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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
30 April 2020 (30.04.2020)

WIPO | PCT

(10) International Publication Number
WO 2020/084470 A1

(51) International Patent Classification:
H04N 19/52 (2014.01) *H04N 19/176* (2014.01)
H04N 19/537 (2014.01) *H04N 19/54* (2014.01)
H04N 19/119 (2014.01) *H04N 19/186* (2014.01)
H04N 19/139 (2014.01)

(21) International Application Number:
PCT/IB2019/058991

(22) International Filing Date:
22 October 2019 (22.10.2019)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
PCT/CN2018/111176
22 October 2018 (22.10.2018) CN

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

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

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

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

(54) Title: STORAGE OF MOTION PARAMETERS WITH CLIPPING FOR AFFINE MODE

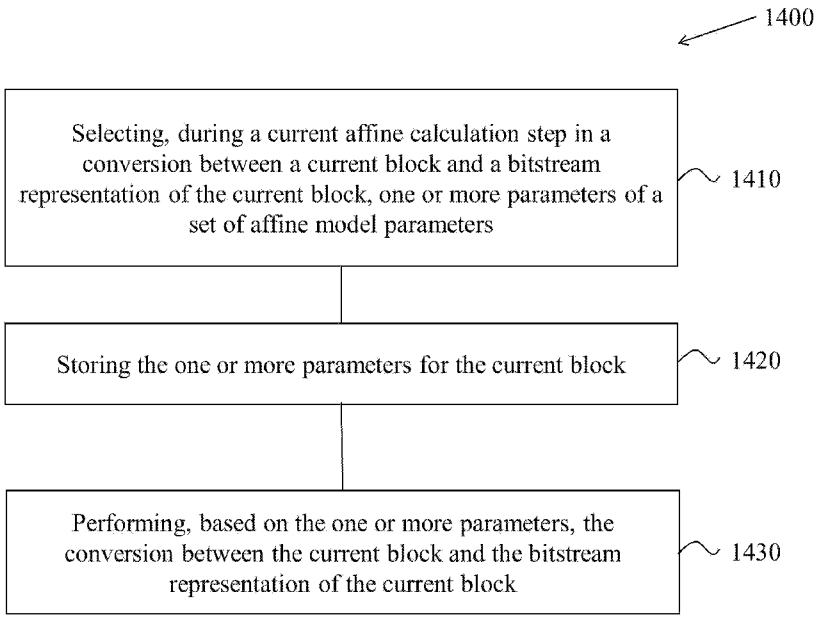

FIG. 14

(57) Abstract: The present disclosure relates to storage of motion information for affine mode. A video processing method is provided, comprising: selecting, during a current affine calculation step in 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; storing the one or more parameters for the current block; and performing, based on the one or more parameters, the conversion between the current block and the bitstream representation of the current block.

WO 2020/084470 A1

[Continued on next page]

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

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

Published:

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

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

STORAGE OF MOTION PARAMETERS WITH CLIPPING FOR AFFINE MODE

CROSS-REFERENCE TO RELATED APPLICATION

[0001] Under the applicable patent law and/or rules pursuant to the Paris Convention, this application is made to timely claim the priority to and benefits of International Patent Application No. PCT/CN2018/111176, filed on October 22, 2018. The entire disclosure of International Patent Application No. PCT/CN2018/111176 is incorporated by reference as part of the disclosure of this application.

TECHNICAL FIELD

[0002] This patent document is directed generally to image and video coding technologies.

BACKGROUND

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

SUMMARY

[0004] Devices, systems and methods related to digital video coding, and specifically, to the video and image coding and decoding in which motion information for affine mode is stored during video encoding or decoding are described. The described methods may be applied to both the existing video coding standards (e.g., High Efficiency Video Coding (HEVC)) and future video coding standards or video codecs.

[0005] In one representative aspect, a video processing method is provided, comprising: selecting, during a current affine calculation step in 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; storing the one or more parameters for the current block; and performing, based on the one or more parameters, the conversion between the current block and the bitstream representation of the current block.

[0006] In another representative aspect, a video processing method is provided, comprising:

acquiring, during a conversion between a current block and a bitstream representation of the current block, motion information of the current block, wherein the motion information of the current block is based on at least one affine model parameter of a neighboring block of the current block; and performing, based on the motion information, the conversion between the current block and the bitstream representation of the current block.

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

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

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

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

BRIEF DESCRIPTION OF THE DRAWINGS

[0011] FIG. 1 shows an example of sub-block based prediction.

[0012] FIGS. 2A and 2B show examples of the simplified 4-parameter affine model and the simplified 6-parameter affine model, respectively.

[0013] FIG. 3 shows an example of an affine motion vector field (MVF) per sub-block.

[0014] FIGS. 4A and 4B show example candidates for the AF_MERGE affine motion mode.

[0015] FIG. 5 shows an example of candidate positions for affine merge mode.

[0016] FIG. 6 shows an example of one coding unit (CU) with sub-blocks and neighboring blocks of the CU.

[0017] FIGS. 7A and 7B show examples of the motion vector field for merge/ skip/ AMVP/ TMVP affine inheritance and for motion compensation, respectively.

[0018] FIGS. 8A and 8B show examples of the motion vector field for affine inheritance and for merge/ skip/ AMVP/ TMVP motion compensation, respectively.

[0019] FIGS. 9A and 9B show examples of motion vector context storage for CPMV storage with 8-pixel granularity and MV storage with 4-pixel granularity, respectively.

[0020] FIG. 10 shows an example of a constrained sub-block motion vector for a CU coded

with an affine mode.

[0021] FIG. 11 shows an example of sixteen 4×4 blocks in a 16×16 region.

[0022] FIG. 12 shows a flowchart of an example method for video coding in accordance with the disclosed technology.

[0023] FIG. 13 is a block diagram of an example of a hardware platform for implementing a visual media decoding or a visual media encoding technique described in the present document.

[0024] FIG. 14 is a flowchart of an example method for video processing in accordance with the disclosed technology.

[0025] FIG. 15 is a flowchart of another example method for video processing in accordance with the disclosed technology.

DETAILED DESCRIPTION

[0026] Due to the increasing demand of higher resolution video, video coding methods and techniques are ubiquitous in modern technology. Video codecs typically include an electronic circuit or software that compresses or decompresses digital video, and are continually being improved to provide higher coding efficiency. A video codec converts uncompressed video to a compressed format or vice versa. There are complex relationships between the video quality, the amount of data used to represent the video (determined by the bit rate), the complexity of the encoding and decoding algorithms, sensitivity to data losses and errors, ease of editing, random access, and end-to-end delay (latency). The compressed format usually conforms to a standard video compression specification, e.g., the High Efficiency Video Coding (HEVC) standard (also known as H.265 or MPEG-H Part 2), the Versatile Video Coding standard to be finalized, or other current and/or future video coding standards.

[0027] Sub-block based prediction is first introduced into the video coding standard by the High Efficiency Video Coding (HEVC) standard. With sub-block based prediction, a block, such as a Coding Unit (CU) or a Prediction Unit (PU), is divided into several non-overlapped sub-blocks. Different sub-blocks may be assigned different motion information, such as reference index or motion vector (MV), and motion compensation (MC) is performed individually for each sub-block. FIG. 1 shows an example of sub-block based prediction.

[0028] Embodiments of the disclosed technology may be applied to existing video coding standards (e.g., HEVC, H.265) and future standards to improve runtime performance. Section

headings are used in the present document to improve readability of the description and do not in any way limit the discussion or the embodiments (and/or implementations) to the respective sections only.

1. Examples of the Joint Exploration Model (JEM)

[0029] In some embodiments, future video coding technologies are explored using a reference software known as the Joint Exploration Model (JEM). In JEM, sub-block based prediction is adopted in several coding tools, such as affine prediction, alternative temporal motion vector prediction (ATMVP), spatial-temporal motion vector prediction (STMVP), bi-directional optical flow (BIO), Frame-Rate Up Conversion (FRUC). Affine prediction has also been adopted into VVC.

1.1 Examples of affine prediction

[0030] In HEVC, only a 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 the VVC, a simplified affine transform motion compensation prediction is applied. As shown in FIGS. 2A and 2B, the affine motion field of the block is described by two (in the 4-parameter affine model that uses the variables *a, b, e* and *f*) or three (in the 6-parameter affine model that uses the variables *a, b, c, d, e* and *f*) control point motion vectors, respectively.

[0031] The motion vector field (MVF) of a block is described by the following equation with the 4-parameter affine model and 6-parameter affine model respectively:

[0032]
$$\begin{cases} mv^h(x,y) = ax - by + e = \dfrac{(mv_1^h - mv_0^h)}{w}x - \dfrac{(mv_1^v - mv_0^v)}{w}y + mv_0^h \\ mv^v(x,y) = bx + ay + f = \dfrac{(mv_1^v - mv_0^v)}{w}x + \dfrac{(mv_1^h - mv_0^h)}{w}y + mv_0^v \end{cases} \quad \text{Eq. (1)}$$

[0033]
$$\begin{cases} mv^h(x,y) = ax + cy + e = \dfrac{(mv_1^h - mv_0^h)}{w}x + \dfrac{(mv_2^h - mv_0^h)}{h}y + mv_0^h \\ mv^v(x,y) = bx + dy + f = \dfrac{(mv_1^v - mv_0^v)}{w}x + \dfrac{(mv_2^v - mv_0^v)}{h}y + mv_0^v \end{cases} \quad \text{Eq. (2)}$$

[0034] Herein, (mv^h_0, mv^h_0) is motion vector of the top-left corner control point (CP), and (mv^h_1, mv^h_1) is motion vector of the top-right corner control point and (mv^h_2, mv^h_2) is motion vector of the bottom-left corner control point, (x, y) represents the coordinate of a representative point relative to the top-left sample within current block. 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).

