) \end{aligned} \quad (8-378)$$

The variables dHorY and dVerY are derived as follows:

- If isCTUboundary is equal to FALSE and MotionModelIdc[xNb][yNb] is equal to 2, the following applies:

$$\begin{aligned} dHorY = & (CpMvLX[xNb][yNb + nNbH - 1][2][0] - CpMvLX[xNb][yNb][2][0 \\ &]) \\ & << (7 - \log_2 NbH) \end{aligned} \quad (8-379)$$

$$\begin{aligned} dVerY = & (CpMvLX[xNb][yNb + nNbH - 1][2][1] - CpMvLX[xNb][yNb][2][1 \\ &]) \\ & << (7 - \log_2 NbH) \end{aligned} \quad (8-380)$$

- Otherwise (isCTUboundary is equal to TRUE or MotionModelIdc[xNb][yNb] is equal to 1), the following applies,

$$dHorY = - dVerX \quad (8-381)$$

$$dVerY = dHorX \quad (8-382)$$

The luma affine control point motion vectors cpMvLX[cpIdx] with cpIdx = 0 .. numCpMv - 1 and X being 0 or 1 are derived as follows:

- When isCTUboundary is equal to TRUE, yNb is set equal to yCb. Then xNbC is set equal to xNb, yNbC is set equal to yNb.
- The first two control point motion vectors cpMvLX[0] and cpMvLX[1] are derived as follows:

$$\text{cpMvLX}[0][0] = (\text{mvScaleHor} + \text{dHorX} * (\text{xCb} - \text{xNbC}) + \text{dHorY} * (\text{yCb} - \text{yNbC})) \quad (8-383)$$

$$\text{cpMvLX}[0][1] = (\text{mvScaleVer} + \text{dVerX} * (\text{xCb} - \text{xNbC}) + \text{dVerY} * (\text{yCb} - \text{yNbC})) \quad (8-384)$$

$$\text{cpMvLX}[1][0] = (\text{mvScaleHor} + \text{dHorX} * (\text{xCb} + \text{cbWidth} - \text{xNbC}) + \text{dHorY} * (\text{yCb} - \text{yNbC})) \quad (8-385)$$

$$\text{cpMvLX}[1][1] = (\text{mvScaleVer} + \text{dVerX} * (\text{xCb} + \text{cbWidth} - \text{xNbC}) + \text{dVerY} * (\text{yCb} - \text{yNbC})) \quad (8-386)$$

- If numCpMv is equal to 3, the third control point vector cpMvLX[2] is derived as follows:

$$\text{cpMvLX}[2][0] = (\text{mvScaleHor} + \text{dHorX} * (\text{xCb} - \text{xNbC}) + \text{dHorY} * (\text{yCb} + \text{cbHeight} - \text{yNbC})) \quad (8-387)$$

$$\text{cpMvLX}[2][1] = (\text{mvScaleVer} + \text{dVerX} * (\text{xCb} - \text{xNbC}) + \text{dVerY} * (\text{yCb} + \text{cbHeight} - \text{yNbC})) \quad (8-388)$$

- The rounding process for motion vectors as specified in clause 8.3.2.12 is invoked with mvX set equal to cpMvLX[cpIdx], rightShift set equal to 7, and leftShift set equal to 0 as inputs and the rounded cpMvLX[cpIdx] as output, with X being 0 or 1 and cpIdx = 0 .. numCpMv – 1.

...

8.3.3.7 Derivation process for luma affine control point motion vector predictors

Inputs to this process are:

...

1. The number of control point motion vector predictor candidates in the list numCpMvpCandLX is set equal to 0.
2. The variables availableFlagA and availableFlagB are both set equal to FALSE.
3. The sample locations (xNbA₀, yNbA₀), (xNbA₁, yNbA₁), (xNbA₂, yNbA₂), (xNbB₀, yNbB₀), (xNbB₁, yNbB₁), and (xNbB₂, yNbB₂) are derived as follows:

$$(x_{A_0}, y_{A_0}) = (x_{Cb} - 2, y_{Cb} + cbHeight + 2) \quad (8-440)$$

$$(x_{A_1}, y_{A_1}) = (x_{Cb} - 2, y_{Cb} + cbHeight - 2) \quad (8-441)$$

$$(x_{B_0}, y_{B_0}) = (x_{Cb} + cbWidth + 2, y_{Cb} - 2) \quad (8-442)$$

$$(x_{B_1}, y_{B_1}) = (x_{Cb} + cbWidth - 2, y_{Cb} - 2) \quad (8-443)$$

$$(x_{B_2}, y_{B_2}) = (x_{Cb} - 2, y_{Cb} - 2) \quad (8-444)$$

4. The following applies for (x_{NbA_k}, y_{NbA_k}) from (x_{NbA_0}, y_{NbA_0}) to (x_{NbA_1}, y_{NbA_1}) :

- The availability derivation process for a block as specified in clause 6.4.X [Ed. (BB): Neighbouring blocks availability checking process tbd] is invoked with the current luma location (x_{Curr}, y_{Curr}) set equal to (x_{Cb}, y_{Cb}) and the neighbouring luma location (x_{NbA_k}, y_{NbA_k}) as inputs, and the output is assigned to the block availability flag $availableA_k$.
- When $availableA_k$ is equal to TRUE and $MotionModelIdx[x_{NbA_k}][y_{NbA_k}]$ is larger than 0 and $availableFlagA$ is equal to FALSE, the following applies:
 - The variable (x_{Nb}, y_{Nb}) is set equal to $(CbPosX[x_{NbA_k}][y_{NbA_k}], CbPosY[x_{NbA_k}][y_{NbA_k}])$, nbW is set equal to $CbWidth[x_{NbA_k}][y_{NbA_k}]$, and nbH is set equal to $CbHeight[x_{NbA_k}][y_{NbA_k}]$.
 - If $PredFlagLX[x_{NbA_k}][y_{NbA_k}]$ is equal to 1 and $DiffPicOrderCnt(RefPicListX[RefIdxLX[x_{NbA_k}][y_{NbA_k}]], RefPicListX[refIdxLX])$ is equal to 0, the following applies:
 - The variable $availableFlagA$ is set equal to TRUE
 - The derivation process for luma affine control point motion vectors from a neighbouring block as specified in clause 8.3.3.5 is invoked with the luma

coding block location (x_{Cb} , y_{Cb}), the luma coding block width and height ($cbWidth$, $cbHeight$), the neighbouring luma coding block location (x_{Nb} , y_{Nb}), the neighbouring luma coding block width and height (nbW , nbH), and the number of control point motion vectors $numCpMv$ as input, the control point motion vector predictor candidates $cpMvpLX[cpIdx]$ with $cpIdx = 0 \dots numCpMv - 1$ as output.

- The rounding process for motion vectors as specified in clause 8.3.2.12 is invoked with mvX set equal to $cpMvpLX[cpIdx]$, $rightShift$ set equal to 2, and $leftShift$ set equal to 2 as inputs and the rounded $cpMvpLX[cpIdx]$ with $cpIdx = 0 \dots numCpMv - 1$ as output.

- The following assignments are made:

$$cpMvpListLX[numCpMvpCandLX][0] = cpMvpLX[0] \quad (8-445)$$

$$cpMvpListLX[numCpMvpCandLX][1] = cpMvpLX[1] \quad (8-446)$$

$$cpMvpListLX[numCpMvpCandLX][2] = cpMvpLX[2] \quad (8-447)$$

$$numCpMvpCandLX = numCpMvpCandLX + 1 \quad (8-448)$$

- Otherwise if $PredFlagLY[x_{NbA_k}][y_{NbA_k}]$ (with $Y = !X$) is equal to 1 and $DiffPicOrderCnt(RefPicListY[RefIdxLY[x_{NbA_k}][y_{NbA_k}]], RefPicListX[refIdxLX])$ is equal to 0, the following applies:

- The variable $availableFlagA$ is set equal to TRUE

- The derivation process for luma affine control point motion vectors from a neighbouring block as specified in clause 8.3.3.5 is invoked with the luma coding block location (x_{Cb} , y_{Cb}), the luma coding block width and height ($cbWidth$, $cbHeight$), the neighbouring luma coding block location (x_{Nb} , y_{Nb}), the neighbouring sub-block center location (x_{NbA_k} , y_{NbA_k}), the neighbouring luma coding block width and height (nbW , nbH), and the number of control point motion vectors $numCpMv$ as input, the control point motion vector predictor candidates $cpMvpLY[cpIdx]$ with $cpIdx = 0 \dots numCpMv - 1$ as output.

- The rounding process for motion vectors as specified in clause 8.3.2.12 is invoked with mvX set equal to cpMvpLY[cpIdx], rightShift set equal to 2, and leftShift set equal to 2 as inputs and the rounded cpMvpLY[cpIdx] with cpIdx = 0 .. numCpMv – 1 as output.
- The following assignments are made:

$$\text{cpMvpListLX}[\text{numCpMvpCandLX}][0] = \text{cpMvpLY}[0] \quad (8-449)$$

$$\text{cpMvpListLX}[\text{numCpMvpCandLX}][1] = \text{cpMvpLY}[1] \quad (8-450)$$

$$\text{cpMvpListLX}[\text{numCpMvpCandLX}][2] = \text{cpMvpLY}[2] \quad (8-451)$$

$$\text{numCpMvpCandLX} = \text{numCpMvpCandLX} + 1 \quad (8-452)$$

5. The following applies for (xNbB_k, yNbB_k) from (xNbB₀, yNbB₀) to (xNbB₂, yNbB₂):

- The availability derivation process for a block as specified in clause 6.4.X [Ed. (BB): Neighbouring blocks availability checking process tbd] is invoked with the current luma location (xCurr, yCurr) set equal to (xCb, yCb) and the neighbouring luma location (xNbB_k, yNbB_k) as inputs, and the output is assigned to the block availability flag availableB_k.
- When availableB_k is equal to TRUE and MotionModelIdc[xNbB_k][yNbB_k] is larger than 0 and availableFlagB is equal to FALSE, the following applies:
 - The variable (xNb, yNb) is set equal to (CbPosX[xNbB_k][yNbB_k], CbPosY[xNbB_k][yNbB_k]), nbW is set equal to CbWidth[xNbB_k][yNbB_k], and nbH is set equal to CbHeight[xNbB_k][yNbB_k].
 - If PredFlagLX[xNbB_k][yNbB_k] is equal to 1 and DiffPicOrderCnt(RefPicListX[RefIdxLX[xNbB_k][yNbB_k]], RefPicListX[refIdxLX]) is equal to 0, the following applies:
 - The variable availableFlagB is set equal to TRUE
 - The derivation process for luma affine control point motion vectors from a neighbouring block as specified in clause 8.3.3.5 is invoked with the luma coding block location (xCb, yCb), the luma coding block width and height

(cbWidth, cbHeight), the neighbouring luma coding block location (xNb, yNb), the neighbouring sub-block center location (xNbB_k, yNbB_k), the neighbouring luma coding block width and height (nbW, nbH), and the number of control point motion vectors numCpMv as input, the control point motion vector predictor candidates cpMvpLX[cpIdx] with cpIdx = 0 .. numCpMv - 1 as output.

- The rounding process for motion vectors as specified in clause 8.3.2.12 is invoked with mvX set equal to cpMvpLX[cpIdx], rightShift set equal to 2, and leftShift set equal to 2 as inputs and the rounded cpMvpLX[cpIdx] with cpIdx = 0 .. numCpMv - 1 as output.

- The following assignments are made:

$$\text{cpMvpListLX}[\text{numCpMvpCandLX}][0] = \text{cpMvpLX}[0] \quad (8-453)$$

$$\text{cpMvpListLX}[\text{numCpMvpCandLX}][1] = \text{cpMvpLX}[1] \quad (8-454)$$

$$\text{cpMvpListLX}[\text{numCpMvpCandLX}][2] = \text{cpMvpLX}[2] \quad (8-455)$$

$$\text{numCpMvpCandLX} = \text{numCpMvpCandLX} + 1 \quad (8-456)$$

- Otherwise if PredFlagLY[xNbB_k][yNbB_k] (with Y = !X) is equal to 1 and DiffPicOrderCnt(RefPicListY[RefIdxLY[xNbB_k][yNbB_k]], RefPicListX[refIdxLX]) is equal to 0, the following applies:

- The variable availableFlagB is set equal to TRUE

- The derivation process for luma affine control point motion vectors from a neighbouring block as specified in clause 8.3.3.5 is invoked with the luma coding block location (xCb, yCb), the luma coding block width and height (cbWidth, cbHeight), the neighbouring luma coding block location (xNb, yNb), the neighbouring luma coding block width and height (nbW, nbH), and the number of control point motion vectors numCpMv as input, the control point motion vector predictor candidates cpMvpLY[cpIdx] with cpIdx = 0 .. numCpMv - 1 as output.

- The rounding process for motion vectors as specified in clause 8.3.2.12 is invoked with mvX set equal to cpMvpLY[cpIdx], rightShift set equal to 2, and leftShift set equal to 2 as inputs and the rounded cpMvpLY[cpIdx] with cpIdx = 0 .. numCpMv – 1 as output.
- The following assignments are made:

$$\text{cpMvpListLX}[\text{numCpMvpCandLX}][0] = \text{cpMvpLY}[0] \quad (8-457)$$

$$\text{cpMvpListLX}[\text{numCpMvpCandLX}][1] = \text{cpMvpLY}[1] \quad (8-458)$$

$$\text{cpMvpListLX}[\text{numCpMvpCandLX}][2] = \text{cpMvpLY}[2] \quad (8-459)$$

$$\text{numCpMvpCandLX} = \text{numCpMvpCandLX} + 1 \quad (8-460)$$

[00161] FIG. 24 is a block diagram of a video processing apparatus 2600. The apparatus 2600 may be used to implement one or more of the methods described herein. The apparatus 2600 may be embodied in a smartphone, tablet, computer, Internet of Things (IoT) receiver, and so on. The apparatus 2600 may include one or more processors 2602, one or more memories 2604 and video processing hardware 2606. The processor(s) 2602 may be configured to implement one or more methods described in the present document. The memory (memories) 2604 may be used for storing data and code used for implementing the methods and techniques described herein. The video processing hardware 2606 may be used to implement, in hardware circuitry, some techniques described in the present document.

[00162] FIG. 25 is a flowchart for an example method 2500 of video processing. The method 2500 includes associating (2502) , with a current video block, a first group of motion vectors (MVs) for determining inherited motion information of other video blocks, a second group of MVs for deriving MVs of sub-blocks of the current video block and a third group of MVs that is included in a bitstream representation of the current video block; and performing (2504) a conversion between the current video block and the bitstream representation using the first group of MVs, the second group of MVs or the third group of MVs.

[00163] It will be appreciated that several techniques have been disclosed that will benefit video encoder and decoder embodiments incorporated within video processing devices such as smartphones, laptops, desktops, and similar devices by allowing a reduction in the amount of

memory used for storing CPMVs during affine coding based video coding and decoding. Various embodiments and techniques may be described using the following clause-based description.

[00164] 1. A method of video processing, comprising:

[00165] associating, with a current video block, a first group of motion vectors (MVs) for determining inherited motion information of other video blocks, a second group of MVs for deriving MVs of sub-blocks of the current video block and a third group of MVs that is included in a bitstream representation of the current video block; and

[00166] performing a conversion between the current video block and the bitstream representation using the first group of MVs, the second group of MVs or the third group of MVs.

[00167] 2. The method of clause 1, wherein the first group of MVs comprises control point MVs (CPMV), MVF0, MVF1 and MVF2, at represented points (xF0, yF0), (xF1, yF1) and (xF2, yF2), respectively.

[00168] 3. The method of any of clauses 1-2, wherein the second group of MVs comprises control point MVs (CPMV), denoted MVS0, MVS1 and MVS2, at represented points (xS0, yS0), (xS1, yS1) and (xS2, yS2), respectively.

[00169] 4. The method of any of clauses 1-3, wherein the third group of MVs comprises control point MVs (CPMV), denoted as MVT0, MVT1 and MVT2, at represented points (xT0, yT0), (xT1, yT1) and (xT2, yT2), respectively.

[00170] 5. The method of any of clauses 1-4, wherein the second group of MVs are same as the third group of MVs.

[00171] 6. The method of any of clauses 1-5, wherein the first group of MVs are derived from the second group of MVs or the third group of MVs.

[00172] 7. The method of any of clauses 1-2, wherein a relative offset between representative points of two control point MVs in the first group is independent of a width or a height of the current video block.

[00173] 8. The method of clause 2, wherein:

[00174] $yF1 = yF0$, $xF1 = xF0 + PW$, or

[00175] $xF1 = xF0$, $yF1 = yF0 + PH$, or

[00176] $yF2 = yF0$, $xF2 = xF0 + PW$, or

[00177] $xF2 = xF0$, $yF2 = yF0 + PH$, or

[00178] $yF2 = yF1$, $xF2 = xF1 + PW$, or

[00179] $x_{F2} = x_{F1}$, $y_{F2} = y_{F1} + PH$, or

[00180] Where PW and PH are integers.

[00181] 8. The method of any of clauses 4-7, wherein

[00182] $MVF0 = MVS0$, $(x_{F0}, y_{F0}) = (x_{S0}, y_{S0})$, or $MVF0 = MVT0$, $(x_{F0}, y_{F0}) = (x_{T0}, y_{T0})$;
or

[00183] $MVF0 = MVS1$, $(x_{F0}, y_{F0}) = (x_{S1}, y_{S1})$, or $MVF0 = MVT1$, $(x_{F0}, y_{F0}) = (x_{T1}, y_{T1})$;
or

[00184] $MVF0 = MVS2$, $(x_{F0}, y_{F0}) = (x_{S2}, y_{S2})$, or $MVF0 = MVT2$, $(x_{F0}, y_{F0}) = (x_{T2}, y_{T2})$.

[00185] 9. The method of any of clauses 2-8, further including storing a difference value $D1 = MVF1 - MVF0$.

[00186] 10. The method of any of clauses 2-9, further including storing a difference value $D2 = MVF2 - MVF0$.

[00187] 11. The method of any of clauses 9-10, wherein the storing includes storing a bit shifted version of motion vector values.

[00188] 12. The method of clause 11, wherein the storing further includes clipping motion vector values prior to storing.

[00189] Additional examples and embodiments of clauses 1-12 are described in Section 4, e.g., Items 1-3.

[00190] 13. A method of video processing, comprising: performing a conversion between a current block and a bitstream representation of the current block using affine inherited motion vectors (MVs) for the current block, wherein the affine inherited MVs are derived from (1) MVs stored for an adjacent neighboring basic block, denoted as B_{adj} , or (2) an affine history list.

[00191] 14. The method of clause 13, wherein the MVs stored for B_{adj} include: L (left), A (above), LB (left below), AR (above right) and AL (above left), and wherein B_{adj} is a 4x4 size block.

[00192] 15. The method of any of clauses 13-14, wherein a MV at a position (x, y) in the current block is computed using a motion vector MV_a of B_{adj} at a position (x_0, y_0) , wherein (x_0, y_0) is one of: (a) a position inside B_{adj} , or (b) a position on outside or on boundary of B_{adj} .

[00193] 16. The method of clause 15, wherein the position (x, y) is (1) in a sub-block of the current block, or (2) a corner of the current block.

[00194] 17. The method of any of clauses 13-16, wherein the current block uses a 4-parameter affine model.

[00195] 18. The method of any of clauses 13-16, wherein the current block uses a 6-parameter affine model.

[00196] Additional examples and embodiments of clauses 13-18 are described in Section 4, e.g., Items 3-6.

[00197] 19. A method of video processing, comprising: performing a conversion between a current block and a bitstream representation of the current block using affine inherited motion vectors (MVs) for the current block, wherein the affine inherited MVs are derived from a first MV stored in a first basic block adjacently neighboring the current block and a second MV stored in a second basic block that is offset from the first building block by an offset.

[00198] 20. The method of clause 19, wherein the affine inherited MVs are derived using a linear weighting of x- and y- differences between motion vectors weighted using coefficients a and b, wherein a and b are derived from the offset.

[00199] 21. The method of any of clauses 19-20 wherein a value of the offset is a function of a position of the first basic block.

[00200] 22. The method of any of clauses 19-21, wherein the second basic block is selected from M candidate basic blocks, where M is an integer, by checking the M candidate basic blocks in an order.

[00201] 23. The method of any of clauses 19-22, wherein the first building block and the second basic building block are inter-coded and refer to a same reference picture.

[00202] 24. The method of any of clauses 1-23, wherein affine parameters for the current block are bit-shifted prior to storing.

[00203] 25. The method of clause 24, wherein an amount bit-shifting (1) changes with motion precision used during the conversion, or (2) is different for different affine parameters.

[00204] 26. The method of clause 24, wherein the affine parameters are clipped prior to the storing.

[00205] 27. The method of any of clauses 1-12, wherein the MVs in the first groups of MVs are stored and used in a history-based motion vector predictor table for a history-based candidate affine model of the current block.

[00206] 28. The method of clauses 9-10 wherein D1 and D2 are stored and used in a history-based motion vector predictor table for a history-based candidate affine model of the current block.