[0035] In a division-free design, Equations (1) and (2) are implemented as:

[0036]
$$\begin{cases} \text{iDMvHorX} = (mv_1^h - mv_0^h) << (S - \log 2(w)) \\ \text{iDMvHorY} = (mv_1^v - mv_0^v) << (S - \log 2(w)) \end{cases} \quad \text{Eq. (3)}$$

[0037] For the 4-parameter affine model shown in Equation (1):

[0038]
$$\begin{cases} \text{iDMvVerX} = \text{-iDMvHorY} \\ \text{iDMvVerY} = \text{iDMvHorX} \end{cases} \quad \text{Eq. (4)}$$

[0039] For the 6-parameter affine model shown in Equation (2):

[0040]
$$\begin{cases} \text{iDMvVerX} = (mv_2^h - mv_0^h) << (S - \log 2(h)) \\ \text{iDMvVerY} = (mv_2^v - mv_0^v) << (S - \log 2(h)) \end{cases} \quad \text{Eq. (5)}$$

[0041] And thus, the motion vectors may be derived as:

[0042]
$$\begin{cases} mv^h(x,y) = Normalize(\text{iDMvHorX} \square x + \text{iDMvVerX} \square y + (mv_0^h << S), S) \\ mv^v(x,y) = Normalize(\text{iDMvHorY} \square x + \text{iDMvVerY} \square y + (mv_0^v << S), S) \end{cases} \quad \text{Eq. (6)}$$

[0043]
$$Normalize(Z,S) = \begin{cases} (Z + Off) >> S & if\ Z \geq 0 \\ -((-Z + Off) >> S) & Otherwise \end{cases} \quad \text{Eq. (7)}$$
$$Off = 1 << (S-1)$$

[0044] Herein, *S* represents the calculation precision. e.g. in VVC, *S*=7. In VVC, the MV used in MC for a sub-block with the top-left sample at (xs, ys) is calculated by Equation (6) with x=xs+2 and y=ys+2.

[0045] To derive motion vector of each 4×4 sub-block, the motion vector of the center sample of each sub-block, as shown in FIG. 3, is calculated according to Equations (1) or (2), and rounded to 1/16 fraction accuracy. Then the motion compensation interpolation filters are applied to generate the prediction of each sub-block with derived motion vector.

[0046] Affine model can be inherited from spatial neighbouring affine-coded block such as left, above, above right, left bottom and above left neighbouring block as shown in FIG. 4A. For example, if the neighbour left bottom block A in FIG. 4A is coded in affine mode as denoted by

A0 in FIG. 4B, 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.

[0047] In some embodiments, sub-block (e.g. 4×4 block in VTM) LT stores mv0, RT stores mv1 if the current block is affine coded. If the current block is coded with the 6-parameter affine model, LB stores mv2; otherwise (with the 4-parameter affine model), LB stores mv2'. Other sub-blocks stores the MVs used for MC.

[0048] In some embodiments, when a CU is coded with affine merge mode, e.g., 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. 4A.

[0049] The derived CP MVs mv_0^C, mv_1^C and mv_2^C of current block can be used as CP MVs in the affine merge mode. Or they can be used as MVP for affine inter mode in VVC. It should be noted that for the merge mode, if the current block is coded with affine mode, after deriving CP MVs of current block, the current block may be further split into multiple sub-blocks and each block will derive its motion information based on the derived CP MVs of current block.

2. Exemplary embodiments of affine prediction

[0050] Different from VTM wherein only one affine spatial neighboring block may be used to derive affine motion for a block, it proposes to construct a separate list of affine candidates for the AF_MERGE mode.

[0051] *(1) Insert inherited affine candidates into candidate list*

[0052] In an example, inherited affine candidate means that the candidate is derived from the valid neighbor reconstructed block coded with affine mode.

[0053] As shown in FIG. 5, the scan order for the candidate block is A_1, B_1, B_0, A_0 and B_2. When a block is selected (e.g., A_1), the two-step procedure is applied:

[0054] (a) Firstly, use the three corner motion vectors of the CU covering the block to derive two/three control points of current block; and

[0055] (b) Based on the control points of current block to derive sub-block motion for each sub-block within current block.

[0056] *(2) Insert constructed affine candidates*

[0057] In some embodiments, if the number of candidates in affine merge candidate list is less than *MaxNumAffineCand*, constructed affine candidates are insert into the candidate list.

[0058] Constructed affine candidate means the candidate is constructed by combining the neighbor motion information of each control point.

[0059] The motion information for the control points is derived firstly from the specified spatial neighbors and temporal neighbor shown in FIG. 5. CPk (k=1, 2, 3, 4) represents the k-th control point. A_0, A_1, A_2, B_0, B_1, B_2 and B_3 are spatial positions for predicting CPk (k=1, 2, 3); T is temporal position for predicting CP4.

[0060] 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.

[0061] The motion information of each control point is obtained according to the following priority order:

[0062] ○ For CP1, the checking priority is $B_2 \rightarrow B_3 \rightarrow A_2$. B_2 is used if it is available. Otherwise, if B_2 is unavailable, B_3 is used. If both B_2 and B_3 are unavailable, A_2 is used. If all the three candidates are unavailable, the motion information of CP1 *cannot* be obtained.

[0063] ○ For CP2, the checking priority is B1→B0;

[0064] ○ For CP3, the checking priority is A1→A0;

[0065] ○ For CP4, T is used.

[0066] Secondly, the combinations of controls points are used to construct the motion model.

[0067] Motion vectors of three control points are needed to compute the transform parameters in 6-parameter affine model. 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}). For example, use CP1, CP2 and CP3 control points to construct 6-parameter affine motion model, denoted as Affine (CP1, CP2, CP3).

[0068] Motion vectors of two control points are needed to compute the transform parameters in 4-parameter affine model. The two control points can be selected from one of the following six combinations ({CP1, CP4}, {CP2, CP3}, {CP1, CP2}, {CP2, CP4}, {CP1, CP3}, {CP3, CP4}). For example, use the CP1 and CP2 control points to construct 4-parameter affine motion model, denoted as Affine (CP1, CP2).

[0069] The combinations of constructed affine candidates are inserted into to candidate list

as following order:

[0070] {CP1, CP2, CP3}, {CP1, CP2, CP4}, {CP1, CP3, CP4}, {CP2, CP3, CP4}, {CP1, CP2}, {CP1, CP3}, {CP2, CP3}, {CP1, CP4}, {CP2, CP4}, {CP3, CP4}

[0071] *(3) Insert zero motion vectors*

[0072] If the number of candidates in affine merge candidate list is less than MaxNumAffineCand, zero motion vectors are insert into the candidate list, until the list is full.

3. Examples of advanced temporal motion vector prediction (ATMVP)

[0073] In some existing implementations (e.g., 10th JVET meeting), advanced temporal motion vector prediction (ATMVP) was included in the benchmark set (BMS)-1.0 reference software, which derives multiple motion for sub-blocks of one coding unit (CU) based on the motion information of the collocated blocks from temporal neighboring pictures. Although it improves the efficiency of temporal motion vector prediction, the following complexity issues are identified for the existing ATMVP design:

[0074] ○ The collocated pictures of different ATMVP CUs may not be the same if multiple reference pictures are used. This means the motion fields of multiple reference pictures need to be fetched.

[0075] ○ The motion information of each ATMVP CU is always derived based on 4x4 units, resulting in multiple invocations of motion derivation and motion compensation for each 4x4 sub-block inside one ATMVP CU.

[0076] Some further simplifications on ATMVP can be adopted.

3.1 Examples of simplified collocated block derivation with one fixed collocated picture

[0077] In this exemplary method, one simplified design is proposed to use the same collocated picture as in HEVC, which is signaled at the slice header, as the collocated picture for ATMVP derivation. At the block level, if the reference picture of a neighboring block is different from this collocated picture, the MV of the block is scaled using the HEVC temporal MV scaling method, and the scaled MV is used in ATMVP.

[0078] Denote the motion vector used to fetch the motion field in the collocated picture R_{col} as MV_{col}. To minimize the impact due to MV scaling, the MV in the spatial candidate list used to derive MV_{col} is selected in the following way: if the reference picture of a candidate MV is the collocated picture, this MV is selected and used as MV_{col} without any scaling. Otherwise, the MV having a reference picture closest to the collocated picture is selected to derive MV_{col} with

scaling.

3.2 Examples of adaptive ATMVP sub-block size

[0079] In this exemplary method, it is proposed to support the slice-level adaptation of the sub-block size for the ATMVP motion derivation. Specifically, one default sub-block size that is used for the ATMVP motion derivation is signaled at sequence level. Additionally, one flag is signaled at slice-level to indicate if the default sub-block size is used for the current slice. If the flag is false, the corresponding ATMVP sub-block size is further signaled in the slice header for the slice.