[00207] 29. The method of clause 19, wherein the first basic block adjacently neighboring the current block is one of A (above), AR (above right) or AL (above left), and wherein a top-left sample of the first basic block is denoted x_{BB} .

[00208] 30. The method of clause 29, wherein ($x_{BB} \% 8 = 0$), wherein a block right of the first basic block is affine coded and has a reference index identical to a reference index of the first basic block for a given reference list, and wherein the second basic block is the block right of the first basic block.

[00209] 31. The method of clause 29, wherein ($x_{BB} \% 8 = 0$), wherein a block right of the first basic block is not affine coded or has a reference index that is different from a reference index of the first basic block for a given reference list, and wherein the second basic block is a block left of the first basic block.

[00210] 32. The method of clause 29, wherein ($x_{BB} \% 8 \neq 0$), wherein a block left of the first basic block is affine coded and has a reference index identical to a reference index of the first basic block for a given reference list, and wherein the second basic block is the block left of the first basic block.

[00211] 33. The method of clause 29, wherein ($x_{BB} \% 8 \neq 0$), wherein a block left of the first basic block is not affine coded or has a reference index different from a reference index of the first basic block for a given reference list, and wherein the second basic block is a block right of the first basic block.

[00212] 34. The method of clause 19, wherein the first basic block adjacently neighboring the current block is located in a row or a column adjacent to the current block.

[00213] 35. The method of clause 19, wherein at least one of the affine inherited MVs is aligned to the first MV or the second MV.

[00214] 36. The method of clause 19, wherein the second basic block is selected depending on a position of the first basic block, wherein the position of the first basic block is above left, or above right, or at left boundary, or at right boundary of the current block and depending on the position, a corresponding offset is selected based on the position.

[00215] 37. The method of clause 19, wherein the second basic block is selected from multiple candidate blocks using a technique that depends on a position of the first basic block or a position of the current block.

[00216] 38. The method of any of clauses 19-37, wherein a decision to find the second block from multiple candidates or from a predefined offset is made based on a position of the first block or a position of the current block.

[00217] 39. The method of clause 38, wherein, in case that the current block inherits an affine model from a basic block, then considering the basic block as a first neighboring basic block and determining a second adjacent basic block based on a rule based on a position of the basic block.

[00218] 40. The method of any of clauses 19-39, wherein in case that a basic block P is selected as the second block and a basic block Q is selected as first block, then the method excludes using the basic block P as the first block and the basic block Q as the second block during the conversion for another video block.

[00219] 41. The method of clause 41, wherein the another video block is in a same coding tree unit row or a same slice as the current video block.

[00220] Additional features and examples of the techniques described in Clauses 36-41 are provide in Section 4, items 7(g), 7(h), 7(m), 7(n), and 7(o).

[00221] 42. The method of any of clauses 1-41, wherein the conversion includes generation of the bitstream representation from pixel values of the current block.

[00222] 43. The method of any of clauses 1-41, wherein the conversion includes generation of pixel values of the current block from the bitstream representation.

[00223] 44. A video encoder apparatus comprising a processor configured to implement a method recited in any one or more of clauses 1-43.

[00224] 45. A video decoder apparatus comprising a processor configured to implement a method recited in any one or more of clauses 1-43.

[00225] 46. A computer program product stored on a non-transitory computer readable media, the computer program product including program code for carrying out the method in any of clauses 1 to 43.

[00226] FIG. 29 is a flowchart for a method 2900 of processing video. The method 2900 includes, associating (2902) a first group of control point motion vectors (CPMVs) for determining inherited motion information of blocks coded after a first block, with a second group

of CPMVs for deriving MVs of sub-blocks of the first block or a third group of CPMVs that is signaled for the first block, wherein the first group of CPMVs is not identical with the second group of CPMVs or the third group of CPMVs; determining (2904) inherited motion information for a second block, which is coded after the first block, based on the first group of CPMVs, and performing (2906) a conversation between the second block and a bitstream representation of the second block by using the inherited motion information.

[00227] In some examples, the first group of CPMVs are derived from the second group of CPMVs or the third group of CPMVs.

[00228] In some examples, the method further comprises: storing the first group of CPMVs after the conversion of the first block.

[00229] In some examples, the second group of CPMVs are same as the third group of CPMVs.

[00230] In some examples, multiple representative points' coordinates of the first group of CPMVs, multiple representative points' coordinates of the second group of CPMVs and/or multiple representative points' coordinates of the third group of CPMVs are defined as coordinates relative to one block or sub-block which is used in an affine motion compensation process.

[00231] In some examples, a relative offset between representative points of two CPMVs in the first group of CPMVs is independent of a width or a height of the first block.

[00232] In some examples, representative points of the first group of CPMVs are inside of the first block or outside of the first block.

[00233] In some examples, $yF1 = yF0$, $xF1 = xF0 + PW$, or $xF1 = xF0$, $yF1 = yF0 + PH$, or $yF2 = yF0$, $xF2 = xF0 + PW$, or $xF2 = xF0$, $yF2 = yF0 + PH$, or $yF2 = yF1$, $xF2 = xF1 + PW$, or $xF2 = xF1$, $yF2 = yF1 + PH$, where $(xF0, yF0)$, $(xF1, yF1)$, $(xF2, yF2)$ are coordinates of multiple representative points of the first group of CPMVs, and PW and PH are integers.

[00234] In some examples, $PW = 2M$, or $PW = -2M$, or $PH = 2M$, or $PH = -2M$, M is an integer in a range of 2 to 7.

[00235] In some examples, PW and PH are not stored.

[00236] In some examples, PW and PH are fixed.

[00237] In some examples, PW and PH are signaled in at least one of Sequence Parameter Set (SPS), Video Parameter Set (VPS), Picture Parameter Set (PPS), slice header, tile group header, tile, or CTU.

[00238] In some examples, PW and PH are different in different standard profiles or levels or tiers.

[00239] In some examples, PW and PH depend on maximum coding unit (CU) size or/and minimum CU size of slice or picture.

[00240] In some examples, in the first group of CPMVs, $MVF0 = MVS0$, $(xF0, yF0) = (xS0, yS0)$, or $MVF0 = MVT0$, $(xF0, yF0) = (xT0, yT0)$; or $MVF0 = MVS1$, $(xF0, yF0) = (xS1, yS1)$, or $MVF0 = MVT1$, $(xF0, yF0) = (xT1, yT1)$; or $MVF0 = MVS2$, $(xF0, yF0) = (xS2, yS2)$, or $MVF0 = MVT2$, $(xF0, yF0) = (xT2, yT2)$; wherein motion vectors $MVF0$, $MVF1$, $MVF2$ are multiple CPMVs at multiple representative points' coordinates $(xF0, yF0)$, $(xF1, yF1)$, $(xF2, yF2)$ of the first group of CPMVs, motion vectors $MVS0$, $MVS1$, $MVS2$ are multiple CPMVs at multiple representative points' coordinates $(xS0, yS0)$, $(xS1, yS1)$, $(xS2, yS2)$ of the second group of CPMVs, and motion vectors $MVT0$, $MVT1$, $MVT2$ are multiple CPMVs at multiple representative points' coordinates $(xT0, yT0)$, $(xT1, yT1)$, $(xT2, yT2)$ of the third group of CPMVs.

[00241] In some examples, motion vectors $MVF0$, $MVF1$ and $MVF2$ in the first group of CPMVs are derived from motion vectors $MVS0$ and $MVS1$ in the second group of CPMVs by using a 4-parameter affine model with coordinates $(xF0, yF0)$, $(xF1, yF1)$ and $(xF2, yF2)$ as the input coordinates of the affine model.

[00242] In some examples, motion vectors $MVF0$, $MVF1$ and $MVF2$ in the first group of CPMVs are derived from motion vectors $MVS0$, $MVS1$ and $MVS2$ in the second group of CPMVs by using a 6-parameter affine model with $(xF0, yF0)$, $(xF1, yF1)$ and $(xF2, yF2)$ as the input coordinates of the affine model.

[00243] In some examples, motion vectors $MVF0$, $MVF1$ and $MVF2$ in the first group of CPMVs are derived from motion vectors $MVT0$ and $MVT1$ in the third group of CPMVs by using a 4-parameter affine model with $(xF0, yF0)$, $(xF1, yF1)$ and $(xF2, yF2)$ as the input coordinates of the affine model.

[00244] In some examples, motion vectors $MVF0$, $MVF1$ and $MVF2$ in the first group of CPMVs are derived from motion vectors $MVT0$, $MVT1$ and $MVT2$ in the third group of CPMVs by using a 6-parameter affine model with $(xF0, yF0)$, $(xF1, yF1)$ and $(xF2, yF2)$ as the input coordinates of the affine model.

[00245] In some examples, motion vectors MVF2 in the first group of CPMVs comprising motion vectors MVF0, MVF1 and MVF2 is only calculated if the first block is coded with a 6-parameter affine model, or motion vectors MVF2 in the first group of CPMVs comprising motion vectors MVF0, MVF1 and MVF2 is calculated no matter the first block is coded with a 4-parameter affine model or a 6-parameter affine model.

[00246] In some examples, the method further comprises: storing one or more differences (D1, D2) between the CPMVs in the first group of CPMVs.

[00247] In some examples, the first group of CPMVs comprises motion vectors MVF0, MVF1 and MVF2, $D1 = MVF1 - MVF0$ is stored, or $D2 = MVF2 - MVF0$ is stored, or both D1 and D2 are stored.

[00248] In some examples, D2 is stored only when the first block is coded with a 6-parameter affine model.

[00249] In some examples, D2 is stored when the first block is coded with the 6-parameter affine model or the 6-parameter affine model.

[00250] In some examples, the method further comprises: storing the first group of CPMVs and one or more differences (D1, D2) between the CPMVs in the first group of CPMVs together.

[00251] In some examples, the multiple CPMVs in the first group of CPMVs and/or one or more differences between the CPMVs in the first group of CPMVs are shifted with a shift function, and the shifted CPMVs and/or differences are stored.

[00252] In some examples, the shift function $SatShift(x, n)$ is defined as: wherein n is an integer, and $offset0$ and/or $offset1$ are set to $(1 \ll n) \gg 1$ or $(1 \ll (n-1))$, or $((1 \ll n) \gg 1) - 1$, or $offset0$ and/or $offset1$ are set to 0.

[00253] In some examples, the shift function $SatShift(x, n)$ is defined as: $Shift(x, n) = (x + offset0) \gg n$, wherein n is an integer, and $offset0$ is set to $(1 \ll n) \gg 1$ or $(1 \ll (n-1))$, or $((1 \ll n) \gg 1) - 1$, or $offset0$ is set to 0.

[00254] In some examples, n is 2 or 4, or n depends on the motion precision.

[00255] In some examples, n in case that the CPMV in the first group of CPMVs is different from n in case that the difference between CPMVs in the first group of CPMVs is stored.

[00256] In some examples, stored CPMV is left shift first before it is used in the affine inheritance of the blocks coded after the first block.

[00257] In some examples, the multiple CPMVs in the first group of CPMVs and/or one or more differences between the CPMVs in the first group of CPMVs are clipped with a clip function, and the clipped CPMVs and/or the differences are stored.

[00258] In some examples, the clip function $\text{Clip3}(\min, \max, x)$ is defined as:

$$\text{Clip3}(\text{Min}, \text{Max}, x) = \begin{cases} \text{Min} & \text{if } x < \text{Min} \\ \text{Max} & \text{if } x > \text{Max} \\ x & \text{Otherwise} \end{cases}$$

wherein Min is a lower threshold of the clip function, Max is a higher threshold of the clip function.

[00259] In some examples, when the CPMV or the difference is stored with K bits, $\text{Min} = -2^{K-1}$ and $\text{Max} = 2^{K-1} - 1$, K is an integer.

[00260] In some examples, K is different depending on whether the CPMV or the difference is to be stored.

[00261] In some examples, the multiple CPMVs in the first group of CPMVs and/or one or more differences between the CPMVs in the first group of CPMVs are processed with the shift function and the clip function sequentially, and the processed CPMVs and/or the differences are stored.

[00262] In some examples, the multiple CPMVs in the first group of CPMVs are stored into an affine (History Motion Vector Prediction) HMVP buffer or table or list to represent one history-based candidate affine model.

[00263] In some examples, the one or more differences between the CPMVs in the first group of CPMVs are stored into an affine (History Motion Vector Prediction) HMVP buffer or table or list to represent one history-based candidate affine model.

[00264] In some examples, one or more CPMVs or one or more differences stored in the HMVP buffer or table or list are shifted with the shift function and/or clipped with the clip function.

[00265] FIG. 30 is a flowchart for a method 3000 of processing video. The method 3000 includes, deriving (3002), for a conversion between a current block of video and a bitstream representation of the current block, affine inherited motion vectors (MVs) for the first block based on stored motion vectors (MVs); performing (3004) the conversion by using the affine inherited MVs.

[00266] In some examples, the MVs are stored in an adjacent neighbouring basic block.

[00267] In some examples, the MVs are stored in affine (History Motion Vector Prediction) HMVP buffer or table or list.

[00268] In some examples, the MVs stored in the adjacent neighbouring basic block at least include: MV stored in a left adjacent neighbouring basic block (L), MV stored in an above adjacent neighbouring basic block (A), MV stored in a left-bottom adjacent neighbouring basic block (LB), MV stored in an above-right adjacent neighbouring basic block (AR) and MV stored in an above left adjacent neighbouring basic block (AL).

[00269] In some examples, the adjacent neighbouring basic block is a 4x4 block

[00270] In some examples, the affine inherited MV at a position (x, y) in the first block is derived by using a first MV of the adjacent neighbouring basic block ($MV_a = (mv_a^h, mv_a^v)$) at a representative point (x_0, y_0) based on an affine model with $(x-x_0, y-y_0)$ as the input coordinates of the affine model.

[00271] In some examples, the representative point (x_0, y_0) is any position inside the basic block.

[00272] In some examples, the representative point (x_0, y_0) is any position outside or at the boundary of the basic block.

[00273] In some examples, the coordinate of the representative point (x_0, y_0) is determined based on the coordinate (xTL, yTL) of a top-left corner sample in the adjacent neighbouring basic block and additional information including two variables (i, j).

[00274] In some examples, a first variable (i) of the two variables depends on width of the basic block, and a second variable (j) of the two variables depends on height of the basic block.

[00275] In some examples, the variables (i, j) depends on the position of the neighbouring basic block.

[00276] In some examples, the coordinate of the representative point (x_0, y_0) is determined based on the coordinate of top-left sample of the first block (xPos00, yPos00), the coordinate of top-right sample of the first block (xPos10, yPos00), and the coordinate of top-right sample of the first block (xPos00, yPos01).

[00277] In some examples, the coordinate of the representative point (x_0, y_0) for the left adjacent neighbouring basic block L is (xPos00-2, yPos01-1); the coordinate of the representative point (x_0, y_0) for the left-bottom adjacent neighbouring basic block (LB) is (xPos00-2, yPos01+3); the coordinate of the representative point (x_0, y_0) for the above adjacent neighbouring basic block (A)

is (xPos10-1, yPos00-2); the coordinate of the representative point (x_0, y_0) for the above-right adjacent neighbouring basic block (AR) is (xPos10+3, yPos00-2); the coordinate of the representative point (x_0, y_0) for the above left adjacent neighbouring basic block (AL) is (xPos00-2, yPos00-2).

[00278] In some examples, the additional information depends on the position of the adjacent neighbouring basic block or the adjacent neighbouring basic block, or the additional information is signaled in at least one of Sequence Parameter Set (SPS), Video Parameter Set (VPS), Picture Parameter Set (PPS), slice header, tile group header, tile, coding tree unit (CTU), CU.

[00279] In some examples, the additional information is different in different standard profiles or levels or tiers.

[00280] In some examples, the affine inherited MV at a position (x, y) in a sub-block of the first block is derived by using a first MV of the adjacent neighbouring basic block (MVa) at a representative point (x_0, y_0) based on an affine model with (x- x_0 , y- y_0) as the input coordinates of the affine model.

[00281] In some examples, the affine inherited MV at a position (x, y) in a sub-block is used to perform motion compensation for the sub-block.

[00282] In some examples, the affine inherited MV at a position (x, y), which is a corner of the first block, is derived as an inherited control point motion vector (CPMV) of the first block by using a first MV of the adjacent neighbouring basic block (MVa) at a representative point (x_0, y_0) based on an affine model with (x- x_0 , y- y_0) as the input coordinates of the affine model.

[00283] In some examples, the inherited CPMVs are used to predict signaled CPMVs of the first block which is affine inter-coded.

[00284] In some examples, the inherited CPMVs are directly used as CPMVs of the first block which is affine merge-coded.

[00285] In some examples, when the first block uses a 4-paramter affine model, the affine model for deriving the affine inherited MV at a position (x, y) in the first block is:

$$\begin{cases} mv^h(x, y) = a(x - x_0) - b(y - y_0) + mv_a^h \\ mv^v(x, y) = b(x - x_0) + a(y - y_0) + mv_a^v \end{cases},$$

wherein a and b are variables of the affine model.

[00286] In some examples, when the first block uses a 6-paramter affine model, the affine model for deriving the affine inherited MV at a position (x, y) in the first block is:

$$\begin{cases} mv^h(x, y) = a(x - x_0) + c(y - y_0) + mv_a^h \\ mv^v(x, y) = b(x - x_0) + d(y - y_0) + mv_a^v \end{cases},$$

wherein a, b, c and d are variables of the affine model.

[00287] In some examples, the variable a, b, or variable a, b, c, d are calculated as:

$$\begin{cases} a = \frac{(mv_{t1}^h - mv_{t0}^h)}{w_t} \\ b = \frac{(mv_{t1}^v - mv_{t0}^v)}{w_t} \\ c = \frac{(mv_{t2}^h - mv_{t0}^h)}{h_t} \\ d = \frac{(mv_{t2}^v - mv_{t0}^v)}{h_t} \end{cases},$$

wherein $mv_{t0} = (mv_{t0}^h, mv_{t0}^v)$, $mv_{t1} = (mv_{t1}^h, mv_{t1}^v)$, $mv_{t2} = (mv_{t2}^h, mv_{t2}^v)$ are CPMVs at three representative points, respectively, in a first group CPMVs for a second block covering the adjacent neighbouring basic block, and w_t and h_t depends on relative offsets between the representative points of the second block, wherein the first group CPMVs is used for determining inherited motion information of blocks coded after the second block.

[00288] In some examples, the variable a, b, or variable a, b, c, d are calculated as:

$$\begin{cases} a = \frac{(mv_{t1}^h - mv_{t0}^h)}{w_t} \\ b = \frac{(mv_{t1}^v - mv_{t0}^v)}{w_t} \\ c = \frac{(mv_{t2}^h - mv_{t0}^h)}{h_t} \\ d = \frac{(mv_{t2}^v - mv_{t0}^v)}{h_t} \end{cases},$$

wherein $mv_{t0} = (mv_{t0}^h, mv_{t0}^v)$, $mv_{t1} = (mv_{t1}^h, mv_{t1}^v)$, $mv_{t2} = (mv_{t2}^h, mv_{t2}^v)$ are CPMVs at three representative points, respectively, in a second group CPMVs or a third group of CPMVs for a second block covering the adjacent neighbouring basic block, and w_t and h_t are the width and height of the second block, wherein the second group CPMVs are used to derive MVs for each sub-block of the second block, and the third group CPMVs are signaled from encoder to decoder.

[00289] In some examples, the variable a, b, or variable a, b, c, d are calculated as:

$$\left\{ \begin{array}{l} a = \frac{(mv_{t1}^h - mv_{t0}^h)}{w_t} \\ b = \frac{(mv_{t1}^v - mv_{t0}^v)}{w_t} \\ c = \frac{(mv_{t2}^h - mv_{t0}^h)}{h_t} \\ d = \frac{(mv_{t2}^v - mv_{t0}^v)}{h_t} \end{array} \right.,$$

wherein $mv_{t0} = (mv_{t0}^h, mv_{t0}^v)$, $mv_{t1} = (mv_{t1}^h, mv_{t1}^v)$, $mv_{t2} = (mv_{t2}^h, mv_{t2}^v)$ are CPMVs at three representative points, respectively, of a second block covering the adjacent neighbouring basic block, and w_t and h_t are the width and height of the second block, wherein $mv_{t1}^h - mv_{t0}^h$, $mv_{t1}^v - mv_{t0}^v$, $mv_{t2}^h - mv_{t0}^h$, $mv_{t2}^v - mv_{t0}^v$ are fetched from a storage for storing difference between CPMVs of the blocks directly.

[00290] In some examples, the conversion generates the first/second block of video from the bitstream representation.

[00291] In some examples, the conversion generates the bitstream representation from the first/second block of video.

[00292] FIG. 31 is a flowchart for a method 3100 of processing video. The method 3100 includes, deriving (3102), for a conversion between a current block of video and a bitstream representation of the current block, affine inherited motion vectors (MVs) for the current block based on a first stored motion vector (MV) and a second stored MV different from the first stored MV, wherein the first stored MV is stored in a first basic block neighbouring to the current block, and the second stored MV is stored in a second basic block with an offset to the first basic block; and; performing (3104) the conversion by using the affine inherited MVs for the current block.