4. Examples of spatial-temporal motion vector prediction (STMVP)

[0080] In the STMVP method, the motion vectors of the sub-CUs are derived recursively, following raster scan order. FIG. 6 shows an example of one CU with four sub-blocks and neighboring blocks. 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.

[0081] The motion derivation for sub-CU A starts by identifying its two spatial neighbours. The first neighbour is the N×N block above sub-CU A (block c). If this block c is not available or is intra coded the other N×N blocks above sub-CU A are checked (from left to right, starting at block c). The second neighbour is a block to the left of the sub-CU A (block b). If block b is not available or is intra coded other blocks to the left of sub-CU A are checked (from top to bottom, staring at block b). The motion information obtained from the neighbouring blocks for each list is scaled to the first reference frame for a given list. Next, temporal motion vector predictor (TMVP) of sub-block A is derived by following the same procedure of TMVP derivation as specified in HEVC. The motion information of the collocated block at location D is fetched and scaled accordingly. Finally, after retrieving and scaling the motion information, all available motion vectors (up to 3) are averaged separately for each reference list. The averaged motion vector is assigned as the motion vector of the current sub-CU.

5. Exemplary embodiments of affine merge candidate lists

5.1 Embodiments of affine merge candidate lists

[0082] In the affine merge mode, only the first available affine neighbour can be used to derive motion information of affine merge mode. A candidate list for affine merge mode is constructed by searching valid affine neighbours and combining the neighbor motion information of each control point.

[0083] The affine merge candidate list is constructed as following steps:

[0084] *(1) Insert inherited affine candidates*

[0085] Inherited affine candidate means that the candidate is derived from the affine motion model of its valid neighbor affine coded block. In the common base, as shown in FIG. 5, the scan order for the candidate positions is: A1, B1, B0, A0 and B2.

[0086] After a candidate is derived, full pruning process is performed to check whether same candidate has been inserted into the list. If a same candidate exists, the derived candidate is discarded.

[0087] *(2) Insert constructed affine candidates*

[0088] If the number of candidates in affine merge candidate list is less than MaxNumAffineCand (set to 5 in this example), 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.

[0089] The motion information for the control points is derived firstly from the specified spatial neighbors and temporal neighbor shown in FIG. 5. CPk (k=1, 2, 3, 4) represents the k-th control point. A0, A1, A2, B0, B1, B2 and B3 are spatial positions for predicting CPk (k=1, 2, 3); T is temporal position for predicting CP4.

[0090] 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.

[0091] The motion information of each control point is obtained according to the following priority order:

[0092] ○ For CP1, the checking priority is $B_2 \rightarrow B_3 \rightarrow A_2$. B_2 is used if it is available. Otherwise, if B_2 is unavailable, B_3 is used. If both B_2 and B_3 are unavailable, A_2 is used. If all the three candidates are unavailable, the motion information of CP1 *cannot* be obtained.

[0093] ○ For CP2, the checking priority is B1→B0;

[0094] ○ For CP3, the checking priority is A1→A0;

[0095] ○ For CP4, T is used.

[0096] Secondly, the combinations of controls points are used to construct the motion model.

[0097] 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.

[0098] Motion information of two control points are needed to construct a 4-parameter affine candidate. The two control points can be selected from one of the following six combinations ({CP1, CP4}, {CP2, CP3}, {CP1, CP2}, {CP2, CP4}, {CP1, CP3}, {CP3, CP4}). Combinations {CP1, CP4}, {CP2, CP3}, {CP2, CP4}, {CP1, CP3}, {CP3, CP4} will be converted to a 4-parameter motion model represented by top-left and top-right control points.

[0099] The combinations of constructed affine candidates are inserted into to candidate list as following order:

[00100] {CP1, CP2, CP3}, {CP1, CP2, CP4}, {CP1, CP3, CP4}, {CP2, CP3, CP4}, {CP1, CP2}, {CP1, CP3}, {CP2, CP3}, {CP1, CP4}, {CP2, CP4}, {CP3, CP4}

[00101] For reference list X (X being 0 or 1) of a combination, the reference index with highest usage ratio in the control points is selected as the reference index of list X, and motion vectors point to difference reference picture will be scaled.

[00102] After a candidate is derived, full pruning process is performed to check whether same candidate has been inserted into the list. If a same candidate exists, the derived candidate is discarded.

[00103] *(3) Padding with zero motion vectors*

[00104] 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.

5.2 Embodiments of affine merge mode

[00105] Simplifications for the affine merge mode are proposed as follows:

[00106] (1) The pruning process for inherited affine candidates is simplified by comparing the coding units covering the neighboring positions, instead of comparing the affine candidates. Up to 2 inherited affine candidates are inserted into affine merge list. The pruning process for constructed affine candidates is totally removed.

[00107] (2) The MV scaling operation in constructed affine candidate is removed. If the reference indices of control points are different, the constructed motion model is discarded.

[00108] (3) The number of constructed affine candidates is reduced from 10 to 6.

[00109] (4) It is also proposed that other merge candidates with sub-block prediction such as ATMVP is also put into the affine merge candidate list. In that case, the affine merge

candidate list may be renamed with some other names such as sub-block merge candidate list.

6. Examples of pairwise average candidates

[00110] Pairwise average candidates are generated by averaging predefined pairs of candidates in the current merge candidate list, and the predefined pairs are defined as {(0, 1), (0, 2), (1, 2), (0, 3), (1, 3), (2, 3)}, where the numbers denote the merge indices to the merge candidate list. The averaged motion vectors are calculated separately for each reference list. If both motion vectors are available in one list, these two motion vectors are averaged even when they point to different reference pictures; if only one motion vector is available, use the one directly; if no motion vector is available, keep this list invalid. The pairwise average candidates replaces the combined candidates in HEVC standard.

7. Examples of control point motion vector (CPMV) offset

[00111] New Affine merge candidates are generated based on the CPMVs offsets of the first Affine merge candidate. If the first Affine merge candidate enables 4-parameter Affine model, then 2 CPMVs for each new Affine merge candidate are derived by offsetting 2 CPMVs of the first Affine merge candidate; Otherwise (6-parameter Affine model enabled), then 3 CPMVs for each new Affine merge candidate are derived by offsetting 3 CPMVs of the first Affine merge candidate. In Uni-prediction, the CPMV offsets are applied to the CPMVs of the first candidate. In Bi-prediction with List 0 and List 1 on the same direction, the CPMV offsets are applied to the first candidate as follows:

[00112] $MV_{new(L0),i} = MV_{old(L0)} + MV_{offset(i)}$

[00113] $MV_{new(L1),i} = MV_{old(L1)} + MV_{offset(i)}$

[00114] In Bi-prediction with List 0 and List 1 on the opposite direction, the CPMV offsets are applied to the first candidate as follows:

[00115] $MV_{new(L0),i} = MV_{old(L0)} + MV_{offset(i)}$

[00116] $MV_{new(L1),i} = MV_{old(L1)} - MV_{offset(i)}$

[00117] In this embodiment, various offset directions with various offset magnitudes are used to generate new Affine merge candidates. Two implementations were tested:

[00118] (1) 16 new Affine merge candidates with 8 different offset directions with 2 different offset magnitudes are generated as shown in the following offsets set:

[00119] Offset set = { (4, 0), (0, 4), (-4, 0), (0, -4), (-4, -4), (4, -4), (4, 4), (-4, 4), (8, 0), (0, 8), (-8, 0), (0, -8), (-8, -8), (8, -8), (8, 8), (-8, 8) }.

[00120] The Affine merge list is increased to 20 for this design. The number of potential Affine merge candidates is 31 in total.

[00121] (2) 4 new Affine merge candidates with 4 different offset directions with 1 offset magnitude are generated as shown in the following offsets set:

[00122] Offset set = { (4, 0), (0, 4), (-4, 0), (0, -4) }.

[00123] The Affine merge list is kept to 5. Four temporal constructed Affine merge candidates are removed to keep the number of potential Affine merge candidates unchanged, i.e., 15 in total. Suppose the coordinates of CPMV1, CPMV2, CPMV3 and CPMV4 are (0, 0), (W, 0), (H, 0) and (W, H). Note that CPMV4 is derived from the temporal MV as shown in FIG. 5. The removed candidates are the following four temporal-related constructed Affine merge candidates: {CP2, CP3, CP4}, {CP1, CP4}, {CP2, CP4}, {CP3, CP4}.

8. Examples of the line-buffer issue

[00124] In the current design of the affine mode, a mixed motion vector field of the affine control point vectors (CPMVs) and sub-block motion vectors is used for the affine motion data inheritance (i.e. for affine merge and affine AMVP list derivation), for the merge/skip and AMVP list derivation (i.e. serve as spatial neighboring candidates), and for storage of temporal motion vectors (TMVPs) for use in future pictures. A separate sub-block motion vector field (computed on-the-fly) is used for the motion compensation of PUs coded in affine mode.

[00125] FIGS. 7A and 7B show an example, in which a CTU is partitioned in two PUs, and both of them are coded in affine mode. As shown in FIG. 7A, in the current design the CPMVs are stored in the top-left, top-right and bottom-left (if a PU uses 6-parameter affine motion models) sub-block locations of the sub-block motion vector field (overwriting the computed sub-block vectors for those locations). This creates a mixed set of CPMVs and sub-block motion vectors that is used for the merge/skip/AMVP/TMVPs and affine inheritance. For motion compensation, a separate sub-block motion vector field (shown in FIG. 7B) is generated again as only the sub-block vectors (not control point vectors) are used in the motion compensation of affine mode.