[00293] In some examples, the first basic block neighbouring to the current block includes at least one of: a left neighbouring basic block (L), an above neighbouring basic block (A), a left-bottom neighbouring basic block (LB), an above-right neighbouring basic block (AR, AR') and an above left neighbouring basic block (AL, AL', AL1, AL'').

[00294] In some examples, the affine inherited MV at a position (x, y) in the current block $((mv^h(x, y), mv^v(x, y)))$ is derived by using the first stored MV ($MV_a = (mv_a^h, mv_a^v)$) stored in the first basic block at a point (x_0, y_0) or the first stored MV ($MV_a = (mv_a^h, mv_a^v)$) associated with a

sub-block of the first basic block at a point (x_0, y_0) based on an affine model with parameters a and b , the motion vector $(mv^h(x, y), mv^v(x, y))$ is derived by:

$$\begin{cases} mv^h(x, y) = a(x - x_0) - b(y - y_0) + mv_a^h \\ mv^v(x, y) = b(x - x_0) + a(y - y_0) + mv_a^v \end{cases}$$

[00295] In some examples, the parameters a and b are calculated as:

$$\begin{cases} a = \frac{(mv_{t1}^h - mv_{t0}^h)}{w_t} \\ b = \frac{(mv_{t1}^v - mv_{t0}^v)}{w_t} \end{cases},$$

wherein $mv_{t0} = (mv_{t0}^h, mv_{t0}^v)$ and $mv_{t1} = (mv_{t1}^h, mv_{t1}^v)$ are set equal to the MV stored in the first basic block and the MV stored in the second basic block, respectively, and w_t is horizontal offset between the first basic block and the second basic block.

[00296] In some examples, the parameters a and b are calculated as:

$$\begin{cases} a = \frac{(mv_{t1}^v - mv_{t0}^v)}{h_t} \\ b = -\frac{(mv_{t1}^h - mv_{t0}^h)}{h_t} \end{cases},$$

wherein $mv_{t0} = (mv_{t0}^h, mv_{t0}^v)$ and $mv_{t1} = (mv_{t1}^h, mv_{t1}^v)$ are set equal to the MV stored in the first basic block and the MV stored in the second basic block, respectively, and h_t is vertical offset between the first basic block and the second basic block.

[00297] In some examples, $w_t = 2^N$, and $h_t = 2^M$, wherein N and M are integers.

[00298] In some examples, when coordinates of the top-left sample of the first basic block and the second basic block are (x_{LT0}, y_{LT0}) and (x_{LT1}, y_{LT1}) , respectively, the horizontal offset between the first basic block and the second basic block is defined as $x_{LT1} - x_{LT0}$, and/or the vertical offset between the first basic block and the second basic block is defined as $y_{LT1} - y_{LT0}$.

[00299] In some examples, when the first basic block is above the current block, the vertical offset is 0.

[00300] In some examples, when the first basic block is left to the current block, the horizontal offset is 0.

[00301] In some examples, the second basic block is selected depending on the position of the first basic block.

[00302] In some examples, coordinates of top-left sample of the first basic block and the second basic block are (x_{LT0}, y_{LT0}) and (x_{LT1}, y_{LT1}) , respectively, and

[00303] wherein when $x_{LT0} \% M$ is not equal to 0, M is an integer, $y_{LT1} = y_{LT0}$ and $x_{LT1} = x_{LT0} - offset$, or $x_{LT1} = x_{LT0}$ and $y_{LT1} = y_{LT0} - offset$, where *offset* is a positive number.

[00304] In some examples, coordinates of top-left sample of the first basic block and the second basic block are (x_{LT0}, y_{LT0}) and (x_{LT1}, y_{LT1}) , respectively, and

[00305] wherein when $x_{LT0} \% M$ is equal to 0, M is an integer, $y_{LT1} = y_{LT0}$ and $x_{LT1} = x_{LT0} + offset$, or $x_{LT1} = x_{LT0}$ and $y_{LT1} = y_{LT0} + offset$, where *offset* is a positive number.

[00306] In some examples, coordinates of top-left sample of the first basic block and the second basic block are (x_{LT0}, y_{LT0}) and (x_{LT1}, y_{LT1}) , respectively, and when the first block is above-left to the current block, or when the first block is above-left to the current block and left boundary of the current block is also the left boundary of a CTU, or when $x_{LT0} - offset < x_{LT_AL}$, where x_{LT_AL} is the top-left coordinate of a neighbouring basic block above-left to the current block, or when $x_{LT0} - offset < x_{LT_AL}$, where x_{LT_AL} is the top-left coordinate of a neighbouring basic block above-left to the current block and left boundary of the current block is also the left boundary of a CTU,

[00307] $y_{LT1} = y_{LT0}$ and $x_{LT1} = x_{LT0} + offset$, where *offset* is a positive number.

[00308] In some examples, coordinates of top-left sample of the first basic block and the second basic block are (x_{LT0}, y_{LT0}) and (x_{LT1}, y_{LT1}) , respectively, and when the first block is above-right to the current block, or when the first block is above-right to the current block and right boundary of the current block is also the right boundary of a CTU, or when $x_{LT0} + offset > x_{RT_AR}$, where x_{RT_AR} is the top-right coordinate of a neighbouring basic block above-right to the current block, or when $x_{LT0} + offset > x_{RT_AR}$, where x_{RT_AR} is the top-right coordinate of a neighbouring basic block above-right to the current block and right boundary of the current block is also the right boundary of a CTU, $y_{LT1} = y_{LT0}$ and $x_{LT1} = x_{LT0} - offset$, where *offset* is a positive number.

[00309] In some examples, the second basic block is selected from M candidate basic blocks, M is an integer.

[00310] In some examples, the second basic block is selected by checking the M candidate basic blocks in order so as to determine one of the M candidate basic blocks, which is inter-coded and has a MV referring to the same reference picture as the MV of the first basic block referring to, as the second basic block.

[00311] In some examples, coordinates of top-left sample of the first basic block is (x_{LT0}, y_{LT0}) and coordinates of top-left positions of the M candidate basic block are $(x_{C0}, y_{C0}), (x_{C1}, y_{C1}), \dots, (x_{CM-1}, y_{CM-1})$, respectively, and when M is 2, $y_{C0} = y_{LT0}$, $x_{C0} = x_{LT0} - offset$, and $y_{C1} = y_{LT0}$, $x_{C1} = x_{LT0} + offset$ or, $x_{C0} = x_{LT0}$, $y_{C0} = y_{LT0} - offset$, and $x_{C1} = x_{LT0}$, $y_{C1} = y_{LT0} + offset$, or $y_{C0} = y_{LT0}$, $x_{C0} = x_{LT0} - offset$, $y_{C1} = y_{LT0}$, $x_{C1} = x_{LT0} + offset$, or $x_{C0} = x_{LT0}$, $y_{C0} = y_{LT0} - offset$, $x_{C1} = x_{LT0}$, $y_{C1} = y_{LT0} + offset$, where *offset* is a positive number.

[00312] In some examples, whether to and/or how to select the second basic block from the M candidate basic blocks depend on the position of the first basic block and/or the position of the current block.

[00313] In some examples, coordinates of top-left sample of the first basic block is (x_{LT0}, y_{LT0}) and coordinates of top-left positions of the M candidate basic block are $(x_{C0}, y_{C0}), (x_{C1}, y_{C1}), \dots, (x_{CM-1}, y_{CM-1})$, respectively, and when M is 1, $y_{C0} = y_{LT0}$, $x_{C0} = x_{LT0} + offset$, if the first basic block is above-left to the current block, or $y_{C0} = y_{LT0}$, $x_{C0} = x_{LT0} - offset$, if the first block is above-right to the current block, or $y_{C0} = y_{LT0}$, $x_{C0} = x_{LT0} + offset$, if the first block is above-left to the current block and left boundary of the current block is also the left boundary of a CTU, or $y_{C0} = y_{LT0}$, $x_{C0} = x_{LT0} - offset$, if the first block is above-right to the current block and the right boundary of the current block is also the right boundary of a CTU, or $y_{C0} = y_{LT0}$, $x_{C0} = x_{LT0} + offset$, if $x_{LT0} - offset < x_{LT_AL}$, where x_{LT_AL} is the top-left coordinate of the neighbouring basic block above-left to the current block, or $y_{C0} = y_{LT0}$, $x_{C0} = x_{LT0} - offset$, if $x_{LT0} + offset > x_{RT_AR}$, where x_{RT_AR} is the top-right coordinate of the neighbouring basic block above-right to the current block, or $y_{C0} = y_{LT0}$, $x_{C0} = x_{LT0} + offset$, if $x_{LT0} - offset < x_{LT_AL}$, where x_{LT_AL} is the top-left coordinate of the neighbouring basic block above-left to the current block and left boundary of the current block is also the left boundary of a CTU, or $y_{C0} = y_{LT0}$, $x_{C0} = x_{LT0} - offset$, if $x_{LT0} + offset > x_{RT_AR}$, where x_{RT_AR} is the top-right coordinate of the neighbouring basic block above-right to the current block and the right boundary of the current block is also the right boundary of a CTU, where *offset* is a positive number.

[00314] In some examples, coordinates of top-left sample of the first basic block is (x_{LT0}, y_{LT0}) and coordinates of top-left positions of the M candidate basic block are $(x_{C0}, y_{C0}), (x_{C1}, y_{C1}), \dots, (x_{CM-1}, y_{CM-1})$, respectively, and when M is 2, $y_{C0} = y_{LT0}$, $x_{C0} = x_{LT0} - offset$, $y_{C1} = y_{LT0}$, $x_{C1} = x_{LT0} + offset$, if the first basic block is above-left to the current block, or $y_{C0} = y_{LT0}$, $x_{C0} = x_{LT0} - offset$, $y_{C1} = y_{LT0}$, $x_{C1} = x_{LT0} + offset$, if the first basic block is above-right to the current block, or $y_{C0} = y_{LT0}$, $x_{C0} = x_{LT0} - offset$, $y_{C1} = y_{LT0}$, $x_{C1} = x_{LT0} + offset$, if the first basic block is above-left to the current block and left boundary of the current block is also the left boundary of a CTU, or $y_{C0} = y_{LT0}$, $x_{C0} = x_{LT0} - offset$, $y_{C1} = y_{LT0}$, $x_{C1} = x_{LT0} + offset$, if the first basic block is above-right to the current block and the right boundary of the current block is also the right boundary of a CTU, or $y_{C0} = y_{LT0}$, $x_{C0} = x_{LT0} - offset$, $y_{C1} = y_{LT0}$, $x_{C1} = x_{LT0} + offset$, if $x_{LT0} - offset < x_{LT_AL}$, where x_{LT_AL} is the top-left coordinate of the neighbouring basic block above-left to the current block, or $y_{C0} = y_{LT0}$, $x_{C0} = x_{LT0} - offset$, $y_{C1} = y_{LT0}$, $x_{C1} = x_{LT0} + offset$, if $x_{LT0} + offset > x_{RT_AR}$, where x_{RT_AR} is the top-right coordinate of the neighbouring basic block above-right to the current block, or $y_{C0} = y_{LT0}$, $x_{C0} = x_{LT0} - offset$, $y_{C1} = y_{LT0}$, $x_{C1} = x_{LT0} + offset$, if $x_{LT0} - offset < x_{LT_AL}$, where x_{LT_AL} is the top-left coordinate of the neighbouring basic block above-left to the current block and left boundary of the current block is also the left boundary of a CTU, or $y_{C0} = y_{LT0}$, $x_{C0} = x_{LT0} - offset$, $y_{C1} = y_{LT0}$, $x_{C1} = x_{LT0} + offset$, if $x_{LT0} + offset > x_{RT_AR}$, where x_{RT_AR} is the top-right coordinate of the neighbouring basic block above-right to the current block and the right boundary of the current block is also the right boundary of a CTU, where *offset* is a positive number.

[00315] In some examples, when the second basic block, which is inter-coded and has a MV referring to the same reference picture as the MV of the first basic block referring to, cannot be found, the affine inherited motion vectors (MVs) for the current block cannot be derived from the first basic block.

[00316] In some examples, whether to and/or how to derive the affine inherited motion vectors (MVs) for the current block depends on the position of the current block.

[00317] In some examples, an affine model for the current block is inherited from an above neighbouring block of the current block in different ways depending on whether the neighbouring block is in a Coding Tree Unit (CTU) or CTU row where in the current block is located or not.

[00318] In some examples, when the affine model for the current block is inherited from an above or left neighbouring block of the current block, which is not in a CTU or CTU row where in the

current block is located, the affine inherited motion vectors (MVs) for the current block is derived based on the first stored motion vector (MV) and the second stored MV.

[00319] In some examples, whether to and/or how to select the second basic block from multiple candidates or from a predefined offset depends on a position of the first block and/or a position of the current block, and wherein the second basic block is a second neighboring basic block.

[00320] In some examples, an affine model for the current block is inherited from the first basic block neighboring the current block, which is affine coded and includes at least one of: an above neighbouring basic block (A), an above-right neighbouring basic block (AR) and an above left neighbouring basic block (AL), and wherein a top-left position of the first basic block is (xBB, yBB).

[00321] In some examples, when $xBB \% 8 = 0$, the second neighboring basic block is selected by:

[00322] checking whether a basic block on the right of the first basic block is affine coded and has a reference index same as that of the first basic block for a given reference list; and if yes, selecting the basic block on the right of the first basic block as the second neighboring basic block, or otherwise, selecting a basic block on the left of the first basic block as the second neighboring basic block.

[00323] In some examples, when $xBB \% 8 \neq 0$, the second neighboring basic block is selected by:

[00324] checking whether a basic block on the left of the first basic block is affine coded and has a reference index same as that of the first basic block for a given reference list; and if yes, selecting the block on the left of the first basic block as the second neighboring basic block, or otherwise, selecting that a block on the right of the first basic block is the second neighboring basic block.

[00325] In some examples, when the first basic block is AL and the left boundary of the current block is the left boundary of a CTU, the second neighboring basic block is selected by: only checking whether a basic block on the right of the first basic block is affine coded and has a reference index same as that of the first basic block for a given reference list; and if yes, selecting the basic block on the right of the first basic block as the second neighboring basic block, or otherwise, the affine model for the current block cannot be inherited from the first basic block.

[00326] In some examples, when $xBB \% S = 0$, the second neighboring basic block is selected by: checking whether a basic block on the right of the first basic block is affine coded and has a reference index same as that of the first basic block for a given reference list; and if yes, selecting the basic block on the right of the first basic block as the second neighboring basic block, or

otherwise, selecting a basic block on the left of the first basic block as the second neighboring basic block.

[00327] In some examples, when $x_{BB\%S!} = 0$, the second neighboring basic block is selected by: checking whether a basic block on the left of the first basic block is affine coded and has a reference index same as that of the first basic block for a given reference list; and if yes, selecting the block on the left of the first basic block as the second neighboring basic block, or otherwise, selecting that a block on the right of the first basic block is the second neighboring basic block.

[00328] In some examples, S is equal to 8

[00329] In some examples, the offset between the first basic block and the second basic block is a positive integer.

[00330] In some examples, the offset is in a form of 2^K , K is an integer, or depends on the minimum allowed CU width and/or height, or depend on width and/or height of the basic block, or depends on the minimum allowed width and/or height of a CU that affine coding is applicable, or is signaled from an encoder to a decoder.

[00331] In some examples, when a basic block P is chosen as the second basic block and when a basic block Q is the first basic block in a first affine inheritance process, the basic block Q is not allowed to be chosen as the second basic block when the basic block P is the first basic block in a second affine inheritance process.

[00332] In some examples, when a basic block P is chosen as the second basic block and when a basic block Q is the first basic block in a first affine inheritance process, the basic block P is not allowed to be the first basic block in the second affine inheritance process.

[00333] In some examples, when the basic block P is the first block in the second affine inheritance process, the second basic block in the second affine inheritance process can only be chosen from a basic block on the left of the basic block P if the basic block Q is on the right of the basic block P, or the second basic block can only be chosen from a basic block on the right of the basic block P if the basic block Q is on the left of the basic block P, or the second basic block can only be chosen from a basic block on the above of the basic block P if the basic block Q is on the below of the basic block P, or the second basic block can only be chosen from a basic block on the below of the basic block P if the basic block Q is on the above of the basic block P.

[00334] In some examples, the neighbouring basic block is on a row or column adjacent to the current block.

[00335] In some examples, the first basic block is determined as valid if it satisfies at least one of the following conditions: i. it is inter-coded; ii. it is not intra-block-copy coded; iii. it is affine-coded; iv. it is affine-merge coded; v. it is affine-inter coded.

[00336] In some examples, whether the second basic block is determined as valid depends on information of the first basic block.

[00337] In some examples, the second basic block is determined as valid if it satisfies at least one of the following conditions: i. it is inter-coded; ii. it is not intra-block-copy coded; iii. it is affine-coded; iv. it is affine-merge coded; v. it is affine-inter coded; vi. it has the same inter-prediction direction as the first basic block; vii. it has the same reference index for reference list 0 as the first basic block; viii. it has the same reference index for reference list 1 as the first basic block; ix. it has the same inter-prediction direction and same reference indices as the first basic block; x. it has the same picture-order-count (POC) value of the reference picture in reference list X as that for the first basic block, where X is 0 and/or 1.

[00338] In some examples, each basic block of above neighbouring basic blocks of the current block is checked in a predetermined order to determine whether it is a valid first basic block.

[00339] In some examples, the above neighbouring basic blocks of the current block are checked in the order from left to right or from right to left.

[00340] In some examples, the above neighbouring basic blocks include at least one of: an above neighbouring basic block (A), an above-right neighbouring basic block (AR) and a first above left neighbouring basic block (AL1).

[00341] In some examples, when one basic block (BB) of the above neighbouring basic blocks is a valid first basic block, a basic block on the left and/or right of the valid first basic block (BB) is checked to determine a corresponding second basic block.

[00342] In some examples, the basic block (BBR) on the right of the valid first basic block (BB) is checked first to determine whether it is a valid second basic block, and when the basic block (BBR) is the valid second basic block, the valid first basic block (BB) and the valid second basic block (BBR) are output as the first basic block and the second basic block.

[00343] In some examples, when the basic block (BBR) is not valid, the basic block (BBL) on the left of the valid first basic block (BB) is checked to determine whether it is the valid second basic block, and when the basic block (BBL) is the valid second basic block, the valid first basic block

(BB) and the valid second basic block (BBL) are output as the first basic block and the second basic block.

[00344] In some examples, when the basic block (BBL) is not valid, a next basic block left to the one basic block (BB) in order is checked to determine whether it is a valid first basic block.

[00345] In some examples, when the basic block (BBL) is not valid, no valid first basic block and valid second basic block are output from the above neighbouring basic blocks.

[00346] In some examples, the basic block (BBL) on the left of the valid first basic block (BB) is check first to determine whether it is a valid second basic block, and when the basic block (BBL) is the valid second basic block, the valid first basic block (BB) and the valid second basic block (BBL) are output as the first basic block and the second basic block.

[00347] In some examples, when the basic block (BBL) is not valid, the basic block (BBR) on the right of the valid first basic block (BB) is check to determine whether it is the valid second basic block, and when the basic block (BBR) is the valid second basic block, the valid first basic block (BB) and the valid second basic block (BBR) are output as the first basic block and the second basic block.

[00348] In some examples, when the basic block (BBR) is not valid, a next basic block right to the one basic block (BB) in order is checked to determine whether it is a valid first basic block.

[00349] In some examples, when the basic block (BBR) is not valid, no valid first basic block and valid second basic block are output from the above neighbouring basic blocks.

[00350] In some examples, only the basic block (BBR) on the right of the valid first basic block (BB) is check to determine whether it is a valid second basic block, and when the basic block (BBR) is the valid second basic block, the valid first basic block (BB) and the valid second basic block (BBR) are output as the first basic block and the second basic block.

[00351] In some examples, when the basic block (BBR) is not valid, the valid first basic block (BB) and the basic block (BBL) on the left of the valid first basic block are output as the first basic block and the second basic block.

[00352] In some examples, when the basic block (BBR) is not valid, a next basic block right to the one basic block (BB) in order is checked to determine whether it is a valid first basic block.

[00353] In some examples, when the basic block (BBL) is not valid, no valid first basic block and valid second basic block are output from the above neighbouring basic blocks.

[00354] In some examples, only the basic block (BBL) on the left of the valid first basic block (BB) is checked to determine whether it is a valid second basic block, and when the basic block (BBL) is the valid second basic block, the valid first basic block (BB) and the valid second basic block (BBL) are output as the first basic block and the second basic block.

[00355] In some examples, when the basic block (BBL) is not valid, the valid first basic block (BB) and the basic block (BBR) on the right of the valid first basic block are output as the first basic block and the second basic block.

[00356] In some examples, when the basic block (BBL) is not valid, a next basic block left to the one basic block (BB) in order is checked to determine whether it is a valid first basic block.