[00126] For actual decoder implementations, the sub-block motion vector field (shown in FIG. 8B for example) needs to be generated in advance, and pushed to the motion compensation engine to set up pre-fetch of reference blocks for motion compensation. Before the sub-block motion vector field is consumed by the motion compensation, it cannot be overwritten by e.g.

CPMVs.

[00127] To minimize the memory buffer size, one possible solution to store the CPMVs and the sub-block vectors separately, in which the CPMVs are stored in one buffer, and the sub-block vectors for the MC are stored in another, so that the sub-block vector field used for the MC won't get overwritten before it is consumed by the MC. For the merge/skip and AMVP list derivation and the storage of the TMVPs, the decoder would need to switch back and forth between those two motion vector buffers to fetch either the CPMVs or sub-block vectors from neighboring PUs as the spatial MV candidates or candidate vectors for TMVP storage. Also, more storage would be needed for the CPMVs, because the non-adjacent CPMVs, which are no longer required by the merge/skip/AMVP list derivation of the current PU, cannot be disposed before they are compressed together with other sub-block motion vectors in the CTU and written out as temporal motion vectors for use in future pictures.

[00128] As discussed above, the CPMVs would need to be stored separately anyway in actual implementations for minimizing the memory buffer size, it makes less sense to mix the CPMVs and sub-block vectors in the merge/skip and AMVP list derivation process and in the storage of TMVPs, as doing so won't reduce the memory footprint. It would be more straightforward and consistent to use the sub-block vectors as spatial candidates for merge/skip and AMVP list derivation and for TMVPs, and leave the CPMVs for the use of the affine motion data inheritance only.

[00129] FIGS. 8A and 8B show an example for the proposed clean-up (e.g., clean-up method 1), in which the CPMVs (e.g., FIG. 8A) are stored separately and used for the affine inheritance only. The sub-block vector field (e.g., FIG. 8B) is used not only for the motion compensation, but also for the merge/skip/AMVP list derivation and for the storage of TMVPs.

[00130] For 128x128 CTU size, the buffers for storing all CPMVs and 4x4 based sub-block motion vectors inside the CTU are about 6,144 bytes and 8,192 bytes (just counting motion vectors, not other parameters such as PU sizes and reference indices for the sake of explanation), respectively.

[00131] Because the CPMVs are only used for the affine motion data in heritance and the non-adjacent CPMVs can be disposed, in actual implementations the CPMVs do not need to be stored for the entire CTU. In order to minimize the storage requirements, a decoder can store only a top-row and left-column of vector context, rather than a full-CTU, or a full-picture worth

of vector context, and as shown in FIGS. 9A and 9B. These figures illustrate the vector context state after decoding CU #10. The CPMV context is shown in FIG. 9A with 8-pixel granularity, and the standard sub-block MV context is shown in FIG. 9B with 4-pixel granularity. As each CU is decoded, the corresponding left and top context are updated, shifting the blue and green lines further down and to the right, working the way across the CTU and eventually across the picture.

[00132] In this type of memory-optimized implementation, the CPMVs would be stored for each 8-pixel segment. There would be 16 such segments for the top-row, 16 for the left-column, and 16 segments for storing top-left context. This CPMV storage would require roughly 24 bytes per segment (3 CPMVs – 8 bytes each), so 1,152 bytes (24 bytes * 48 segments) for the top/left CPMV context within the CTU. Additional such storage per 8-pixel segment would be required across the top of FIG. 9A (top line buffer) if affine inheritance was allowed to extend outside the current CTU.

[00133] The sub-block MVs would be stored for each 4-pixel segment. There would be 32 such segments for the top-row, 32 for the left-column, and 32 segments for storing top-left context. This MV storage would require roughly 8 bytes per segment (1 bi-directional MV – 8 bytes each), so 768 bytes (8 bytes * 96 segments) for the top/left sub-block MV context within the CTU.

[00134] This kind of memory-optimized implementation effectively cuts the memory footprint for the CPMVs from 6,144 bytes to 1,152 bytes for a 128×128 CTU. For the 64×64 block based decoder pipeline, the proposed cleanup avoids the need of buffering the CPMVs for the 64×64 blocks, which saves about 1,536 bytes memory (e.g., 6144/4 bytes). In addition, this implementation supports the motion vector reconstruction on a small block basis (e.g., 64×64 instead of 128×128), which further reduces the memory footprint for storage of the sub-block motion vectors from e.g. 8,192 bytes to 2,816 bytes (e.g., 8192/4 + 768).

9. Example of the bandwidth problem of affine motion compensation

[00135] In some embodiments, since the current block is divided into 4×4 sub-blocks for luma component and 2×2 sub-blocks for the two chroma components to do the motion compensation, the total bandwidth requirement is much higher than non sub-block inter-prediction.

10. Exemplary embodiments of sub-block size

[00136] In some embodiments, a 4x4 block is used as the sub-block size for a uni-directional

affine coded CU while 8x4/4x8 block is used as the sub-block size for a bi-directional affine coded CU.

11. Exemplary embodiments of affine mode

[00137] For affine mode, sub-block motion vectors of an affine CU are constrained to be within a pre-defined motion vector field. Assume that the motion vectors of 1st (top left) sub-block is (v_{0x},v_{0y}) and the second sub-block is (v_{ix}, v_{iy}), values of v_{ix} and v_{iy} exhibit the following constraints:

[00138] $v_{ix} \in [v_{0x}-H, v_{0x}+H]$

[00139] $v_{iy} \in [v_{0y}-V, v_{0y}+V]$

[00140] If the motion vector of any sub-block exceeds the pre-defined motion vector field, the motion vector is clipped. An illustration of the idea of constrained sub-block motion vector is shown in FIG. 10.

[00141] In some embodiments, and assuming memory is retrieved per CU instead of per sub-block, values H and V are chosen so that worst case memory bandwidth of affine CU will not exceed that of normal inter MC of a 8×8 bi-prediction block. Note that values of H and V are adaptive to CU size and uni-prediction or bi-prediction.

[00142] 12.Exemplary embodiments of affine mode

[00143] In some embodiments, and to reduce the memory bandwidth requirement in affine prediction, each 8x8 block within the block is regarded as the basic unit. The MVs of all four 4x4 sub-blocks inside the 8x8 block are constrained such that the max difference between integer parts of the four 4x4 sub-block MVs is no more than 1 pixel. So that the bandwidth is (8+7+1)*(8+7+1)/(8*8)=4 sample/pixel.

[00144] For example, after the MVs of all sub-blocks inside the current block are calculated with affine model, the MV of the sub-blocks containing the control points are firstly replaced with the corresponding control point MV. This means that, the MV of the top-left, top-right and bottom-left sub-blocks are replaced by the top-left, top-right and bottom-left control points MV, respectively. Then, for each 8×8 block within the current block, the MVs of all four 4×4 sub-blocks are clipped to guarantee the max difference between integer parts of the four MVs no more than 1 pixel. Here it should be noted that the sub-blocks containing the control points (top-left, top-right and bottom-left sub-blocks) use the corresponding control point MV to involve in the MV clipping process. During the clipping process, the MV of the top-right control point is

kept un-changed.

[00145] The clipping process applied to each 8×8 block is described as follows:

[00146] (1) The minimal and maximal values for the MV components, MVminx, MVminy, MVmaxx, MVmaxy are determined first for each 8x8 block as follows:

[00147] (a) Get the minimal MV component among the four 4x4 sub-block MVs

[00148] MVminx = min(MVx0, MVx1, MVx2, MVx3)

[00149] MVminy = min(MVy0, MVy1, MVy2, MVy3)

[00150] (b) Use the integer part of MVminx and MVminy as the minimal MV component, e.g.,

[00151] MVminx = MVminx >> MV_precision << MV_precision

[00152] MVminy = MVminy >> MV_precision << MV_precision

[00153] (c) The maximal MV component is calculated as follows:

[00154] MVmaxx = MVminx + (2 << MV_precision) – 1

[00155] MVmaxy = MVminy + (2 << MV_precision) – 1

[00156] (d) if the top-right control point is in current 8x8 block

[00157] if (MV1x > MVmaxx)

[00158] MVminx = (MV1x >> MV_precision << MV_precision) – (1 << MV_precision)

[00159] MVmaxx = MVminx + (2 << MV_precision) – 1

[00160] if (MV1y > MVmaxy)

[00161] MVminy = (MV1y >> MV_precision << MV_precision) – (1 << MV_precision)

[00162] MVmaxy = MVminy + (2 << MV_precision) – 1

[00163] (2) The MV components of each 4x4 block inside this 8×8 block are clipped as follows:

[00164] MVxi = max(MVminx, min(MVmaxx, MVxi))

[00165] MVyi = max(MVminy, min(MVmaxy, MVyi))

[00166] Herein, (MVxi, MVyi) is the MV of ith sub-block within one 8×8 block, where i is 0, 1, 2, 3; (MV1x, MV1y) is the MV of the top-right control point; MV_precision is equal to 4 corresponding to 1/16 motion vector fraction accuracy. Since the difference between integer parts of MVminx and MVmaxx (MVminy and MVmaxy) is 1 pixel, the max difference between

integer parts of the four 4×4 sub-block MVs is no more than 1 pixel.

13. Exemplary embodiments of affine mode

[00167] In some embodiments, there may be restrictions to the affine mode for the worst-case bandwidth reduction. To ensure that the worst-case bandwidth of the affine block is not worse than an INTER_4x8/INTER_8x4 block or even an INTER_9x9 block, the motion vector differences between affine control points are used to decide whether the subblock size of the affine block is 4x4 or 8x8.

13.1 General affine restriction for worst-case bandwidth reduction

[00168] The memory bandwidth reduction for the affine mode is controlled by restricting the motion vector difference between the affine control points (also named as the control points difference). In general, if the control points differences satisfy the restriction below, the affine motion is using 4×4 sub-blocks (namely 4x4 affine mode). Otherwise, it is using 8×8 sub-blocks (8×8 affine mode). The restrictions for the 6-parameters and 4-parameters model are given as follows.

[00169] To derive the constraints for different block sizes (*w*x*h*), the motion vector differences of the control points are normalized as:

[00170] $$Norm(v_{1x} - v_{0x}) = (v_{1x} - v_{0x}) * 128/w$$

[00171] $$Norm(v_{1y} - v_{0y}) = (v_{1y} - v_{0y}) * 128/w$$

[00172] $$Norm(v_{2x} - v_{0x}) = (v_{2x} - v_{0x}) * 128/h$$

[00173] $$Norm(v_{2x} - v_{0x}) = (v_{2x} - v_{0x}) * 128/h \quad \text{Eq. (8)}$$

[00174] In the 4-parameters affine model, $(v_{2x} - v_{0x})$ and $(v_{2y} - v_{0y})$ are set as the follows:

[00175] $$(v_{2x} - v_{0x}) = -(v_{1y} - v_{0y})$$

[00176] $$(v_{2y} - v_{0y}) = -(v_{1x} - v_{0x}) \quad \text{Eq. (9)}$$

[00177] Hence, the *Norms* of $(v_{2x} - v_{0x})$ and $(v_{2y} - v_{0y})$ are given as:

[00178] $$Norm(v_{2x} - v_{0x}) = -Norm(v_{1y} - v_{0y})$$

[00179] $$Norm(v_{2y} - v_{0y}) = Norm(v_{1x} - v_{0x}) \quad \text{Eq. (10)}$$

13.2 Restriction to ensure worst-case bandwidth for INTER_4x8 or INTER_8x4

$$\begin{aligned}&|Norm(v_{1x}-v_{0x})+Norm(v_{2x}-v_{0x})+128|+\\&|Norm(v_{1y}-v_{0y})+Norm(v_{2y}-v_{0y})+128|+\\&|Norm(v_{1x}-v_{0x})-Norm(v_{2x}-v_{0x})|+\\&|Norm(v_{1y}-v_{0y})-Norm(v_{2y}-v_{0y})|\\&<128*3.25\end{aligned}$$ Eq. (11)

[00180] Herein, the left-hand side of the above equation represents the shrink or span level of the sub affine blocks while (3.25) indicates a 3.25 pixels shift.

13.3 Restriction to ensure worst-case bandwidth for INTER_9x9

$$\begin{aligned}&(4*Norm(v_{1x}-v_{0x})>-4*pel\ \&\&\ 4*Norm(v_{1x}-v_{0x})<pel)\ \&\&\\&(4*Norm(v_{1y}-v_{0y})>-pel\ \&\&\ 4*Norm(v_{1y}-v_{0y})<pel)\ \&\&\\&(4*Norm(v_{2x}-v_{0x})>-pel\ \&\&\ 4*Norm(v_{2x}-v_{0x})<pel)\ \&\&\\&(4*Norm(v_{2y}-v_{0y})>-4*pel\ \&\&\ 4*Norm(v_{2y}-v_{0y})<pel)\ \&\&\\&((4*Norm(v_{1x}-v_{0x})+4*Norm(v_{2x}-v_{0x})>-4*pel)\ \&\&\\&(4*Norm(v_{1x}-v_{0x})+4*Norm(v_{2x}-v_{0x})<pel))\ \&\&\\&((4*Norm(v_{1y}-v_{0y})+4*Norm(v_{2y}-v_{0y})>-4*pel)\ \&\&\\&(4*Norm(v_{1y}-v_{0y})+4*Norm(v_{2y}-v_{0y})<pel))\end{aligned}$$ Eq. (12)

[00181] Herein, $pel = 128 * 16$ (128 and 16 indicate the normalization factor and motion vector precision, respectively).

14. Drawbacks of existing methods for combined affine merge candidates

[00182] In some existing implementations, CPMVs are stored separately, therefore, additional memory is required.

[00183] In addition to the separately stored CPMVs, the width, height and the top-left position of a neighboring CU must be known to inherit the merge model from the neighboring affine coded CU. These pieces of side information will increase the line buffer size.

[00184] Other existing implementations that try to constrain the affine bandwidth impose additional computational burden at decoder.

15. Exemplary methods for representation of affine motion data

[00185] Embodiments of the disclosed technology store the affine model parameters instead

of storing the control point motion vectors (CPMVs), which address the bandwidth and line-buffer issues of affine prediction, and may improve video coding efficiency and enhance both existing and future video coding standards, is elucidated in the following examples described for various implementations. In the following examples, which should not be construed to be limiting, the coordinate of the top-left corner/top-right corner/bottom-left corner/bottom-right corner of the affine coded above or left neighboring CU are (LTNx, LTNy)/(RTNx, RTNy)/(LBNx, LBNy)/(RBNx, RBNy), respectively; the coordinate of the top-left corner/top-right corner/bottom-left corner/bottom-right corner of the current CU are (LTCx,LTCy)/(RTCx, RTCy)/(LBCx, LBCy)/(RBCx, RBCy), respectively; the width and height of the affine coded above or left neighboring CU are w' and h', respectively; the width and height of the affine coded current CU are w and h, respectively.

[00186] Furthermore, MV is 2-dimension vector noted as (MVx, MVy). MV1+MV2=MV3 means MV1x+MV2x=MV3x and MV1y+MV2y=MV3y. k×MV1=MV2 means k×MV1x=MV2x and k×MV1y=MV2y. Average(MV1, MV2)=((MV1x+MV2x)>>1, (MV1y+MV2y)>>1) or Average(MV1, MV2)=((MV1x+MV2x+1)>>1, (MV1y+MV2y+1)>>1).

[00187] In the examples that follow, SatShift(x,n) is defined as

[00188] $$SatShift(x,n)=\begin{cases}(x+offsset0)>>n & if \quad x\geq 0\\ -((-x+offset1)>>n) & if \quad x<0\end{cases}.$$

[00189] In one example, *offset0* and *offset1* are set to $(1 \ll (n-1))$.

[00190] In the examples that follow, Clip3(min, max, x) is defined as

[00191] $$Clip3(Min,Max,x)=\begin{cases}Min & if \quad x<Min\\ Max & if \quad x>Max\\ x & Otherwise\end{cases}.$$

[00192] Although the following examples are described in the context of an "affine merge candidate list," the are equally applicable to other merge candidate lists, e.g. "sub-block merge candidate list" and when other kinds of sub-block merge candidate such as ATMVP candidate is also put into the merge candidate list.

[00193] *Examples of affine motion information to be stored*

[00194] Example 1. The parameters *a*, *b*, *c*, *d*, *e* and *f* defined in Eq. (2) may be stored for a block if it is coded with affine mode.

[00195] (a) Alternatively, *a*, *b*, *c* and *d* defined in Eq. (2) may be stored for a block if it is

coded with affine mode. In this case, e and f are not stored any more.

[00196] (b) Alternatively, *a* and *b* defined in Eq. (1) are stored for a block if it is coded with the 4-parameter affine mode.

[00197] (c) Alternatively, *a*, *b*, *e* and *f* defined in Eq. (1) are stored for a block if it is coded with the 4-parameter affine mode.

[00198] (d) The parameters *a*, *b*, *c*, *d*, *e* and *f* defined in Eq. (2) are always stored for an affine coded block, but it is restricted that *c*=-*b*, *d*= *a*, if it is coded with 4-parameter affine mode.

[00199] (e) The parameters *a*, *b*, *c* and *d* defined in Eq. (2) are always stored for an affine coded block, but it is restricted that *c*=-*b*, *d*= *a*, if it is coded with 4-parameter affine mode.

[00200] (f) Which parameters to be stored may depend on the affine modes, inter or merge mode, block size, picture type, etc. al.

[00201] Example 2. In one example, the parameters to be stored can be calculated as below:

[00202] (a) $a = \left(mv_1^h - mv_0^h\right)/w$

[00203] (b) $b = \left(mv_1^v - mv_0^v\right)/w$

[00204] (c) $c = \left(mv_2^h - mv_0^h\right)/h$

[00205] (d) $d = \left(mv_2^v - mv_0^v\right)/h$

[00206] (e) $c = -b$ for 4-parameter affine prediction

[00207] (f) $d = a$ for 4-parameter affine prediction

[00208] (g) $e = mv_0^h$

[00209] (h) $f = mv_0^v$

[00210] (i) (*e*, *f*) = (mvx, mvy), where (mvx, mvy) can be any MV.

[00211] Example 3. It is proposed to calculate affine model parameters without division operations. Suppose the width and height of the current block noted as *w* and *h* are equal to 2^{WB} and 2^{HB}. *P* is an integer number defining the calculation precision, e.g., *P* is set to 7.