[00357] In some examples, when the basic block (BBL) is not valid, no valid first basic block and valid second basic block are output from the above neighbouring basic blocks.

[00358] In some examples, selection of candidate basic blocks from the above neighbouring basic blocks to be checked for the determination of the first basic blocks depends on the position of the current block and/or sub-block sizes of affine motion compensation.

[00359] In some examples, when the current block is at the left boundary of a CTU, the candidate basic blocks includes at least one of: an above neighbouring basic block (A), an above-right neighbouring basic block (AR) and a first above left neighbouring basic block (AL1), and when the current block is not at the left boundary of a CTU, the candidate basic blocks includes at least one of: an above neighbouring basic block (A), an above-right neighbouring basic block (AR) and a second above left adjacent neighbouring basic block (AL'').

[00360] In some examples, whether a candidate basic blocks from the above neighbouring basic blocks can be used as the first basic blocks depends on the position of the current block.

[00361] In some examples, when the current block is at the left boundary of a CTU, an above left neighbouring basic block (AL1, AL'') cannot be used as the first basic block.

[00362] In some examples, when the current block is at the right boundary of a CTU, an above-right neighbouring basic block (AR, AR') cannot be used as the first basic block.

[00363] In some examples, whether a candidate basic blocks from the above neighbouring basic blocks can be used as the second basic blocks depends on the position of the current block.

[00364] In some examples, when the current block is at the left boundary of a CTU, an above left neighbouring basic block (AL1, AL') cannot be used as the first basic block.

[00365] In some examples, when the current block is at the right boundary of a CTU, an above-right neighbouring basic block (AR, AR') cannot be used as the first basic block.

[00366] In some examples, the first basic block and the second basic block are exchangeable.

[00367] In some examples, the first basic block and second basic block are firstly exchanged, and the conversion of the current block is performed by using the exchanged first basic block and second basic block.

[00368] In some examples, the determination process of the first basic block and the second basic block are exchangeable.

[00369] In some examples, the conversion generates the current block of video from the bitstream representation.

[00370] In some examples, the conversion generates the bitstream representation from the current block of video.

[00371] FIG. 32 is a flowchart for a method 3200 of processing video. The method 3200 includes, deriving (3202), for a conversion between a current block of video and a bitstream representation of the current block, one or more parameters of a set of affine model parameters associated with affine model for the current block; shifting (3204) the one or more parameters; and storing (3206) the shifted one or more parameters.

[00372] In some examples, the shifting the one or more parameters further comprises shifting the one or more parameters with a first shift function $SatShift(x, n)$, which is defined as:

$$SatShift(x, n) = \begin{cases} (x + offset0) \gg n & \text{if } x \geq 0 \\ -((-x + offset1) \gg n) & \text{if } x < 0 \end{cases},$$

wherein x is one of the one or more parameters, n is an integer, and $offset0$ and/or $offset1$ are set to $(1 \ll n) \gg 1$ or $(1 \ll (n-1))$, or $((1 \ll n) \gg 1) - 1$, or $offset0$ and/or $offset1$ are set to 0.

[00373] In some examples, the shifting the one or more parameters further comprises shifting the one or more parameters with a second shift function $Shift(x, n)$, which is defined as:

$$Shift(x, n) = (x + offset0) \gg n,$$

wherein x is one of the one or more parameters, n is an integer, and $offset0$ is set to $(1 \ll n) \gg 1$ or $(1 \ll (n-1))$, or $((1 \ll n) \gg 1) - 1$, or $offset0$ is set to 0.

[00374] In some examples, n is 2 or 4.

[00375] In some examples, n depends on motion precision, or n is different for different parameters in the set of affine model parameters.

[00376] In some examples, n is signaled in at least one of Sequence Parameter Set (SPS), Video Parameter Set (VPS), Picture Parameter Set (PPS), slice header, tile group header, tile, coding tree unit (CTU), coding unit (CU).

[00377] In some examples, n is different in different standard profiles or levels or tiers.

[00378] In some examples, the stored parameter is left shift first before it is used in affine inheritance of blocks coded after the current block.

[00379] In some examples, the stored parameter is shift with a shift function and clipped with a clip function sequentially before it is used in affine inheritance of blocks coded after the current block.

[00380] In some examples, a parameter a of the set of affine model parameters is calculated by

$a = \frac{(mv_1^h - mv_0^h)}{w}$, where mv_0^h is a horizontal motion vector component of a top-left corner control point of the current block, mv_1^h is a horizontal motion vector component of a top-right corner control point of the current block, and w is a width of the current block.

[00381] In some examples, a parameter b of the set of affine model parameters is calculated by

$b = \frac{(mv_1^v - mv_0^v)}{w}$, where mv_0^v is a vertical motion vector component of a top-left corner control point of the current block, mv_1^v is a vertical motion vector component of a top-right corner control point of the current block, and w is a width of the current block.

[00382] In some examples, a parameter c of the set of affine model parameters is calculated by

$c = \frac{(mv_2^h - mv_0^h)}{h}$ where mv_0^h is a horizontal motion vector component of a top-left corner control point of the current block, mv_2^h is a horizontal motion vector component of a bottom-left corner control point of the current block, and h is a height of the current block.

[00383] In some examples, a parameter d of the set of affine model parameters is calculated by

$d = \frac{(mv_2^v - mv_0^v)}{h}$ where mv_0^v is a vertical motion vector component of a top-left corner control point of the current block, mv_2^v is a vertical motion vector component of a bottom-left corner control point of the current block, and h is a height of the current block.

[00384] In some examples, a parameter e of the set of affine model parameters is calculated by $e = mv_0^h$, where mv_0^h is a horizontal motion vector component of a top-left corner control point of the current block.

[00385] In some examples, a parameter f of the set of affine model parameters is calculated by $f = mv_0^v$, where mv_0^v is a vertical motion vector component of a top-left corner control point of the current block.

[00386] In some examples, parameters e and f of the set of affine model parameters are calculated by $(e, f) = (mv_{xi}, mv_{yi})$, where (mv_{xi}, mv_{yi}) is a motion vector of any point.

[00387] In some examples, the width and height of the current block are noted as w and h are equal to 2^{WB} and 2^{HB} , where WB and HB are integers greater than one.

[00388] In some examples, a parameter a of the set of affine model parameters is calculated by $a = SatShift(P \cdot (mv_1^h - mv_0^h), WB)$, where P is an integer and represents a calculation precision, mv_0^h is a horizontal motion vector component of a top-left corner control point of the current block, and mv_1^h is a horizontal motion vector component of a top-right corner control point of the current block.

[00389] In some examples, wherein a parameter b of the set of affine model parameters is calculated by $b = SatShift(P \cdot (mv_1^v - mv_0^v), WB)$, where P is an integer and represents a calculation precision, mv_0^v is a vertical motion vector component of a top-left corner control point of the current block, and mv_1^v is a vertical motion vector component of a top-right corner control point of the current block.

[00390] In some examples, a parameter c of the set of affine model parameters is calculated by $c = SatShift(P \cdot (mv_2^h - mv_0^h), HB)$, where P is an integer and represents a calculation precision, mv_0^h is a horizontal motion vector component of a top-left corner control point of the current block, and mv_2^h is a horizontal motion vector component of a bottom-left corner control point of the current block.

[00391] In some examples, a parameter d of the set of affine model parameters is calculated by $d = SatShift(P \cdot (mv_2^v - mv_0^v), HB)$, where P is an integer and represents a calculation precision, mv_0^v is a vertical motion vector component of a top-left corner control point of the current block,

and mv^v_2 is a vertical motion vector component of a bottom-left corner control point of the current block.

[00392] In some examples, P is set to 7.

[00393] In some examples, the method further comprises: clipping the one or more parameters, prior to the storing the one or more parameters.

[00394] In some examples, if one of the one or more parameters, X, is stored with K bits, then $X = \text{Clip3}(-2^{K-1}, 2^{K-1}-1, X)$, where $X=a$ or b or c or d , and K is an integer that is greater than one.

[00395] In some examples, X is a, b, c, d, e, or f.

[00396] In some examples, K is equal to 8.

[00397] In some examples, the set of affine model parameters comprises six variables (a, b, c, d, e, f) corresponding to a six-parameter affine model given by

$$\begin{cases} mv^h(x, y) = ax + cy + e = \frac{(mv^h_1 - mv^h_0)}{w}x + \frac{(mv^h_2 - mv^h_0)}{h}y + mv^h_0 \\ mv^v(x, y) = bx + dy + f = \frac{(mv^v_1 - mv^v_0)}{w}x + \frac{(mv^v_2 - mv^v_0)}{h}y + mv^v_0 \end{cases}$$

where $mv^h(x, y)$ is a horizontal component of a motion vector of the current block, $mv^v(x, y)$ is a vertical component of a motion vector of the current block, and (x, y) represents the coordinate of a representative point relative to a top-left sample within the current block; (mv^h_0, mv^v_0) is a motion vector of a top-left corner control point (CP), and (mv^h_1, mv^v_1) is a motion vector of a top-right corner control point and (mv^h_2, mv^v_2) is a motion vector of a bottom-left corner control point for the current block.

[00398] In some examples, the one or more parameters comprise a, b, c , and d .

[00399] In some examples, the set of affine model parameters comprises four variables (a, b, e, f) corresponding to a four-parameter affine model given by

$$\begin{cases} mv^h(x, y) = ax - by + e = \frac{(mv^h_1 - mv^h_0)}{w}x - \frac{(mv^v_1 - mv^v_0)}{w}y + mv^h_0 \\ mv^v(x, y) = bx + ay + f = \frac{(mv^v_1 - mv^v_0)}{w}x + \frac{(mv^h_1 - mv^h_0)}{w}y + mv^v_0 \end{cases}$$

where $mv^h(x, y)$ is a horizontal component of a motion vector of the current block, $mv^v(x, y)$ is a vertical component of a motion vector of the current block, and (x, y) represents the coordinate of a representative point relative to a top-left sample within the current block; (mv^h_0, mv^v_0) is a motion vector of a top-left corner control point (CP), and (mv^h_1, mv^v_1) is a motion vector of a top-right corner control point for the current block.

- [00400] In some examples, the one or more parameters comprise a and b .
- [00401] In some examples, the one or more parameters comprise a , b , e and f .
- [00402] In some examples, the one or more parameters comprise a , b , c , d , e and f , and wherein it is restricted that $c=-b$ and $d=a$, when the conversion between the current block and the bitstream representation of the current block is performed with a four-parameter affine model.
- [00403] In some examples, the one or more parameters comprise a , b , c and d , and wherein it is restricted that $c=-b$ and $d=a$, when the conversion between the current block and the bitstream representation of the current block is performed with a four-parameter affine mode.
- [00404] In some examples, the parameter $c=-b$, when the conversion between the current block and the bitstream representation of the current block is performed with a four-parameter affine mode.
- [00405] In some examples, the parameter $d=a$, when the conversion between the current block and the bitstream representation of the current block is performed with a four-parameter affine mode.
- [00406] In some examples, the method further comprises: performing a conversion between a block coded after the current block and a bitstream representation of the block coded after the current block, based on the stored shifted one or more parameters.
- [00407] In some examples, the method further comprises: performing, based stored on the one or more parameters, the conversion between the current block and the bitstream representation of the current block.
- [00408] In some examples, the conversion generates the block coded after the current block from the bitstream representation.
- [00409] In some examples, the conversion generates the bitstream representation from the block coded after the current block.
- [00410] It will be appreciated by one of skill in the art that techniques for using motion candidate lists under various video coding scenarios are disclosed. Video blocks may be encoded into bitstream representations that include non-contiguous bits that are placed in various headers or in network adaption layer, and so on.
- [00411] The disclosed and other solutions, examples, 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.

[00412] 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.

[00413] 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).

[00414] 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.

[00415] While this patent document contains many specifics, these should not be construed as limitations on the scope of any subject matter or of what may be claimed, but rather as descriptions of features that may be specific to particular embodiments of particular techniques. 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.

[00416] 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.

[00417] 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:

deriving, for a conversion between a current block and a bitstream representation of the current block, one or more parameters of a set of affine model parameters associated with affine model for the current block;

shifting the one or more parameters; and

storing the shifted one or more parameters.

2. The method of claim 1, wherein the shifting the one or more parameters further comprises shifting the one or more parameters with a first shift function $SatShift(x, n)$, which is defined as:

$$SatShift(x, n) = \begin{cases} (x + offset0) \gg n & \text{if } x \geq 0 \\ -((-x + offset1) \gg n) & \text{if } x < 0 \end{cases},$$

wherein x is one of the one or more parameters, n is an integer, and $offset0$ and/or $offset1$ are set to $(1 \ll n) \gg 1$ or $(1 \ll (n-1))$, or $((1 \ll n) \gg 1) - 1$, or $offset0$ and/or $offset1$ are set to 0.

3. The method of claim 1, wherein the shifting the one or more parameters further comprises shifting the one or more parameters with a second shift function $Shift(x, n)$, which is defined as:

$$Shift(x, n) = (x + offset0) \gg n,$$

wherein x is one of the one or more parameters, n is an integer, and $offset0$ is set to $(1 \ll n) \gg 1$ or $(1 \ll (n-1))$, or $((1 \ll n) \gg 1) - 1$, or $offset0$ is set to 0.

4. The method of claim 2 or 3, wherein n is 2 or 4.

5. The method of claim 2 or 3, wherein n depends on motion precision, or n is different for different parameters in the set of affine model parameters.

6. The method of claim 2 or 3, wherein n is signaled in at least one of Sequence Parameter Set (SPS), Video Parameter Set (VPS), Picture Parameter Set (PPS), slice header, tile group header, tile, coding tree unit (CTU), coding unit (CU).

7. The method of claim 2 or 3, wherein n is different in different standard profiles or levels or tiers.

8. The method of any one of claims 1-7, wherein the stored parameter is left shift first before it is used in affine inheritance of blocks coded after the current block.

9. The method of any one of claims 1-8, wherein the stored parameter is shift with a shift function and clipped with a clip function sequentially before it is used in affine inheritance of blocks coded after the current block.

10. The method of anyone of claims 1-9, wherein a parameter a of the set of affine model

parameters is calculated by $a = \frac{(mv_1^h - mv_0^h)}{w}$, where mv_0^h is a horizontal motion vector

component of a top-left corner control point of the current block, mv_1^h is a horizontal motion vector component of a top-right corner control point of the current block, and w is a width of the current block.

11. The method of anyone of claims 1-9, wherein a parameter b of the set of affine model

parameters is calculated by $b = \frac{(mv_1^v - mv_0^v)}{w}$, where mv_0^v is a vertical motion vector component

of a top-left corner control point of the current block, mv_1^v is a vertical motion vector component of a top-right corner control point of the current block, and w is a width of the current block.

12. The method of anyone of claims 1-9, wherein a parameter c of the set of affine model

parameters is calculated by $c = \frac{(mv_2^h - mv_0^h)}{h}$ where mv_0^h is a horizontal motion vector

component of a top-left corner control point of the current block, mv_2^h is a horizontal motion vector component of a bottom-left corner control point of the current block, and h is a height of the current block.

13. The method of anyone of claims 1-9, wherein a parameter d of the set of affine model

parameters is calculated by $d = \frac{(mv_2^v - mv_0^v)}{h}$ where mv_0^v is a vertical motion vector component

of a top-left corner control point of the current block, mv^v_2 is a vertical motion vector component of a bottom-left corner control point of the current block, and h is a height of the current block.

14. The method of anyone of claims 1-13, wherein a parameter e of the set of affine model parameters is calculated by $e = mv^h_0$, where mv^h_0 is a horizontal motion vector component of a top-left corner control point of the current block.

15. The method of anyone of claims 1-9, wherein a parameter f of the set of affine model parameters is calculated by $f = mv^v_0$, where mv^v_0 is a vertical motion vector component of a top-left corner control point of the current block.

16. The method of anyone of claims 1-9, wherein parameters e and f of the set of affine model parameters are calculated by $(e, f) = (mv_{xi}, mv_{yi})$, where (mv_{xi}, mv_{yi}) is a motion vector of any point.

17. The method of anyone of claims 1-16, wherein the width and height of the current block are noted as w and h are equal to 2^{WB} and 2^{HB} , where WB and HB are integers greater than one.

18. The method of claim 17, wherein a parameter a of the set of affine model parameters is calculated by $a = SatShift(P \cdot (mv^h_1 - mv^h_0), WB)$, where P is an integer and represents a calculation precision, mv^h_0 is a horizontal motion vector component of a top-left corner control point of the current block, and mv^h_1 is a horizontal motion vector component of a top-right corner control point of the current block.

19. The method of claim 17, wherein a parameter b of the set of affine model parameters is calculated by $b = SatShift(P \cdot (mv^v_1 - mv^v_0), WB)$, where P is an integer and represents a calculation precision, mv^v_0 is a vertical motion vector component of a top-left corner control point of the current block, and mv^v_1 is a vertical motion vector component of a top-right corner control point of the current block.

20. The method of claim 17, wherein a parameter c of the set of affine model parameters is calculated by $c = SatShift(P \cdot (mv^h_2 - mv^h_0), HB)$, where P is an integer and represents a calculation

precision, mv^h_0 is a horizontal motion vector component of a top-left corner control point of the current block, and mv^h_2 is a horizontal motion vector component of a bottom-left corner control point of the current block.

21. The method of claim 17, wherein a parameter d of the set of affine model parameters is calculated by $d = SatShift(P \cdot (mv^v_2 - mv^v_0), HB)$, where P is an integer and represents a calculation precision, mv^v_0 is a vertical motion vector component of a top-left corner control point of the current block, and mv^v_2 is a vertical motion vector component of a bottom-left corner control point of the current block.

22. The method of anyone of claims 18-21, wherein P is set to 7.

23. The method of anyone of claims 1-22, further comprising: clipping the one or more parameters, prior to the storing the one or more parameters.

24. The method of claim 23, wherein if one of the one or more parameters, X , is stored with K bits, then $X = Clip3(-2^{K-1}, 2^{K-1}-1, X)$, where $X=a$ or b or c or d , and K is an integer that is greater than one.

25. The method of claim 24, wherein X is a , b , c , d , e , or f .

26. The method of claim 24, wherein K is equal to 8.

27. The method of anyone of claims 1 to 26, wherein the set of affine model parameters comprises six variables (a , b , c , d , e , f) corresponding to a six-parameter affine model given by

$$\begin{cases} mv^h(x, y) = ax + cy + e = \frac{(mv^h_1 - mv^h_0)}{w}x + \frac{(mv^h_2 - mv^h_0)}{h}y + mv^h_0 \\ mv^v(x, y) = bx + dy + f = \frac{(mv^v_1 - mv^v_0)}{w}x + \frac{(mv^v_2 - mv^v_0)}{h}y + mv^v_0 \end{cases}$$

where $mv^h(x, y)$ is a horizontal component of a motion vector of the current block, $mv^v(x, y)$ is a vertical component of a motion vector of the current block, and (x, y) represents the coordinate of a representative point relative to a top-left sample within the current block; (mv^h_0, mv^v_0) is a motion vector of a top-left corner control point (CP), and (mv^h_1, mv^v_1) is a motion vector of a top-

right corner control point and (mv^h_2, mv^v_2) is a motion vector of a bottom-left corner control point for the current block.

28. The method of claim 27, wherein the one or more parameters comprise a , b , c , and d .

29. The method of anyone of claims 1 - 26, wherein the set of affine model parameters comprises four variables (a, b, e, f) corresponding to a four-parameter affine model given by

$$\begin{cases} mv^h(x, y) = ax - by + e = \frac{(mv^h_1 - mv^h_0)}{w}x - \frac{(mv^v_1 - mv^v_0)}{w}y + mv^h_0 \\ mv^v(x, y) = bx + ay + f = \frac{(mv^v_1 - mv^v_0)}{w}x + \frac{(mv^h_1 - mv^h_0)}{w}y + mv^v_0 \end{cases}$$

where $mv^h(x, y)$ is a horizontal component of a motion vector of the current block, $mv^v(x, y)$ is a vertical component of a motion vector of the current block, and (x, y) represents the coordinate of a representative point relative to a top-left sample within the current block; (mv^h_0, mv^v_0) is a motion vector of a top-left corner control point (CP), and (mv^h_1, mv^v_1) is a motion vector of a top-right corner control point for the current block.

30. The method of claim 29, wherein the one or more parameters comprise a and b .

31. The method of claim 29, wherein the one or more parameters comprise a , b , e and f .

32. The method of claim 27, wherein the one or more parameters comprise a , b , c , d , e and f , and wherein it is restricted that $c=-b$ and $d=a$, when the conversion between the current block and the bitstream representation of the current block is performed with a four-parameter affine model.

33. The method of claim 27, wherein the one or more parameters comprise a , b , c and d , and wherein it is restricted that $c=-b$ and $d=a$, when the conversion between the current block and the bitstream representation of the current block is performed with a four-parameter affine mode.

34. The method of claim 28 or 29, wherein the parameter $c=-b$, when the conversion between the current block and the bitstream representation of the current block is performed with a four-parameter affine mode.

35. The method of claim 28 or 29, wherein the parameter $d=a$, when the conversion between the current block and the bitstream representation of the current block is performed with a four-parameter affine mode.