[00212] (a) $a = SatShift\left(P\left(mv_1^h - mv_0^h\right), WB\right)$

[00213] (b) $b = SatShift\left(P\left(mv_1^v - mv_0^v\right), WB\right)$

[00214] (c) $c = SatShift\left(P\left(mv_2^h - mv_0^h\right), WB\right)$

[00215] (d) $d = SatShift\left(P\left(mv_2^v - mv_0^v\right), WB\right)$

[00216] Example 4. The affine model parameters may be further clipped before being stored.

[00217] (a) In one example, suppose a parameter *x* (e.g. x= *a* or *b* or *c* or *d*) is stored with

K bits, then x = Clip3(-2^{K-1}, $2^{K-1}-1$, x).

[00218] (b) For example, a=Clip(-128, 127, a), then a is stored as a 8 bit signed integer.

[00219] Example 5. The affine model parameters may be stored for each M×N region. All sub-blocks (e.g., subM × subN sub-blocks) inside a M×N region share the same stored parameters.

[00220] (a) For example, M=N=4 or M=N=8, or M=N=16 or M=N=32, subM=subN=4.

[00221] (b) One of the sub-blocks within the region may be selected and its corresponding affine model parameters may be stored for the whole M×N region. Alternatively, the affine model parameters to be stored may be generated from affine model parameters of multiple sub-blocks.

[00222] (c) After encoding/decoding one tile/picture, the region size may be further adjusted wherein M'xN' region may be utilized to have the same affine model parameters and it could not happen that both M' = M and N' = N.

[00223] (d) In one example, the parameters stored in a region is set to be the parameters of a subM x subN block at the bottom-right corner of the region, e.g., B33 in FIG. 11.

[00224] (e) In one example, the parameters stored in a region is set to be the parameters of a subM x subN block at the bottom-left corner of the region, e.g., B03 in FIG. 11.

[00225] (f) In one example, the parameters stored in a region is set to be the parameters of a subM x subN block at the top-left corner of the region, e.g., B00 in FIG. 11.

[00226] (g) In one example, the parameters stored in a region is set to be the parameters of a subM x subN block at the top-right corner of the region, e.g., B30 in FIG. 11.

[00227] (h) In one example, the parameters stored in a region is set to be the parameters of a subM x subN block at the center of the region, e.g., B11 or B21 or B12 or B22 in FIG. 11.

[00228] (i) In one example, the parameters stored in a region is set to be the parameters of a subM x subN block at the middle of the bottom line of the region, e.g., B13 or B23 in FIG. 11.

[00229] (j) In one example, the parameters stored in a region is set to be the parameters of a subM x subN block at the middle of the left line of the region, e.g., B01 or B02 in FIG. 11.

[00230] (k) In one example, the parameters stored in a region is set to be the parameters of a subM x subN block at the middle of the right line of the region, e.g., B31 or B32 in FIG. 11.

[00231] (l) In one example, the parameters stored in a region is set to be the parameters of a subM x subN block at the middle of the top line of the region, e.g., B10 or B20 in FIG. 11.

[00232] *Examples of usage of stored affine model parameters*

[00233] Example 6. The affine model parameters stored in a neighboring block may be used to derive the affine prediction of a current block.

[00234] (a) In one example, the parameters stored in neighboring blocks may be utilized for motion vector prediction or motion vector coding of current block.

[00235] (b) In one example, the parameters stored in a neighboring block may be used to derive the control point MVs (CPMVs) of the current affine-coded block.

[00236] (c) In one example, the parameters stored in a neighboring block may be used to derive the MVs used in motion compensation for sub-blocks of the current affine-coded block.

[00237] (d) In one example, the parameters stored in a neighbouring block may be used to derive the prediction for CPMVs of the current affine-coded block. This prediction for CPMVs can be used to predict the CPMVs of the current block when CPMVs need to be coded.

[00238] Example 7. The affine model parameters stored in a neighboring block may be inherited by the current block if the current block utilizes the affine merge mode merging from the neighboring block. Alternatively, the affine model of current block may be derived from affine model parameters stored in one or more neighboring blocks.

[00239] Example 8. The affine model parameters of the current block can be signaled from the encoder to the decoder.

[00240] (a) The parameters stored in neighboring blocks can be used to predict the parameters of the current block.

[00241] Example 9. The affine model parameters may be stored after coding/decoding a picture. The stored parameters can be used to predict the affine parameters of a block coded in another picture to be coded/decoded.

[00242] Example 10. The parameters for the two reference lists (List0 and List1) are both stored.

[00243] (a) In one example, the parameters for the two reference lists are stored independently.

[00244] (b) Alternatively, the parameters for the two reference lists can be stored with prediction from one to the other.

[00245] Example 11. It is proposed that affine model parameters for luma component may be stored also for chroma components. When coding the chroma components, the associated affine

model parameters may be inherited or derived from those associated with luma component.

[00246] (a) Alternatively, affine model parameters for luma component may be stored and affine model parameters for two chroma components may be stored together or just stored for one chroma component or separately stored.

[00247] (b) In one example, whether to store separate affine model parameters for different color component may depend on the color format.

[00248] *Examples of temporal prediction of affine model parameters may be utilized*

[00249] Example 12. In one example, the stored affine model parameters of one or more blocks (e.g., a collocated block) in one or multiple pictures may be treated as the affine parameters of the current coding block with affine merge mode.

[00250] (a) In one example, the stored parameters of one or more blocks (e.g., a collocated block) in one or multiple pictures may be scaled (if necessary) before using as the affine parameters of the current coding block with affine merge mode.

[00251] (b) In one example, multiple temporal neighboring blocks may be checked in order to select one set of affine model parameters. For example, collocated block of the bottom-right neighboring 4x4/8x8 block of the CU, and collocated block of the center 4x4/8x8 block of the CU.

[00252] (c) In one example, temporal neighboring blocks may be identified by neighboring motion information.

[00253] (d) In one example, temporal neighboring blocks are from one so-called collocated picture. In one example, this collocated picture may be the same as that used in TMVP/ATMVP.

[00254] (e) In one example, the collocated picture may be signaled in VPS/SPS/PPS/slice header/tile group header.

[00255] (f) In one example, instead of inheriting or directly deriving from the temporal block, the associated affine model parameters of one or more blocks in different pictures may be used to predict the coding of affine model parameters of a current block, or predict the coding of CPMVs for a current block.

[00256] Example 13. The stored parameters of one or multiple blocks (e.g., collocated block) in one or more other pictures may be used as the prediction of the affine parameters of the current coding block with affine inter mode.

[00257] (a) In one example, the stored parameters of a collocated block in another picture may be scaled (if necessary) before using as the prediction of the affine parameters of the current coding block with affine inter mode.

[00258] (b) In one example, multiple temporal neighboring blocks may be checked. For example, collocated block of the bottom-right neighboring 4x4/8x8 block of the CU, and collocated block of the center 4x4/8x8 block of the CU.

[00259] (c) In one example, temporal neighboring blocks may be identified by neighboring motion information.

[00260] (d) In one example, the collocated picture may be the same as used in TMVP/ATMVP.

[00261] (e) In one example, the collocated picture may be signaled in VPS/SPS/PPS/slice header/tile group header.

[00262] Example 14. The motion information stored in a neighboring M×N unit block (e.g. 4×4 block in VTM) and the affine parameters stored in that unit block can be used together to derive the CPMVs or the MVs of sub-blocks used in motion compensation.

[00263] (a) Suppose the coordinate of the top-left position of the unit block is (x0, y0), then the coordinate of the center position of the unit block (xm, ym) can be calculated as:

[00264] (i) xm = x0+M/2, ym = y0+N/2;

[00265] (ii) xm = x0+M/2-1, ym = y0+N/2-1;

[00266] (iii) xm = x0+M/2-1, ym = y0+N/2;

[00267] (iv) xm = x0+M/2, ym = y0+N/2-1;

[00268] (b) Suppose the MV stored in the unit block is (mv^h_0, mv^v_0), the coordinate of the position (x, y) for which the MV $(mv^h(x,y), mv^v(x,y))$ is derived. Suppose the coordinate of the top-left corner of the current block is (x0', y0'), the width and height of the current block is w and h, then

[00269] (i) To derive a CPMV, (x, y) can be (x0', y0') or (x0'+w, y0') or (x0', y0'+h) or (x0'+w, y0'+h).

[00270] (ii) To derive a MV for a sub-block of the current block, (x, y) can be the center of the sub-block. Suppose (x00, y00) is the top-left position of a sub-block, the sub-block size is M×N, then

[00271] (1) xm = x00+M/2, ym = y00+N/2;

[00272] (2) xm = x00+M/2-1, ym = y00+N/2-1;

[00273] (3) xm = x00+M/2-1, ym = y00+N/2;

[00274] (4) xm = x00+M/2, ym = y00+N/2-1;

[00275] (iii) In one example,

[00276] $$\begin{cases} mv^h(x,y) = a(x-xm) - b(y-ym) + mv_0^h \\ mv^v(x,y) = b(x-xm) + a(y-ym) + mv_0^v \end{cases}$$

[00277] if the neighboring block is coded with the 4-parameter affine mode.

[00278] (iv) In one example,

[00279] $$\begin{cases} mv^h(x,y) = ax + cy + mv_0^h \\ mv^v(x,y) = bx + dy + mv_0^v \end{cases}$$

[00280] if the neighboring block is coded with the 6-parameter affine mode.

[00281] (v) In one example,

[00282] $$\begin{cases} mv^h(x,y) = ax + cy + mv_0^h \\ mv^v(x,y) = bx + dy + mv_0^v \end{cases}$$

[00283] regardless if the neighboring block is coded with the 4-parameter affine mode or the 6-parameter affine mode.

[00284] (c) In one example, CPMVs of the current block are derived from the motion vector and parameters stored in a neighboring block, and these CPMVs serves as MVPs for the signaled CPMVs of the current block.

[00285] (d) In one example, CPMVs of the current block are derived from the motion vector and parameters stored in a neighboring block, and these CPMVs are used to derive the MVs of each sub-block used for motion compensation.

[00286] (e) In one example, the MVs of each sub-block used for motion compensation are derived from the motion vector and parameters stored in a neighboring block, if the current block is affine merge coded.

[00287] Example 15. Pruning of affine model parameters from multiple neighboring blocks (spatial and/or temporal neighboring blocks) may be applied before being used for affine model parameter inheritance/derivation/prediction.

[00288] (a) The affine model parameters of different neighboring blocks can be compared to decide whether the parameters from one neighboring block are not same or similar to the

parameters from another neighboring blocks already in the affine merge or affine AMVP candidate list, and the parameters from the neighboring block should be put into the candidate list. Suppose {a, b, c, d, e, f} are parameters from of a neighboring block and {a', b', c', d', e', f'} are parameters from of a neighboring block. The affine model with parameters {a, b, c, d, e, f} is considered redundant and not put into the candidate list when the affine model with parameters {a', b', c', d', e', f'} is already in the candidate list if and only if:

[00289] (b) $a == a'$ in one example.

[00290] (c) $b == b'$ in one example.

[00291] (d) $c == c'$ in one example.

[00292] (e) $d == d'$ in one example.

[00293] (f) $a == a'$ and $b == b'$ in one example.

[00294] (g) $c == c'$ and $d == d'$ in one example.

[00295] (h) $a == a'$ and $b == b'$ and $c == c'$ in one example.

[00296] (i) $a == a'$ and $b == b'$ and $c == c'$ and $d == d'$ in one example.

[00297] Example 16. Whether to and how to apply the proposed methods may depend on the position of the current block and neighboring blocks.

[00298] (a) In one example, the one or multiple of proposed methods may only be applied if the current block is at the bottom in a CTU, or at the bottom in a M×N region (e.g. M=N=64).

[00299] (b) In one example, the one or multiple of proposed methods may only be applied if the current block is at the top in a CTU, or at the top in a M×N region (e.g. M=N=64).

[00300] (c) In one example, the one or multiple of proposed methods may only be applied if the current block is at the left in a CTU, or at the left in a M×N region (e.g. M=N=64).

[00301] (d) In one example, the one or multiple of proposed methods may only be applied if the current block is at the right in a CTU, or at the right in a M×N region (e.g. M=N=64).

[00302] The examples described above may be incorporated in the context of the methods described below, e.g., method 1200, which may be implemented at a video decoder/encoder.

[00303] FIG. 12 shows a flowchart of an exemplary method for video coding. The method 1200 includes, at step 1210, selecting, during a current affine calculation step in a conversion between a current video block and a bitstream representation of the current video block, one or more parameters of a first set of affine model parameters, where the first set of affine model parameters is based on one or more previous affine calculation steps.

[00304] In some embodiments, and in the context of Example 1, the first set of affine model parameters comprises six variables (*a, b, c, d, e, f*) corresponding to a six-parameter affine model defined in Eq. (1). In an example, the one or more parameters comprise (*a, b, c, d, e, f*). In another example, the one or more parameters comprise (*a, b, c, d*).

[00305] In some embodiments, and in the context of Example 1, the first set of affine model parameters comprises four variables (*a, b, e, f*) corresponding to a four-parameter affine model defined in Eq. (2). In one example, the one or more parameters comprise (*a, b, e, f*). In another example, the one or more parameters comprise (*a, b*).

[00306] In some embodiments, and in the context of Example 11, the current video block comprises a luma component and a chroma component, and the first set of affine model parameters are associated with both the luma component and the chroma component.

[00307] In some embodiments, and in the context of Example 12, the first set of affine model parameters are associated with one or more collocated blocks. In other embodiments, the first set of affine model parameters are associated with one temporal neighboring block of a plurality of temporal neighboring blocks. For example, the one temporal neighboring block is identified based on neighboring motion information. For example, the one temporal neighboring block is from a collocated picture. For example, TMVP or ATMVP for the current video block is based on the collocated picture. For example, the collocated picture is signaled in a video parameter set (VPS), a sequence parameter set (SPS), a picture parameter set (PPS), a slice header or a tile group header.

[00308] In some embodiments, and in the context of Example 14, the first set of affine model parameters are associated with a neighboring M×N unit block, and performing the conversion is further based on motion information corresponding to the neighboring M×N unit block. Furthermore, the method 1200 further includes the step of deriving, for motion compensation, CPMVs or motion vectors of sub-blocks of the current video block.

[00309] The method 1200 includes, at step 1220, performing, based on the one or more parameters and by refraining from using control point motion vectors (CPMVs) of the one or more previous affine calculation steps, the conversion between the current video block and the bitstream representation. In some embodiments, the conversion generates the current block from the bitstream representation (e.g., as might be implemented in a video decoder). In other embodiments, the conversion generates the bitstream representation from the current block (e.g.,

as might be implemented in a video encoder).

[00310] In some embodiments, and in the context of Example 6, the first set of affine model parameters are associated with a neighboring block of the current video block. In an example, performing the conversion comprises motion vector prediction or motion vector coding of the current video block. In another example, performing the conversion comprises deriving one or more CPMVs of the current video block. In yet another example, performing the conversion comprises deriving one or more motion vectors for motion compensation for sub-blocks of the current video block. In another example, performing the conversion comprises deriving a prediction for one or more CPMVs of the current video block.

[00311] In some embodiments, and in the context of Example 10, the first set of affine model parameters are associated with a first reference list, and performing the conversion is further based on one or more parameters of a second set of affine model parameters that are associated with a second reference list.

[00312] In some embodiments, and in the context of Example 13, the method 1200 further includes the step of scaling, prior to the performing the conversion, the one or more parameters of the first set of affine model parameters.

[00313] In some embodiments, and in the context of Example 4, the method 1200 further includes the step of clipping, prior to the performing the conversion, the one or more parameters of the first set of affine model parameters.

[00314] In some embodiments, and in the context of Example 15, the method 1200 further includes the step of pruning, prior to the performing the conversion, a plurality of sets of affine model parameters associated with the plurality of temporal neighboring blocks.

[00315] In some embodiments, and in the context of Example 16, performing the conversion is further based on a position of the current video block.

16. Example implementations of the disclosed technology

[00316] FIG. 13 is a block diagram of a video processing apparatus 1300. The apparatus 1300 may be used to implement one or more of the methods described herein. The apparatus 1300 may be embodied in a smartphone, tablet, computer, Internet of Things (IoT) receiver, and so on. The apparatus 1300 may include one or more processors 1302, one or more memories 1304 and video processing hardware 1306. The processor(s) 1302 may be configured to implement one or more methods (including, but not limited to, method 1200) described in the present document.

The memory (memories) 1304 may be used for storing data and code used for implementing the methods and techniques described herein. The video processing hardware 1306 may be used to implement, in hardware circuitry, some techniques described in the present document.

[00317] FIG. 14 shows a flowchart of an exemplary method for video coding. The method 1400 includes, at step 1410, selecting, during a current affine calculation step in 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; at step 1420, storing the one or more parameters for the current block; and at step 1430, performing, based on the one or more parameters, the conversion between the current block and the bitstream representation of the current block.

[00318] FIG. 15 shows a flowchart of another exemplary method for video coding. The method 1500 includes, at step 1510, acquiring, during a conversion between a current block and a bitstream representation of the current block, motion information of the current block, wherein the motion information of the current block is based on at least one affine model parameter of a neighboring block of the current block; and at step 1520, performing, based on the motion information, the conversion between the current block and the bitstream representation of the current block.

[00319] In the present document, the term "video processing" may refer to video encoding, video decoding, video compression or video decompression. For example, video compression algorithms may be applied during conversion from pixel representation of a video to a corresponding bitstream representation or vice versa.

[00320] From the foregoing, it will be appreciated that specific embodiments of the presently disclosed technology have been described herein for purposes of illustration, but that various modifications may be made without deviating from the scope of the invention. Accordingly, the presently disclosed technology is not limited except as by the appended claims.

[00321] Implementations of the subject matter and the functional operations described in this patent document can be implemented in various systems, digital electronic circuitry, or in computer software, firmware, or hardware, including the structures disclosed in this specification and their structural equivalents, or in combinations of one or more of them. Implementations of the subject matter described in this specification can be implemented as one or more computer program products, i.e., one or more modules of computer program instructions encoded on a tangible and non-transitory computer readable medium for execution by, or to control the

operation of, data processing apparatus. The computer readable medium can be a machine-readable storage device, a machine-readable storage substrate, a memory device, a composition of matter effecting a machine-readable propagated signal, or a combination of one or more of them. The term "data processing unit" or "data processing apparatus" encompasses all apparatus, devices, and machines for processing data, including by way of example a programmable processor, a computer, or multiple processors or computers. The apparatus can include, in addition to hardware, code that creates an execution environment for the computer program in question, e.g., code that constitutes processor firmware, a protocol stack, a database management system, an operating system, or a combination of one or more of them.