36. The method of any one of claims 1-35, further comprising:

performing a conversion between a block coded after the current block and a bitstream representation of the block coded after the current block, based on the stored shifted one or more parameters.

37. The method of claim 36, wherein the conversion generates the block coded after the current block from the bitstream representation.

38. The method of claim 36, wherein the conversion generates the bitstream representation from the block coded after the current block.

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

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

41. A video decoding apparatus comprising a processor configured to implement a method recited in one or more of claims 1 to 38.

42. A video encoding apparatus comprising a processor configured to implement a method recited in one or more of claims 1 to 38.

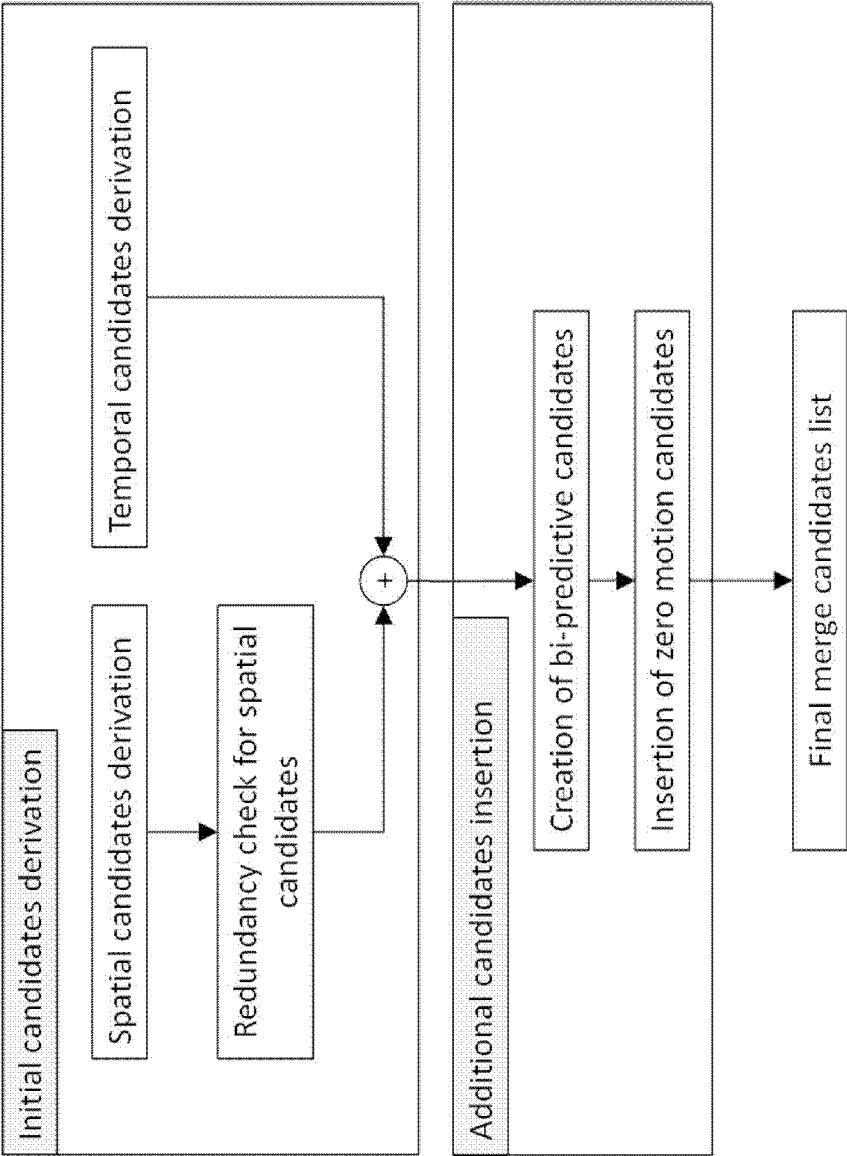


FIG. 1

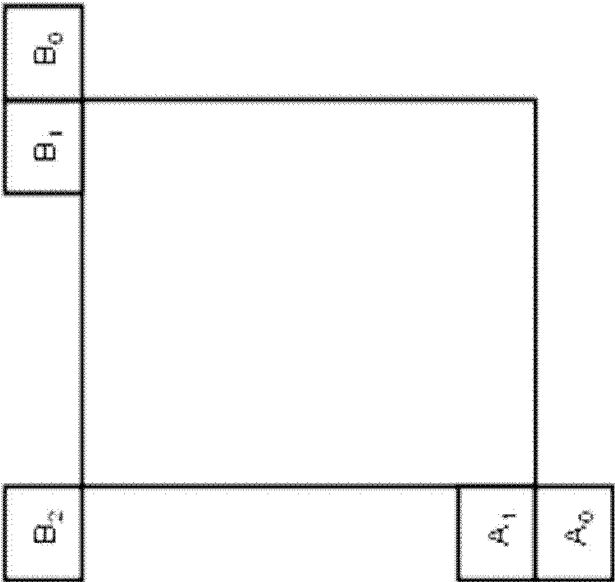


FIG. 2

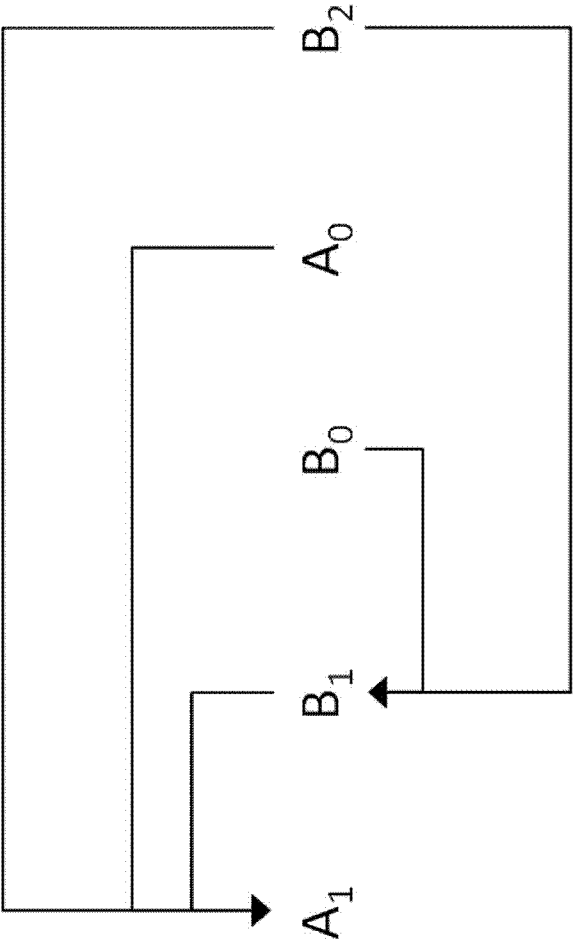


FIG. 3

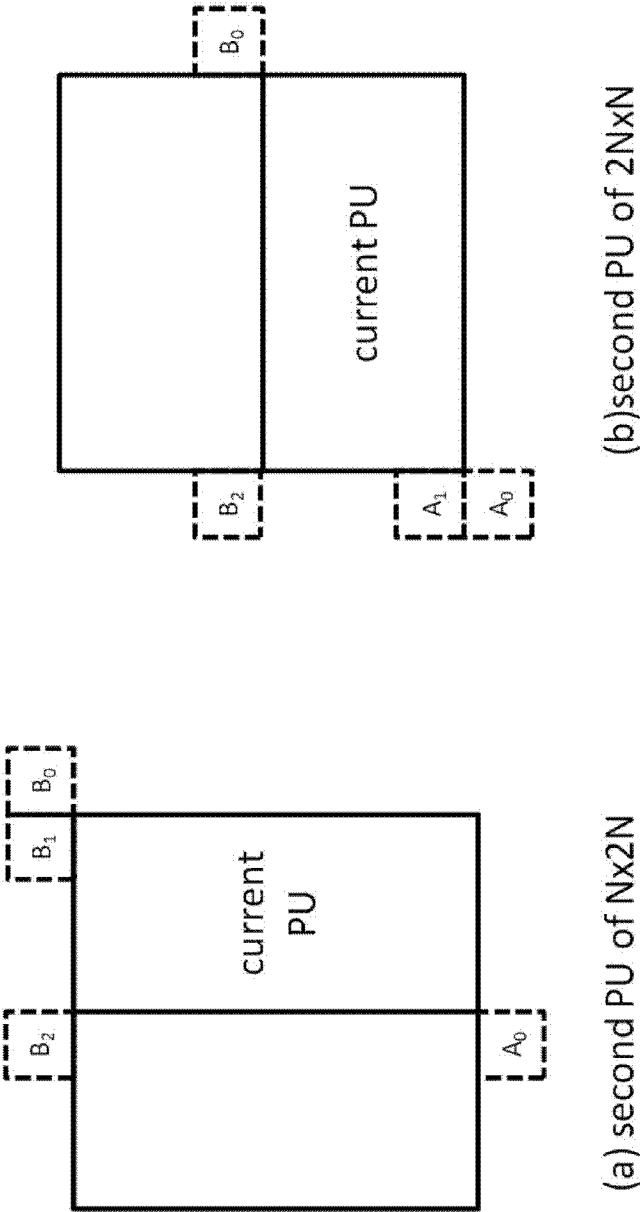


FIG. 4A

FIG. 4B

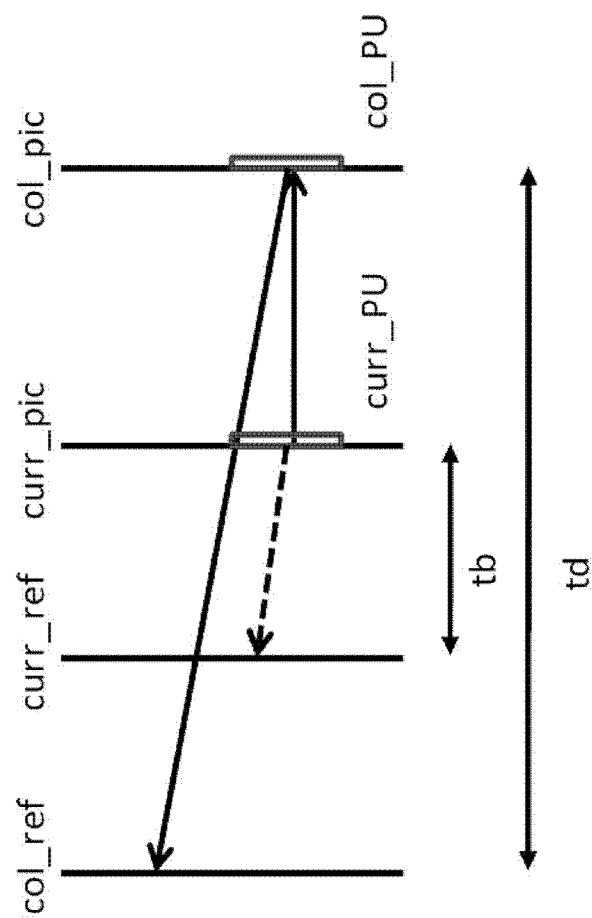


FIG. 5

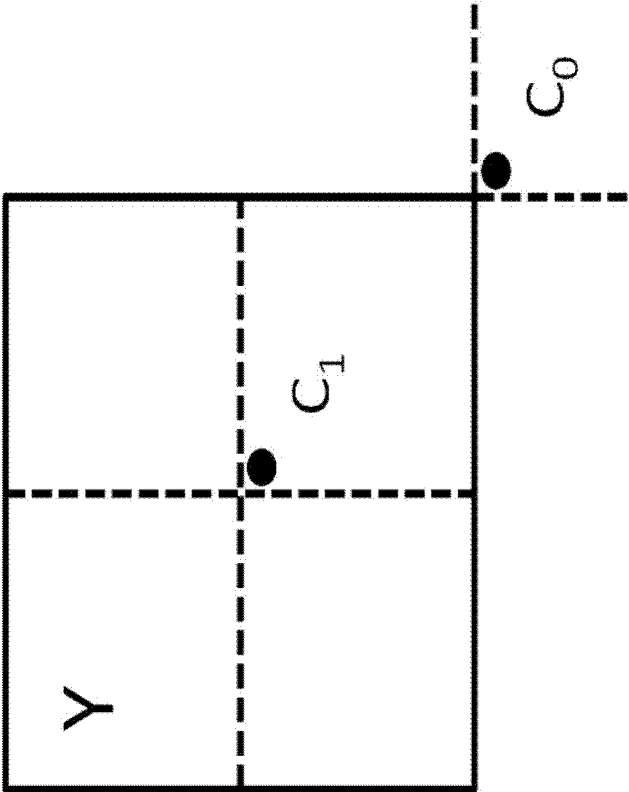


FIG. 6

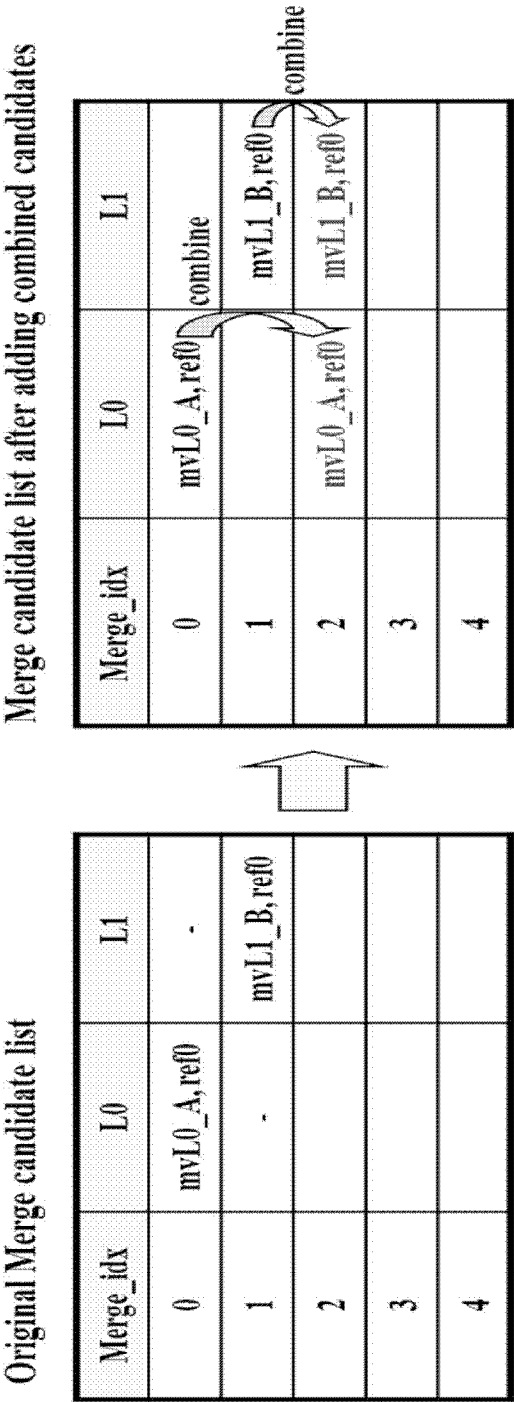


FIG. 7

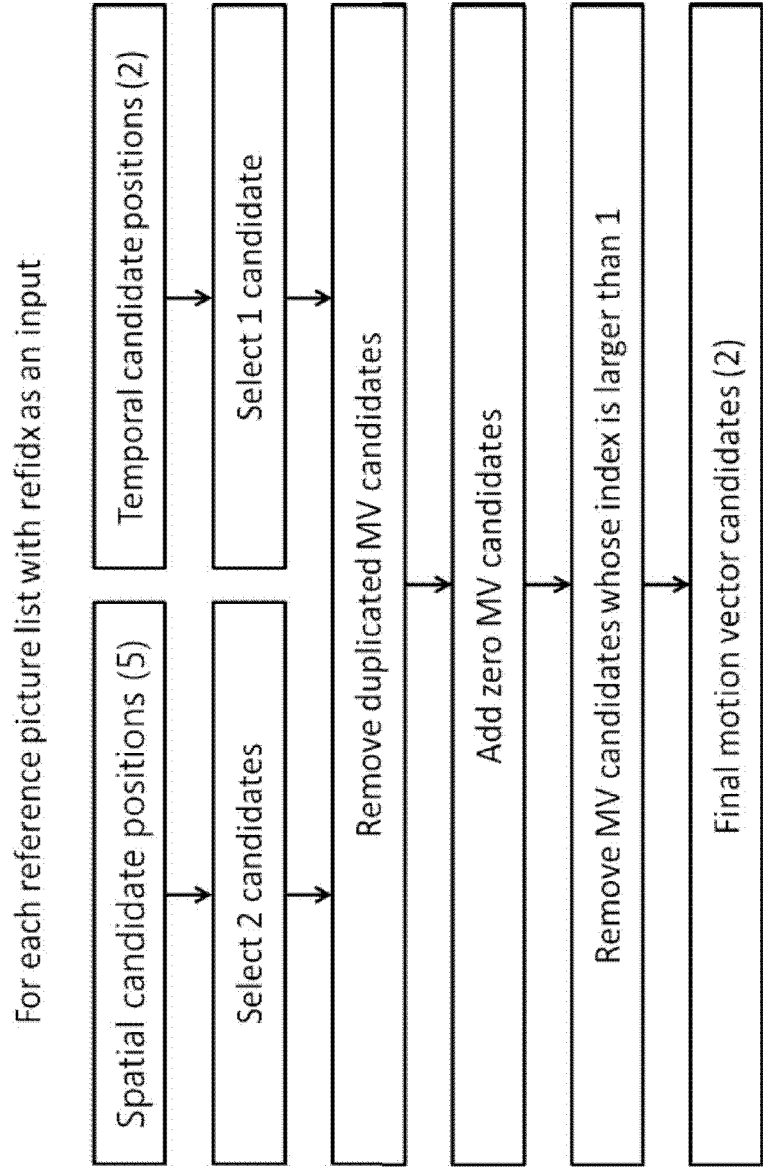


FIG. 8

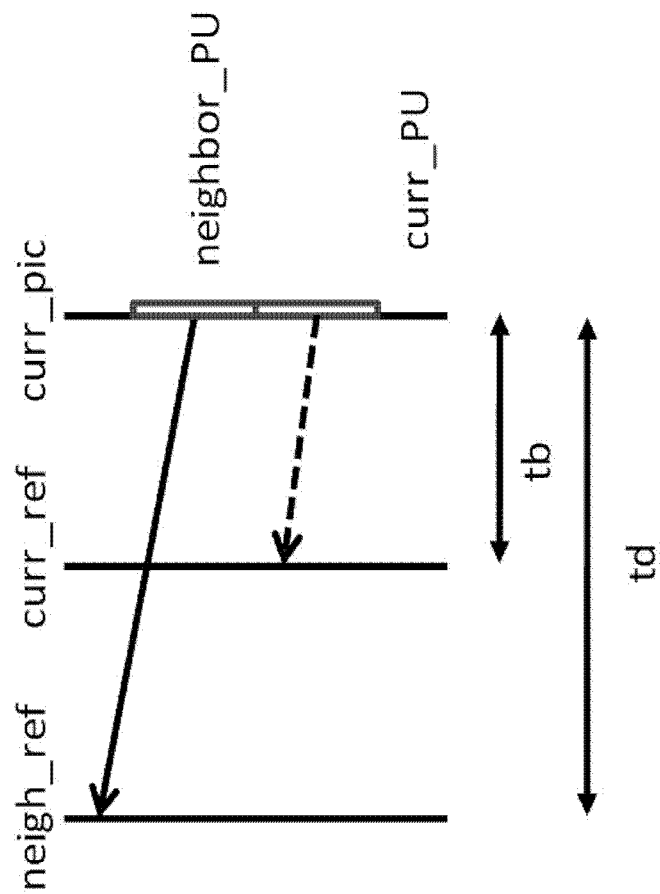


FIG. 9

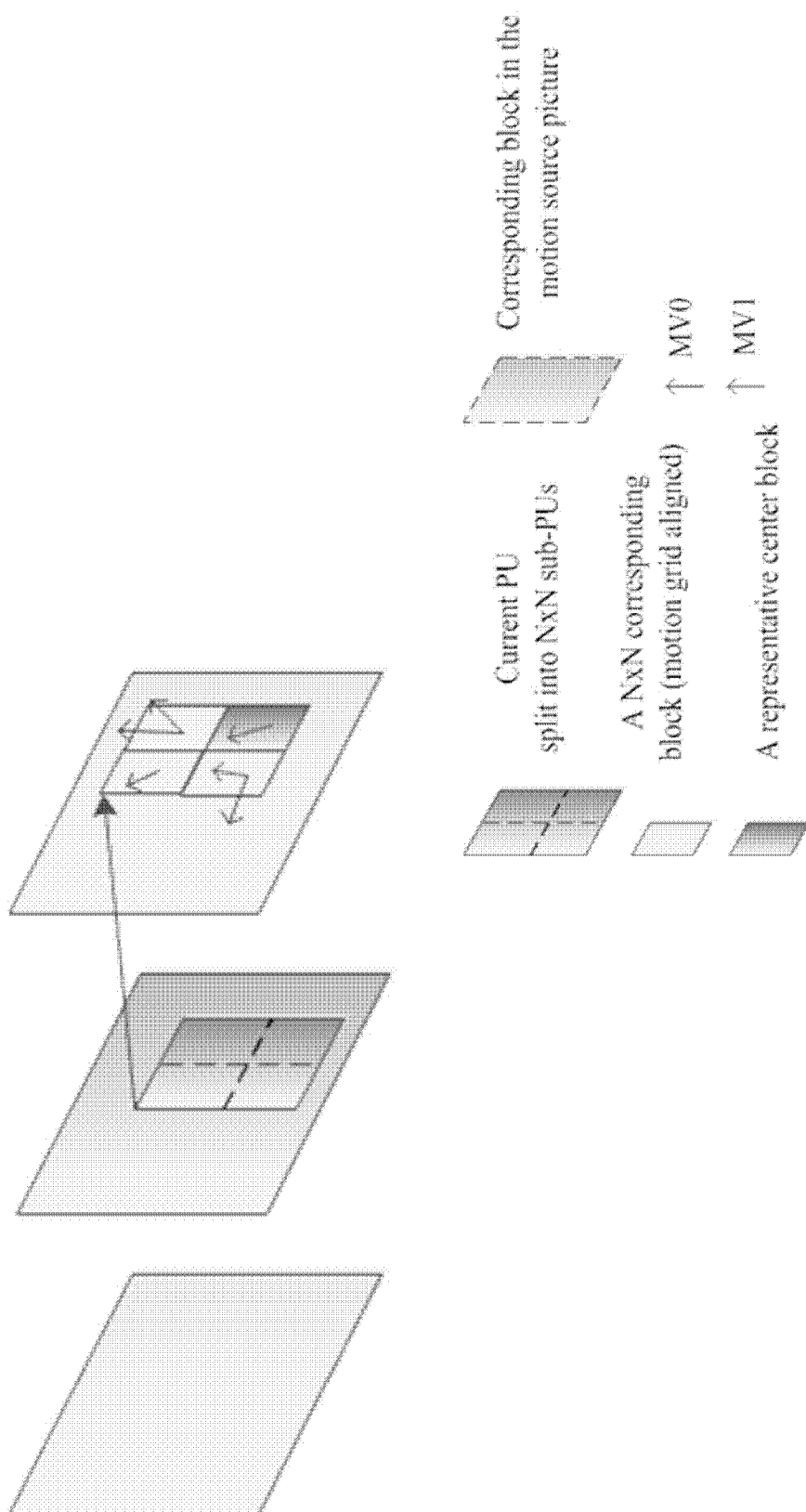


FIG. 10

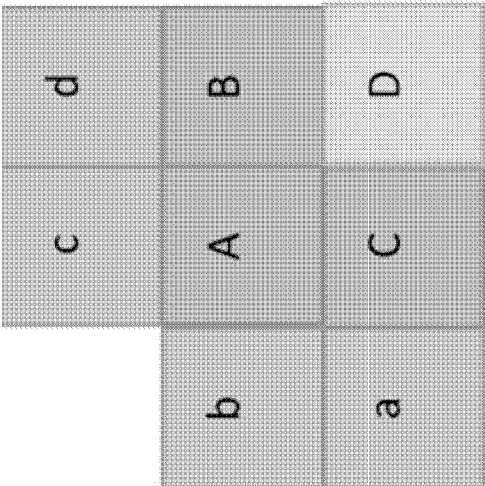


FIG. 11

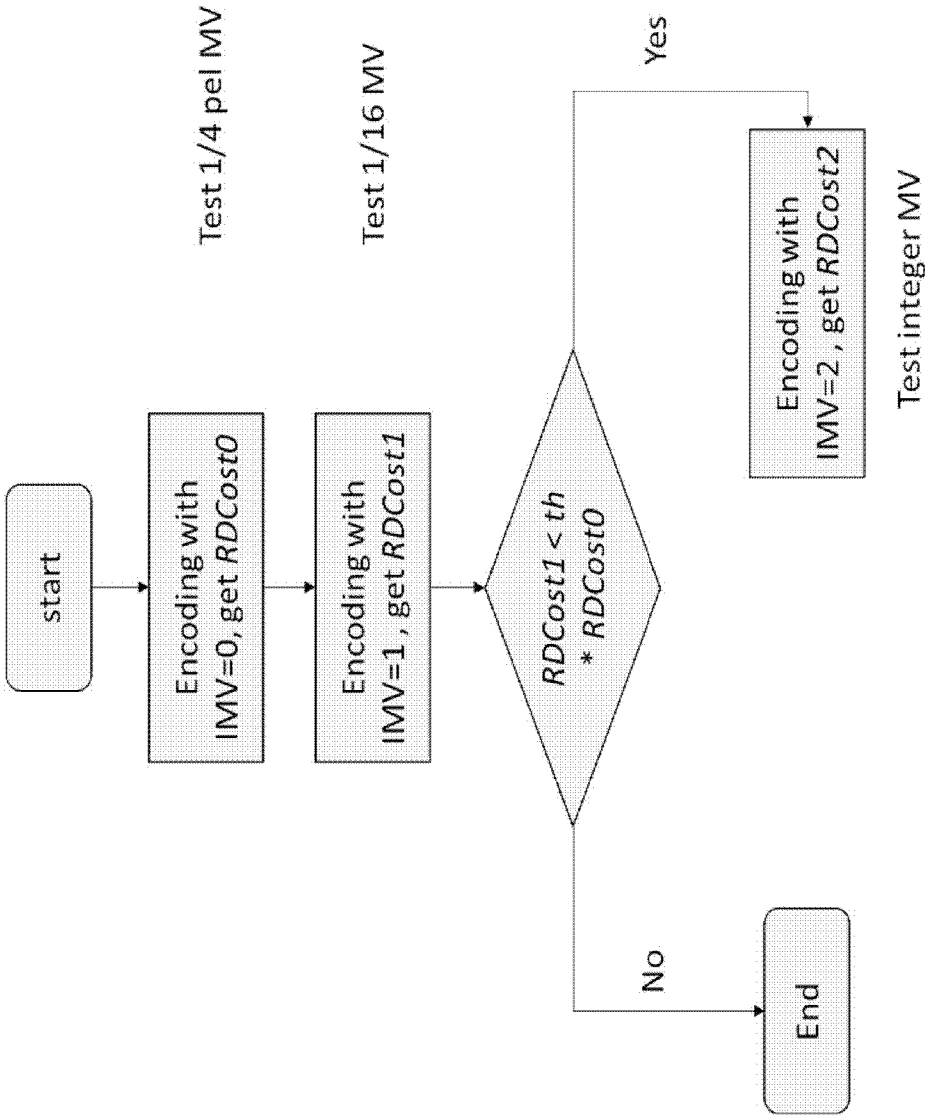


FIG. 12

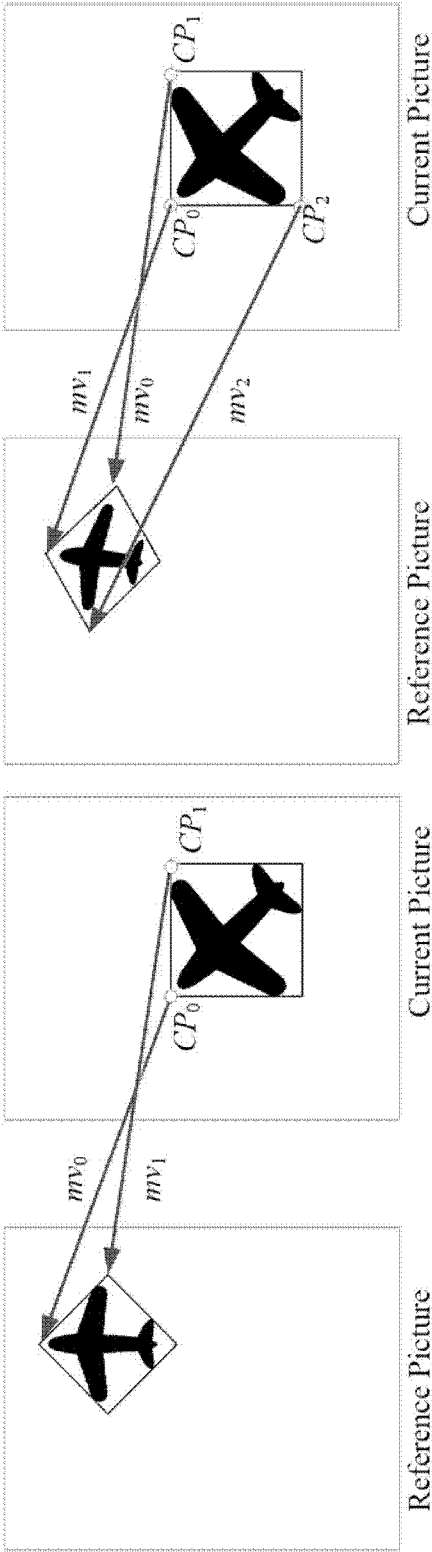


FIG. 13A

FIG. 13B

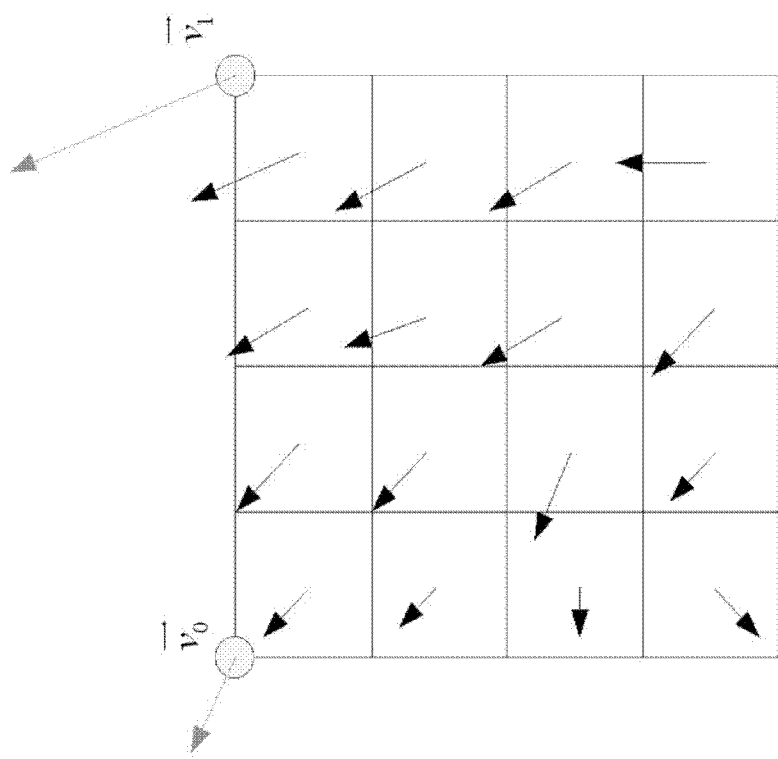


FIG. 14

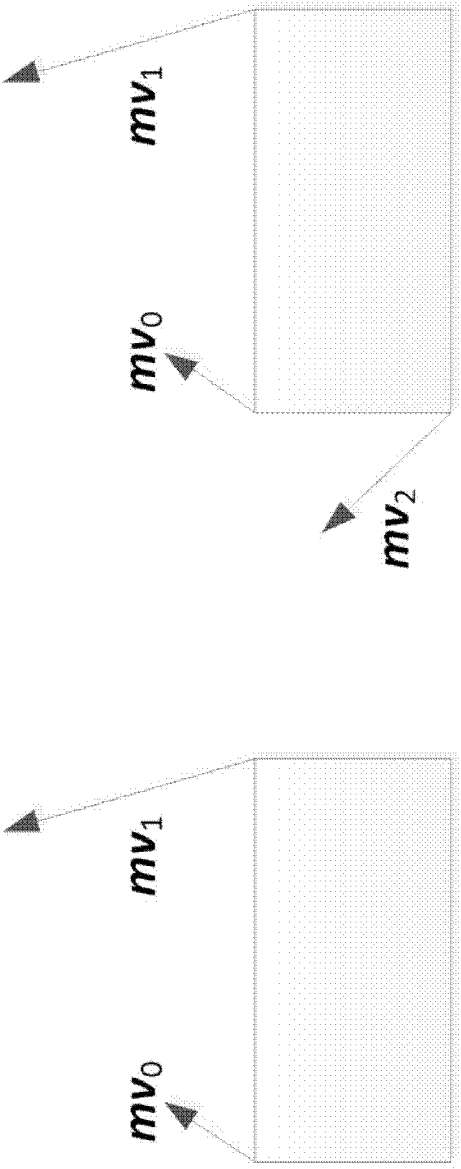


FIG. 15B

FIG. 15A

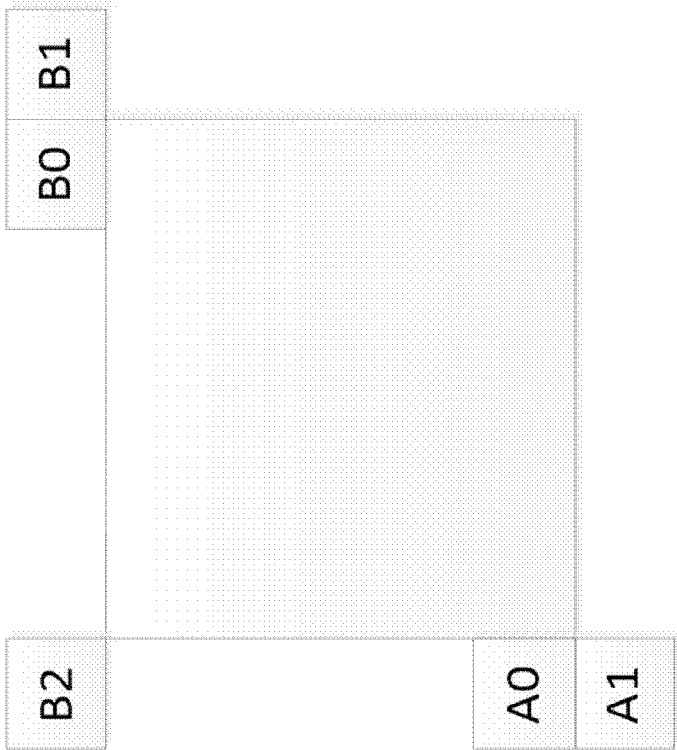


FIG. 16

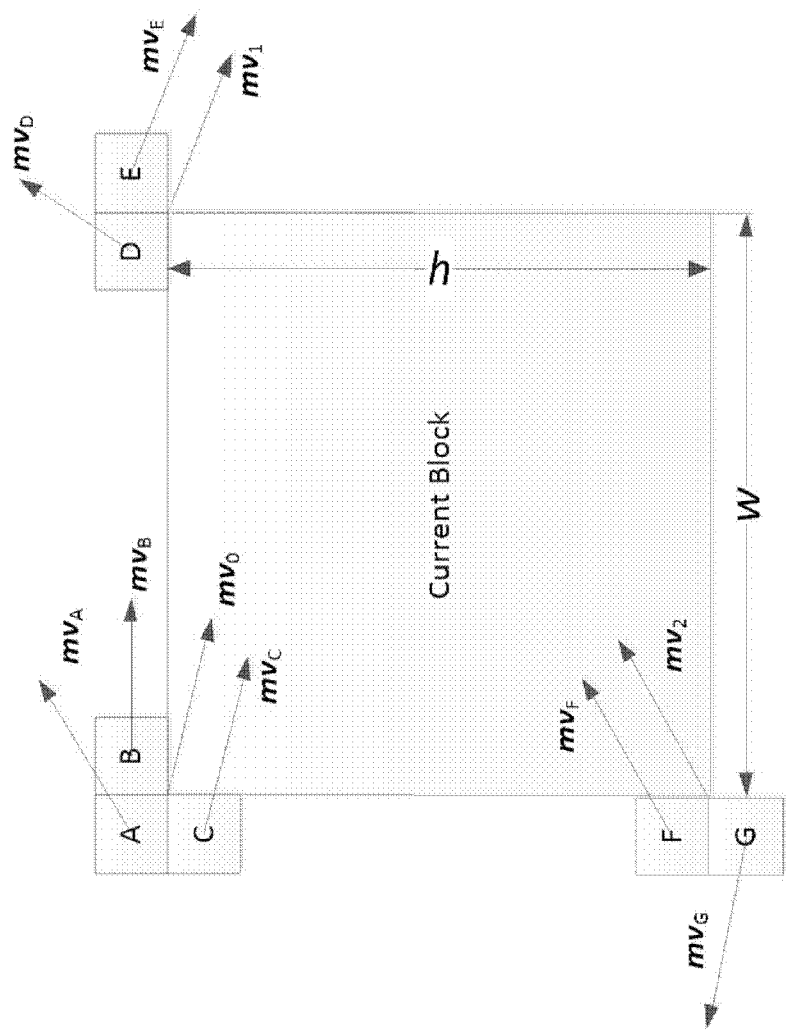
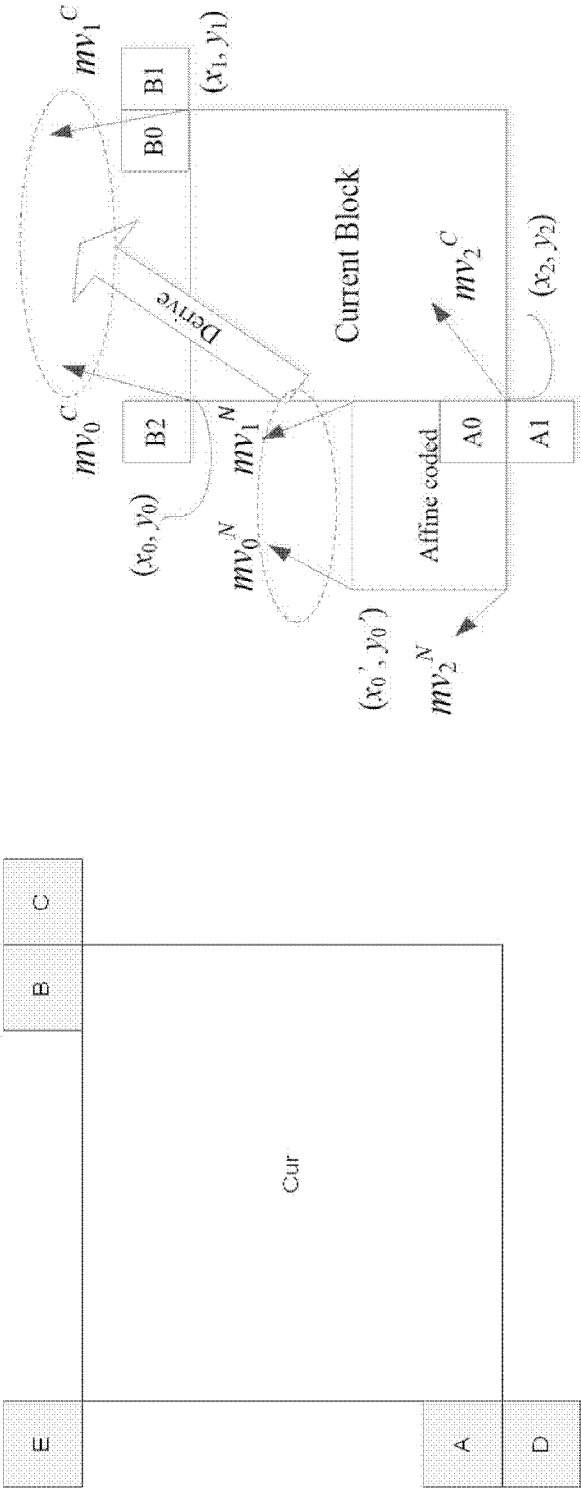


FIG. 17



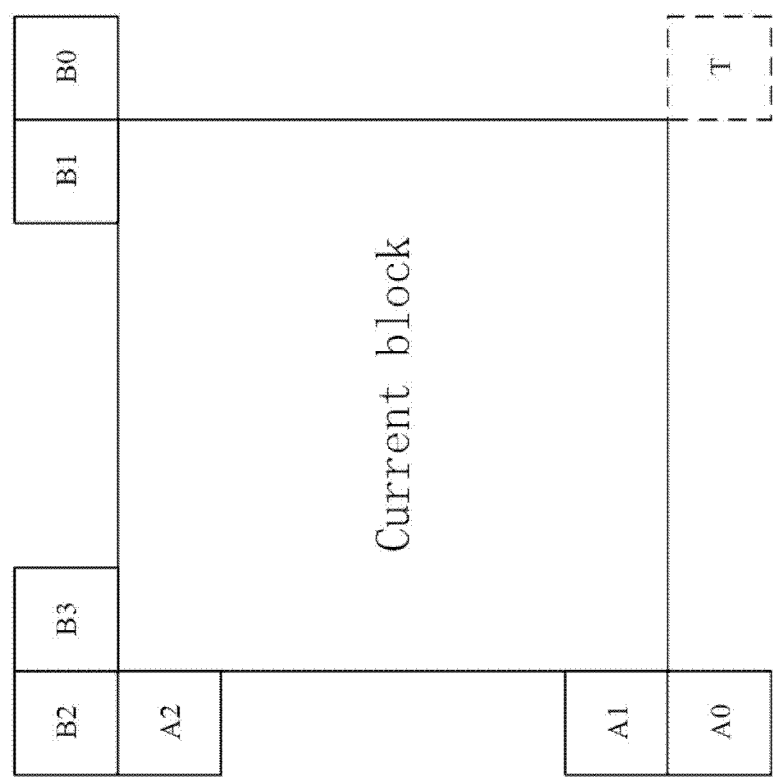


FIG. 19

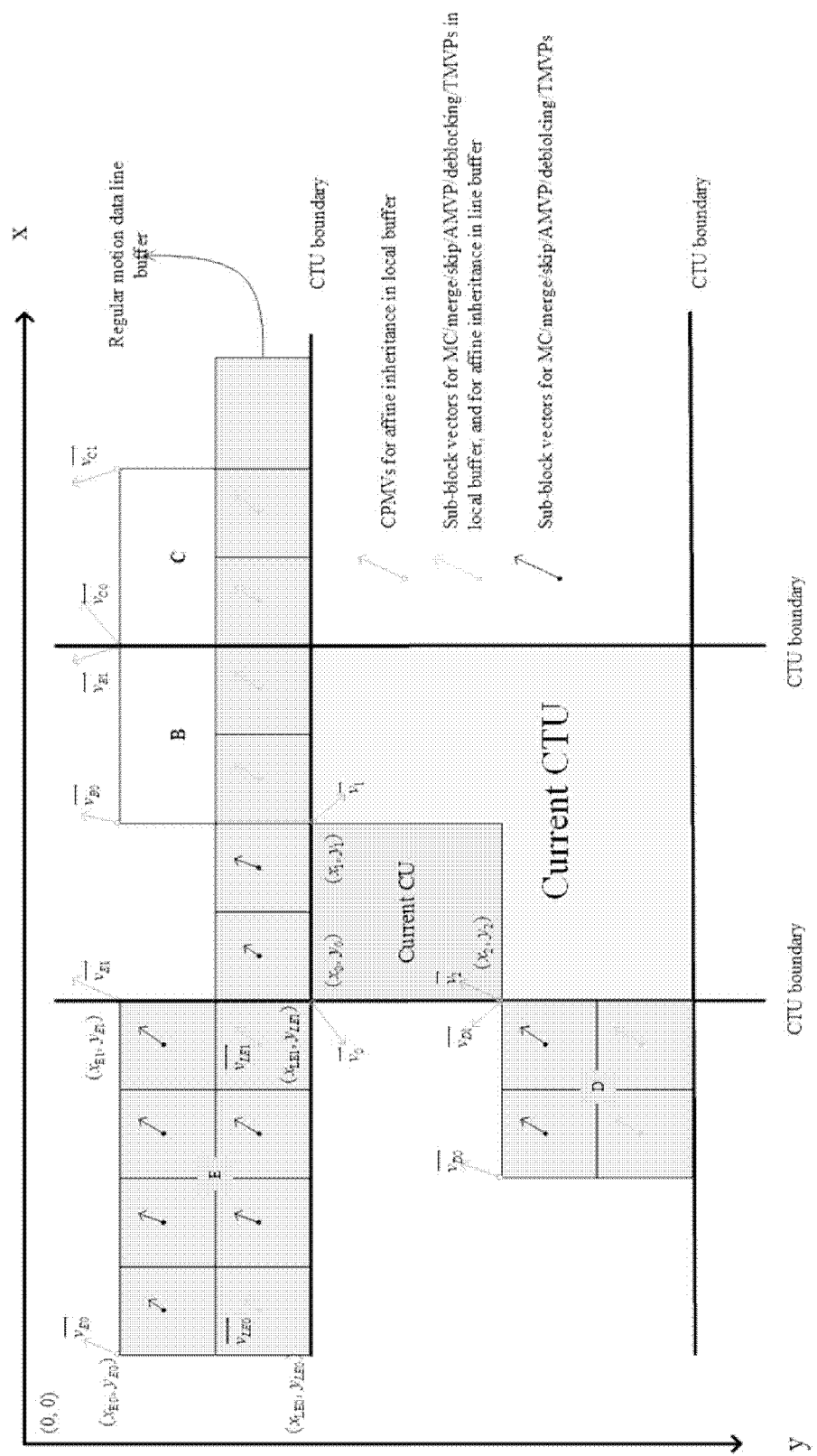


FIG. 20

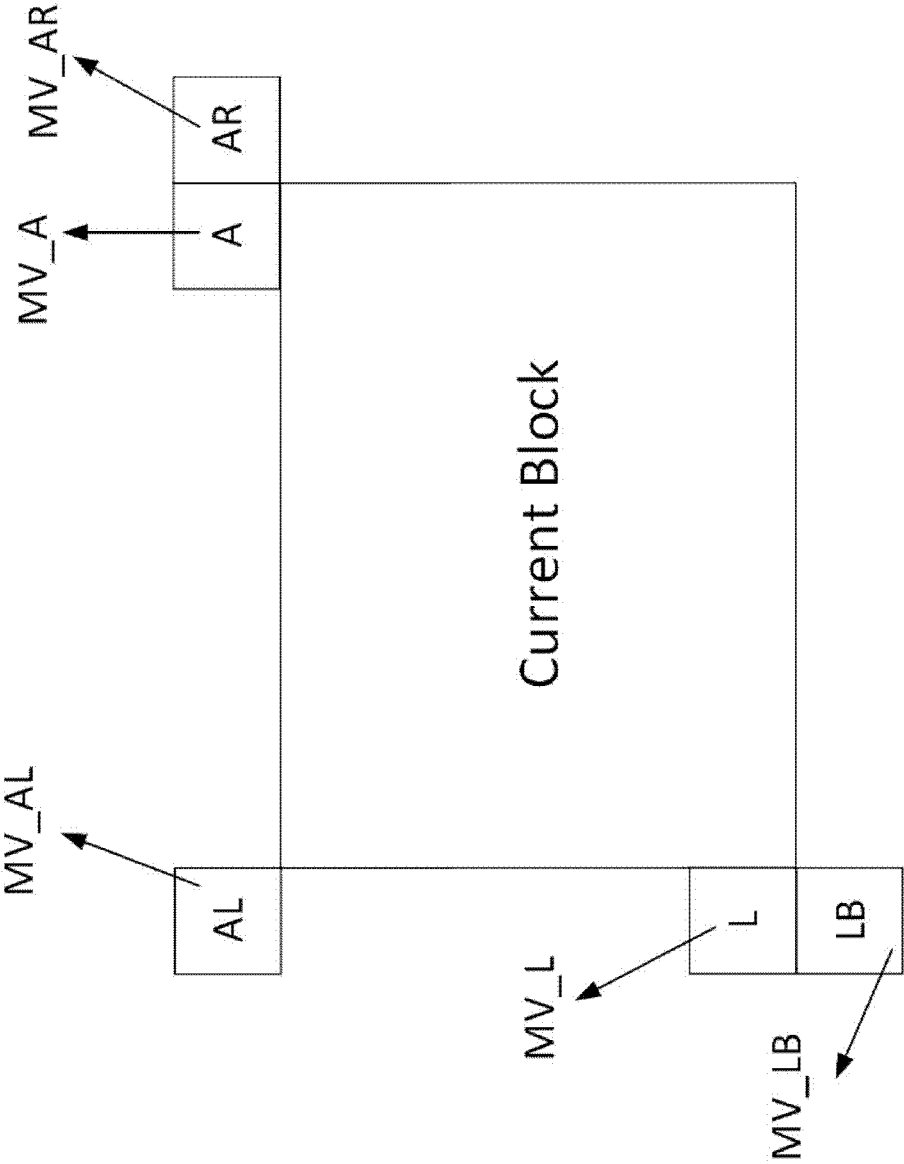


FIG. 21

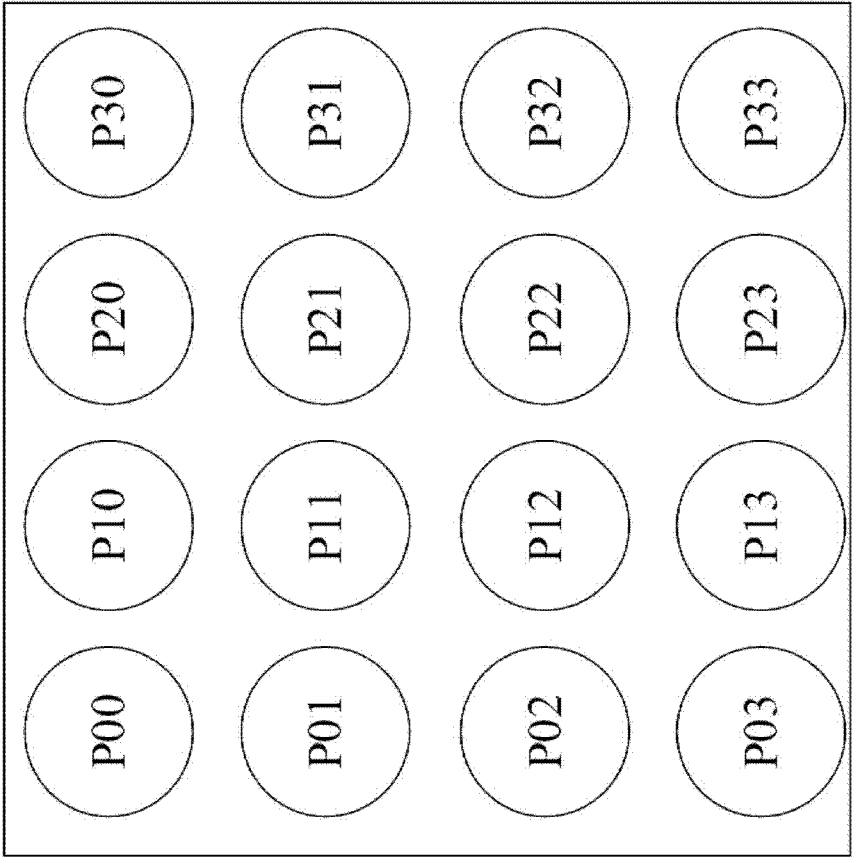


FIG. 22

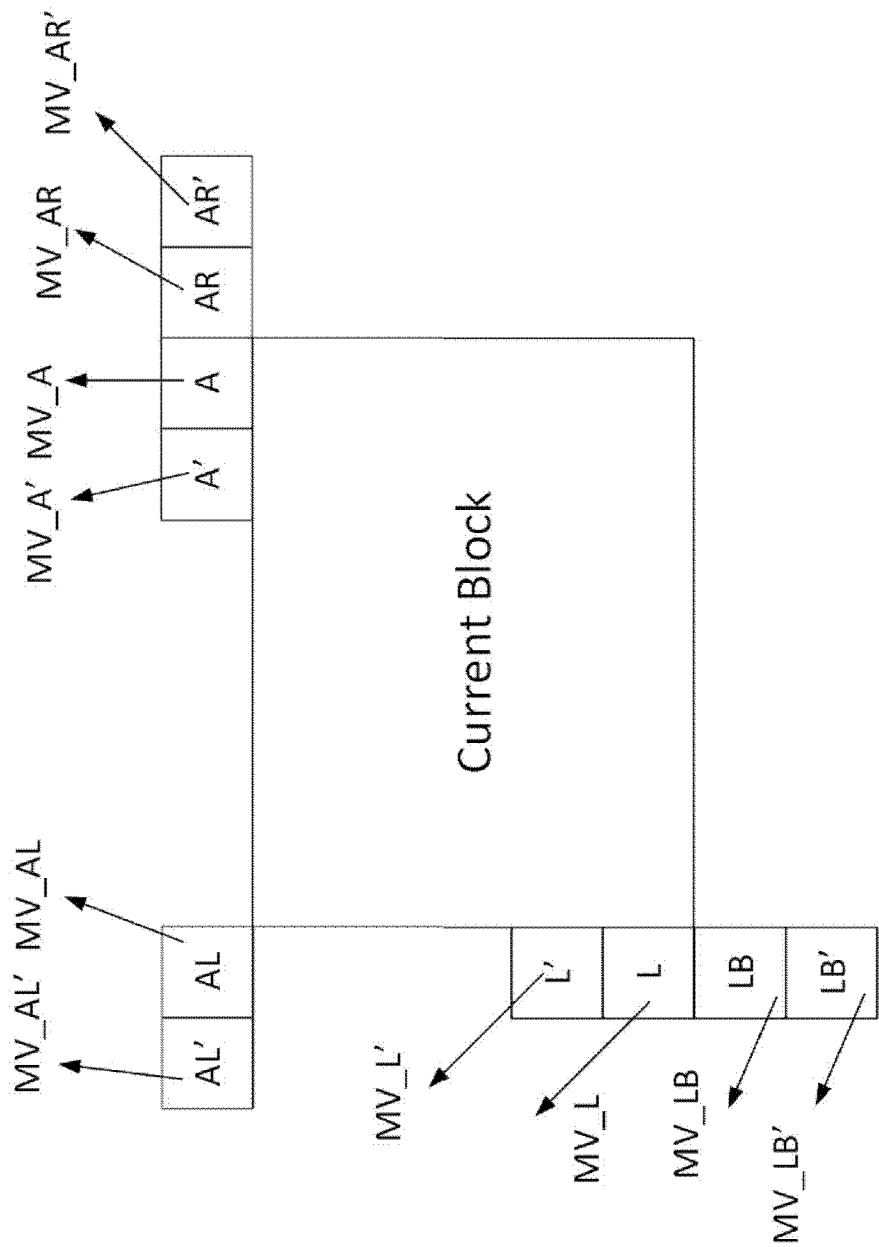


FIG. 23

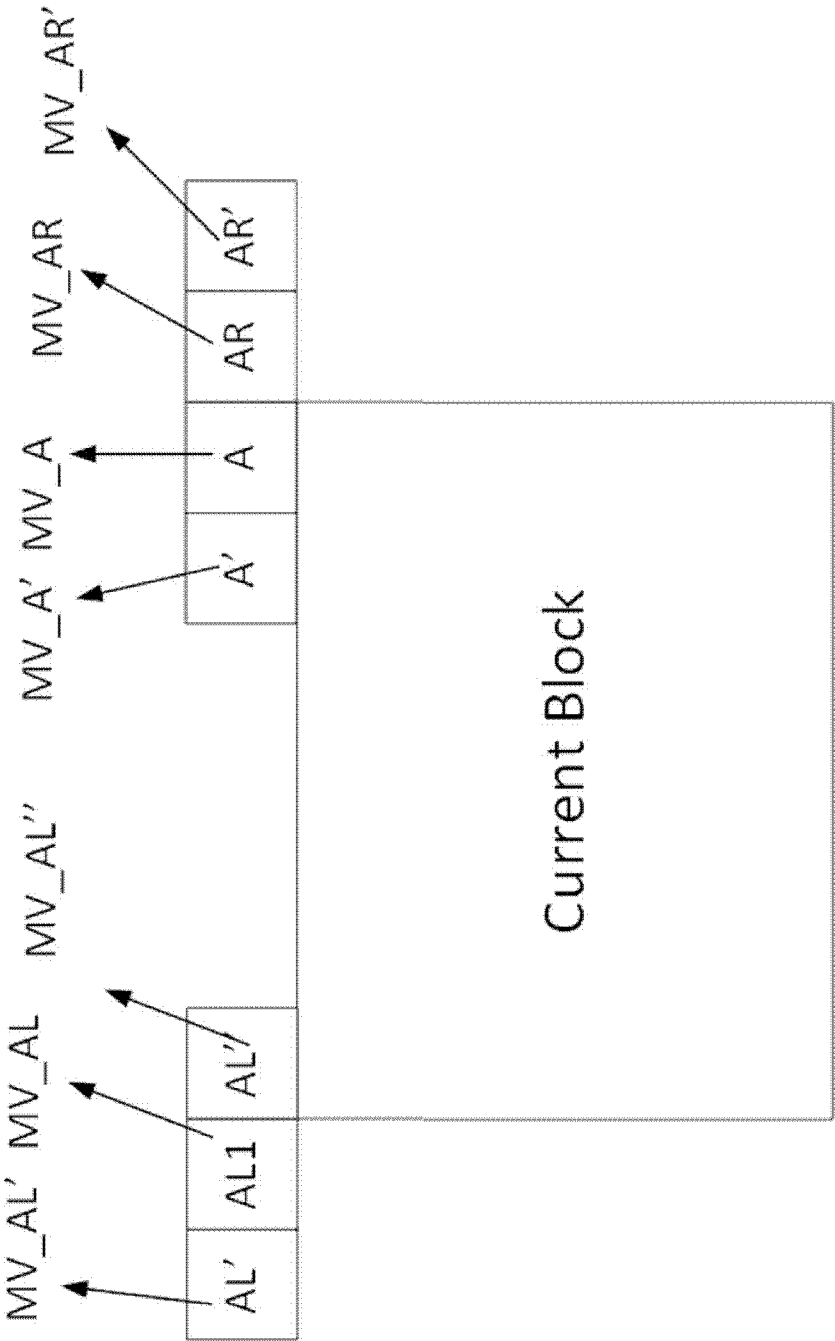


FIG. 24

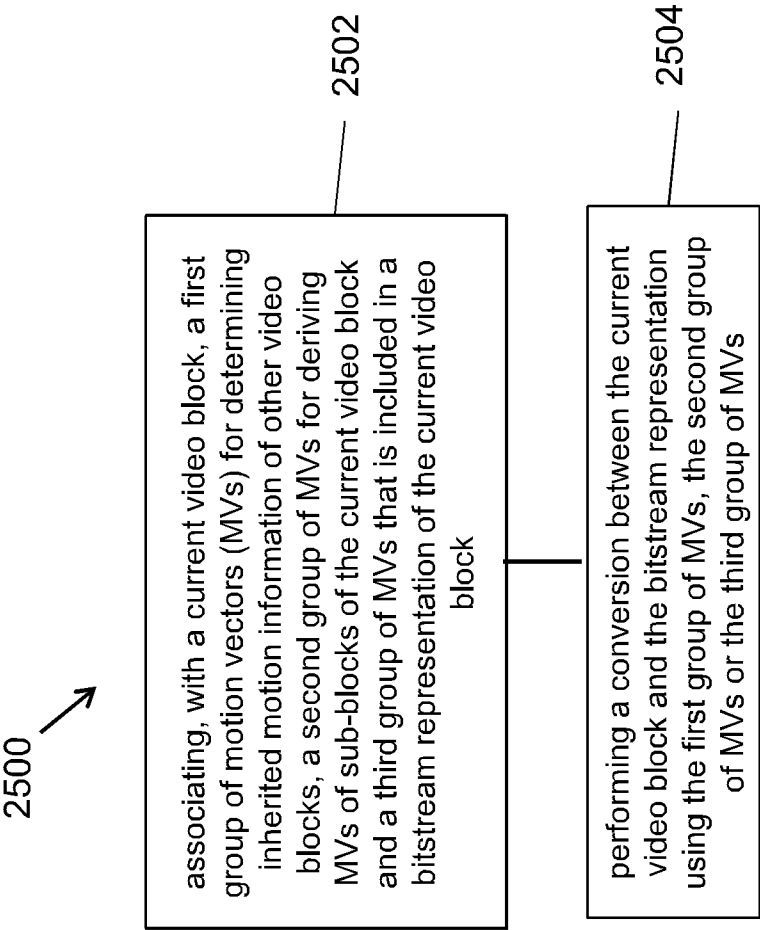


FIG. 25

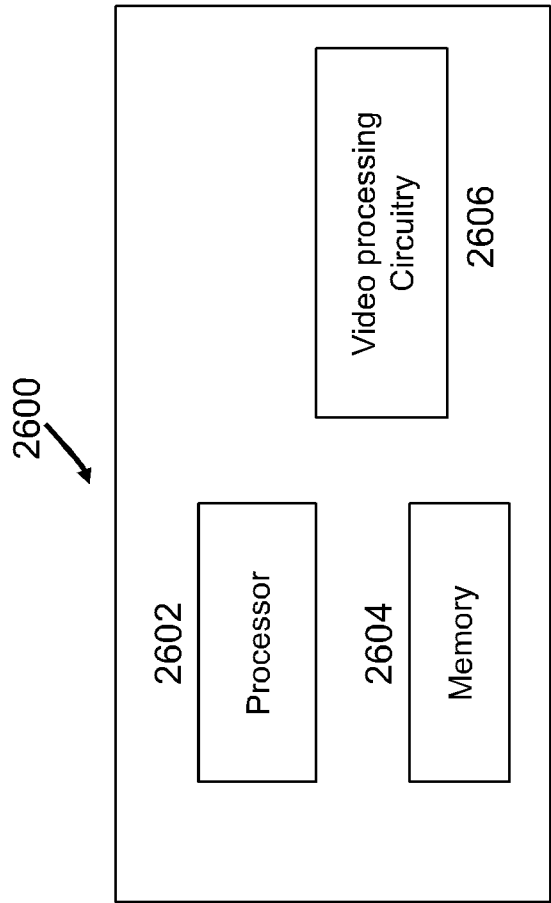


FIG. 26

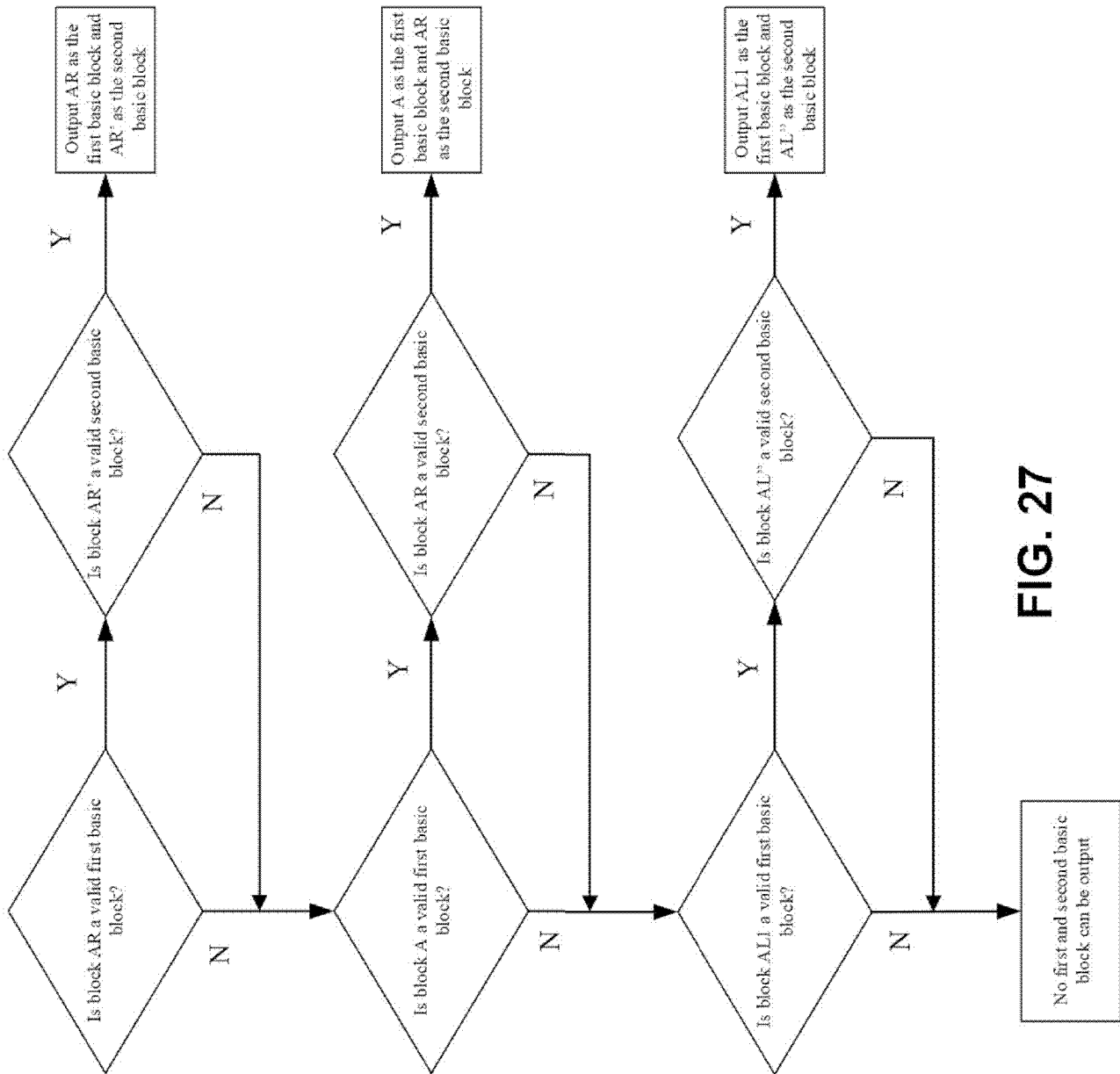


FIG. 27

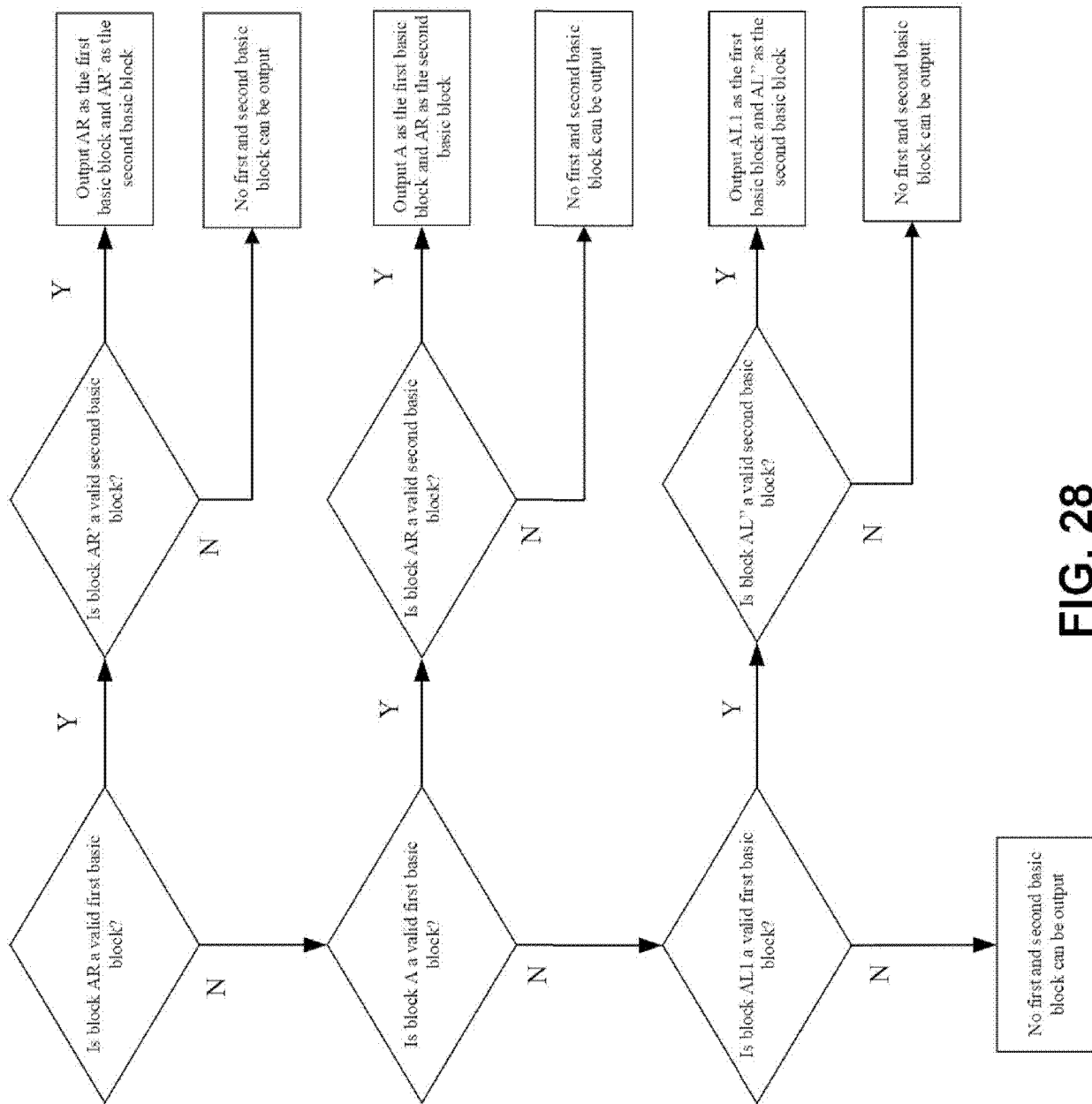


FIG. 28

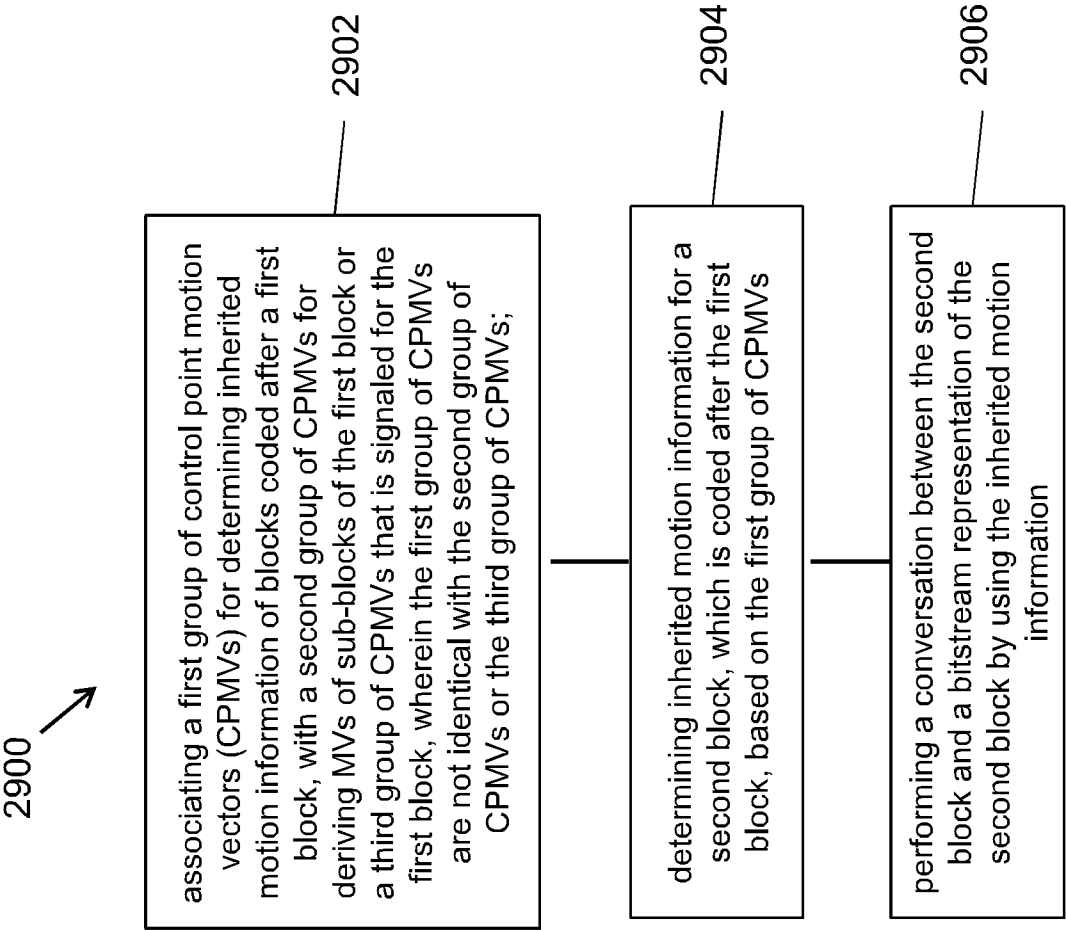


FIG. 29

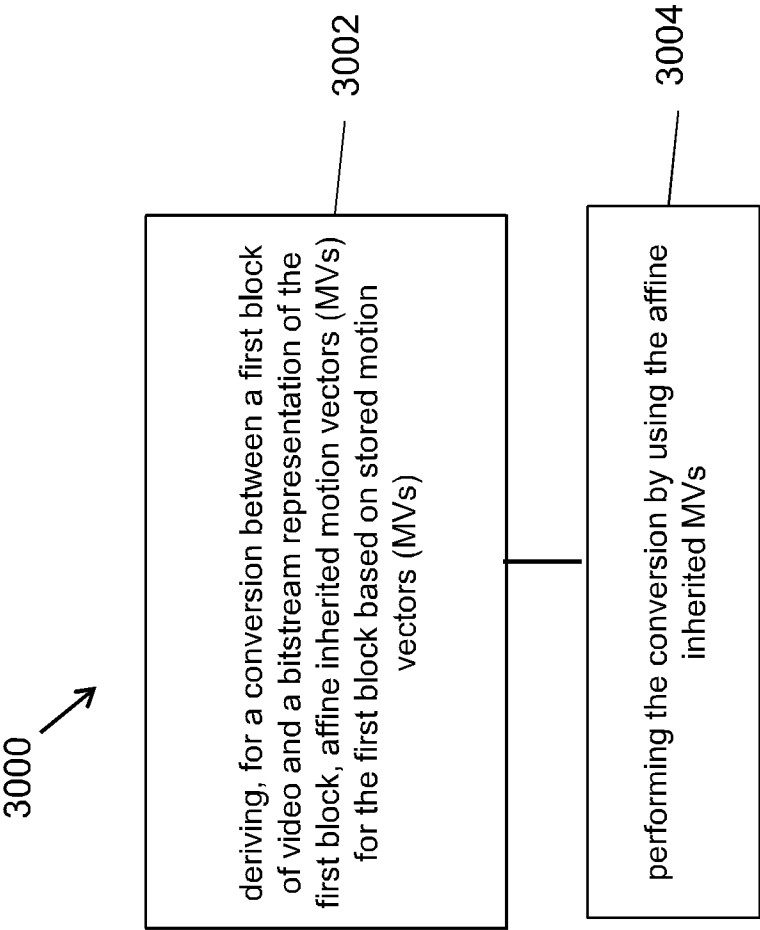


FIG. 30

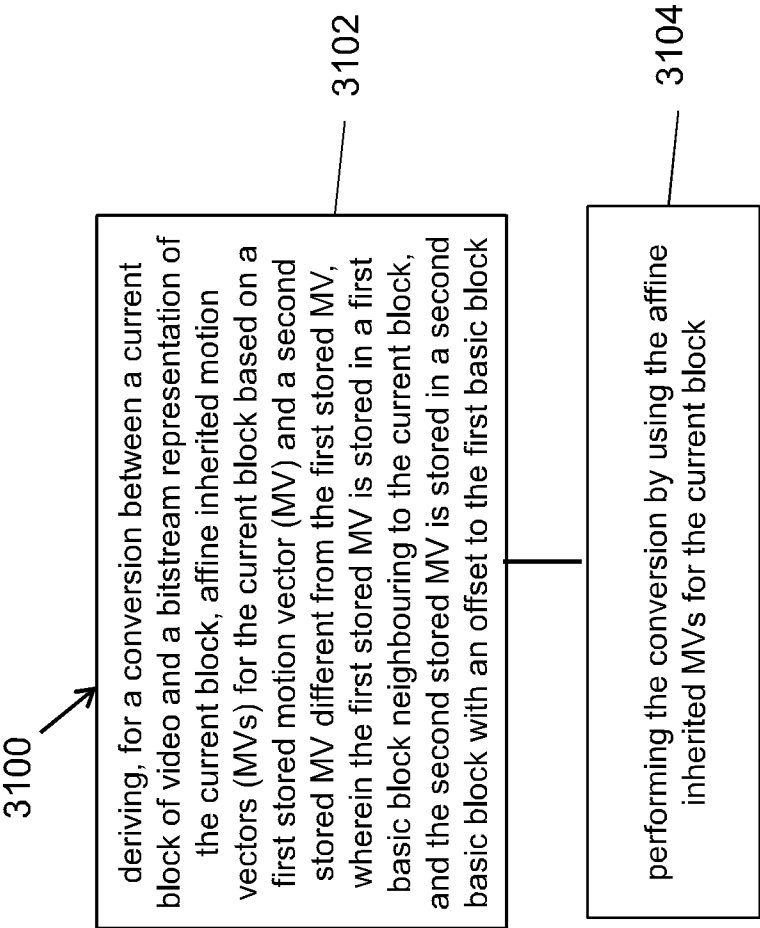
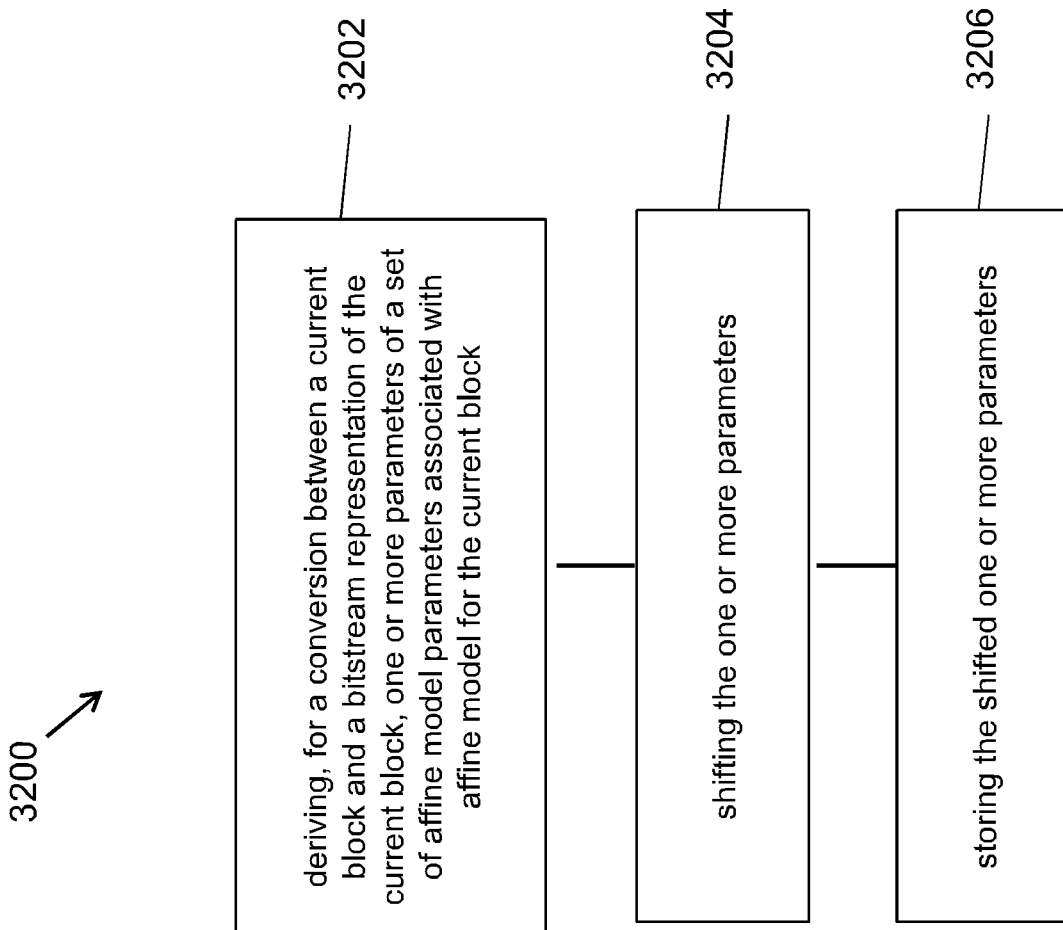


FIG. 31

**FIG. 32**

INTERNATIONAL SEARCH REPORT

International application No.

PCT/CN2019/124048

A. CLASSIFICATION OF SUBJECT MATTER

H04N 19/42(2014.01)i; H04N 19/52(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)

CNKI, CNPAT, WPI, EPODOC: encod+, decod+, compress+, decompress+, cod+, shift+, offset+, mov+, affine, model, parameter?, block, current, motion, vector, MV, horizontal, vertical, left, right

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
PX	WO 2019143841 A1 (QUALCOMM INC.) 25 July 2019 (2019-07-25) description, paragraphs [0062]-[0197], and figures 2-7B	1-42
X	Han Huang et al. "Control-Point Representation and Differential Coding Affine-Motion Compensation" <i>IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY</i> , Vol. 23, No. 10, 31 October 2013 (2013-10-31), the abstract, sections I-V, pages 1651-1659	1, 8-17, 23-42
X	CN 108781284 A (MEDIATEK INC.) 09 November 2018 (2018-11-09) description, paragraphs [0048]-[0093], and figures 4A-8	1, 8-17, 23-42
A	WO 2017148345 A1 (MEDIATEK INC.) 08 September 2017 (2017-09-08) the whole document	1-42



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

28 February 2020

Date of mailing of the international search report

09 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

QI, Ying

Telephone No. 86-10-53961718

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.

PCT/CN2019/124048

Patent document cited in search report			Publication date (day/month/year)	Patent family member(s)			Publication date (day/month/year)
WO	2019143841	A1	25 July 2019	US	2019230361	A1	25 July 2019
CN	108781284	A	09 November 2018	US	2019082191	A1	14 March 2019
				RU	2696551	C1	02 August 2019
				EP	3414900	A1	19 December 2018
				SG	11201806865Y	A	27 September 2018
				WO	2017157259	A1	21 September 2017
				TW	201739252	A	01 November 2017
WO	2017148345	A1	08 September 2017	EP	3414905	A1	19 December 2018
				CN	108605137	A	28 September 2018
				WO	2017147765	A1	08 September 2017
				TW	201803351	A	16 January 2018
				US	2019058896	A1	21 February 2019
				BR	112018067475	A2	02 January 2019