[00322] 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.

[00323] The processes and logic flows described in this specification can be performed by one or more programmable processors executing one or more computer programs to perform functions by operating on input data and generating output. The processes and logic flows can also be performed by, and apparatus can also be implemented as, special purpose logic circuitry, e.g., an FPGA (field programmable gate array) or an ASIC (application specific integrated circuit).

[00324] Processors suitable for the execution of a computer program include, by way of example, both general and special purpose microprocessors, and any one or more processors of any kind of digital computer. Generally, a processor will receive instructions and data from a read only memory or a random access memory or both. The essential elements of a computer are a processor for performing instructions and one or more memory devices for storing instructions

and data. Generally, a computer will also include, or be operatively coupled to receive data from or transfer data to, or both, one or more mass storage devices for storing data, e.g., magnetic, magneto optical disks, or optical disks. However, a computer need not have such devices. Computer readable media suitable for storing computer program instructions and data include all forms of nonvolatile memory, media and memory devices, including by way of example semiconductor memory devices, e.g., EPROM, EEPROM, and flash memory devices. The processor and the memory can be supplemented by, or incorporated in, special purpose logic circuitry.

[00325] It is intended that the specification, together with the drawings, be considered exemplary only, where exemplary means an example. Additionally, the use of "or" is intended to include "and/or", unless the context clearly indicates otherwise.

[00326] While this patent document contains many specifics, these should not be construed as limitations on the scope of any invention or of what may be claimed, but rather as descriptions of features that may be specific to particular embodiments of particular inventions. Certain features that are described in this patent document in the context of separate embodiments can also be implemented in combination in a single embodiment. Conversely, various features that are described in the context of a single embodiment can also be implemented in multiple embodiments separately or in any suitable subcombination. Moreover, although features may be described above as acting in certain combinations and even initially claimed as such, one or more features from a claimed combination can in some cases be excised from the combination, and the claimed combination may be directed to a subcombination or variation of a subcombination.

[00327] 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.

[00328] Only a few implementations and examples are described and other implementations, enhancements and variations can be made based on what is described and illustrated in this patent document.

CLAIMS

What is claimed is:

1. A video processing method, comprising:

selecting, during a current affine calculation step in 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;

storing the one or more parameters for the current block; and

performing, based on the one or more parameters, the conversion between the current block and the bitstream representation of the current block.

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

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

4 The method of anyone of claims 1-3, 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^h_0 is a horizontal motion vector component of a top-left corner control point of the current block, mv^h_1 is a horizontal motion vector component of a top-right corner control point of the current block, and w is width of the current block.

5. The method of anyone of claims 1-3, 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^v_0 is a vertical motion vector component of a top-left corner control point of the current block, mv^v_1 is a vertical motion vector component of a top-right corner control point of the current block, and w is width of the current block.

6. The method of claims 1-3, 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^h_0 is a horizontal motion vector component of a top-left

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

7. The method of claims1-3, 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^v_0 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 height of the current block.

8. The method of anyone of claims 1-3, wherein a parameter *e* of the set of affine model parameters is calculated by $e = mv_0^h$, where mv^h_0 is a horizontal motion vector component of a top-left corner control point of the current block.

9. The method of anyone of claims 1-3, wherein a parameter *f* of the set of affine model parameters is calculated by $f = mv_0^v$, where mv^v_0 is a vertical motion vector component of a top-left corner control point of the current block.

10. The method of anyone of claims1-3, 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.

11. The method of anyone of claims 1-3, 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.

12 The method of claim 11, wherein 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^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.

13. The method of claim 11, 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^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.

14. The method of claim 11, wherein 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^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.

15. The method of claim 11, wherein 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^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.

16. The method of anyone of claims 12-15, wherein P is set to 7.

17 The method of anyone of claims 1-16, further comprising: clipping the one or more parameters, prior to the storing the one or more parameters for the current block.

18. The method of claim 17, 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.

19. The method of claim 18, wherein X=a, a=Clip (-128, 127, a), and the parameter a is stored as an eight (8) bit signed integer.

20. The method of anyone of claims 1 to 19, 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

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

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.

21. The method of claim 20, wherein the one or more parameters comprise a, b, c, and d.

22. The method of anyone of claims 1 to 3, wherein the set of affine model parameters comprises four variables (a, b, e, f) corresponding to a four-parameter affine model given by

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

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, m^{hv}_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.

23. The method of claim 22, wherein the one or more parameters comprise a and b.

24. The method of claim 22, wherein the one or more parameters comprise a, b, e and f.

25. The method of claim 20, 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 mode.

26. The method of claim 20, 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.

27. The method of anyone of claims 1-20 and 22, wherein the selecting of the one or more parameters depends on at least one of: affine modes, inter or merge mode, block size, picture type.

28. The method of claim 20 or 21, 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.

29. The method of claim 20 or 21, 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.

30. The method of anyone of claims 1-29, wherein for a region with a size of M×N, which includes sub-blocks with dimensions subM × subN, all the sub-blocks share same one or more parameters stored for the region, where M, N, subM and subN are integers greater than one.

31. The method of claim 30, wherein M=N=4 or M=N=8 or M=N=16 or M=N=32, and subM=subN=4.

32. The method of claim 30 or 31, wherein one sub-block of the sub-blocks is selected, and the one or more parameters stored for the region are based on the set of affine model parameters of the selected one sub-block.

33. The method of claim 30 or 31, wherein multiple sub-blocks of the sub-blocks are selected, and the one or more parameters stored for the region are based on the sets of affine model parameters of the selected multiple sub-blocks.

34. The method of anyone of claims 30-33, wherein after the performing, based on the one or more parameters, the conversion between the current block and the bitstream representation of the current block, the size of the region is adjusted to be M'×N', where M' ≠ M and/or N' ≠ N, and wherein the region with the size M'×N' utilize the same one or more affine model parameters.

35. The method of claim 32, wherein selected one sub-block is a sub-block at a bottom-right

corner of the region.

36. The method of claim 32, wherein selected one sub-block is a sub-block at a bottom-left corner of the region.

37. The method of claim 32, wherein selected one sub-block is a sub-block at a top-left corner of the region.

38. The method of claim 32, wherein selected one sub-block is a sub-block at a top-right corner of the region.

39. The method of claim 32, wherein selected one sub-block is a sub-block at the center of the region, that is, the selected one sub-block is not on any boundary of the region.

40. The method of claim 32, wherein selected one sub-block is a sub-block on a bottom boundary of the region, but not at any corner of the region.

41. The method of claim 32, wherein selected one sub-block is a sub-block on a left boundary of the region, but not at any corner of the region.

42. The method of claim 32, wherein selected one sub-block is a sub-block on a right boundary of the region, but not at any corner of the region.

43. The method of claim 32, wherein selected one sub-block is a sub-block on a top boundary of the region, but not at any corner of the region.

44. 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 43.

45. 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 43.

46. A video decoding apparatus comprising a processor configured to implement a method recited in one or more of claims 1 to 43.

47. A video encoding apparatus comprising a processor configured to implement a method recited in one or more of claims 1 to 43.

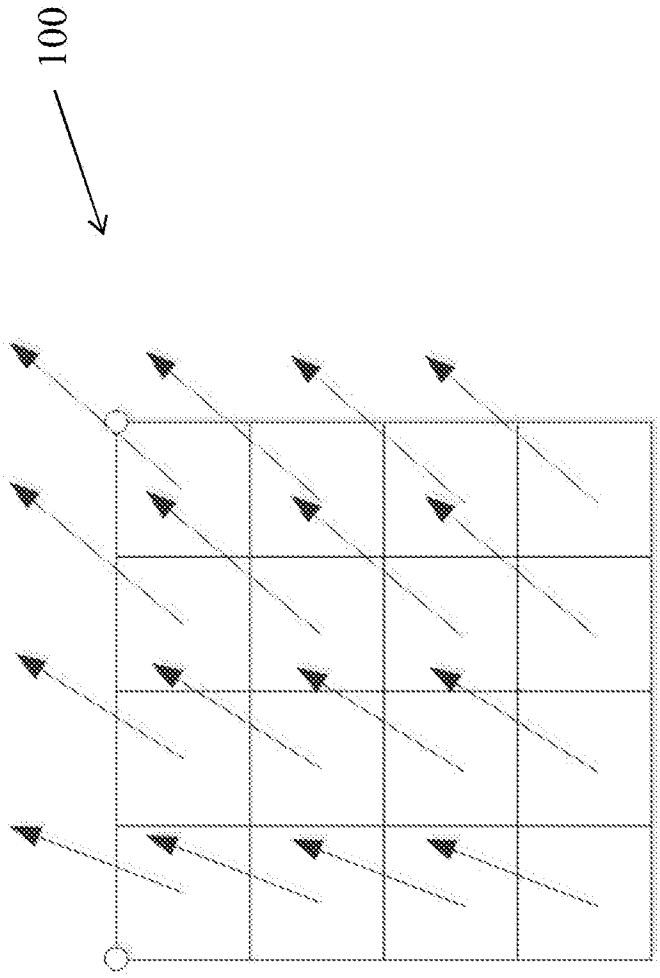

FIG. 1

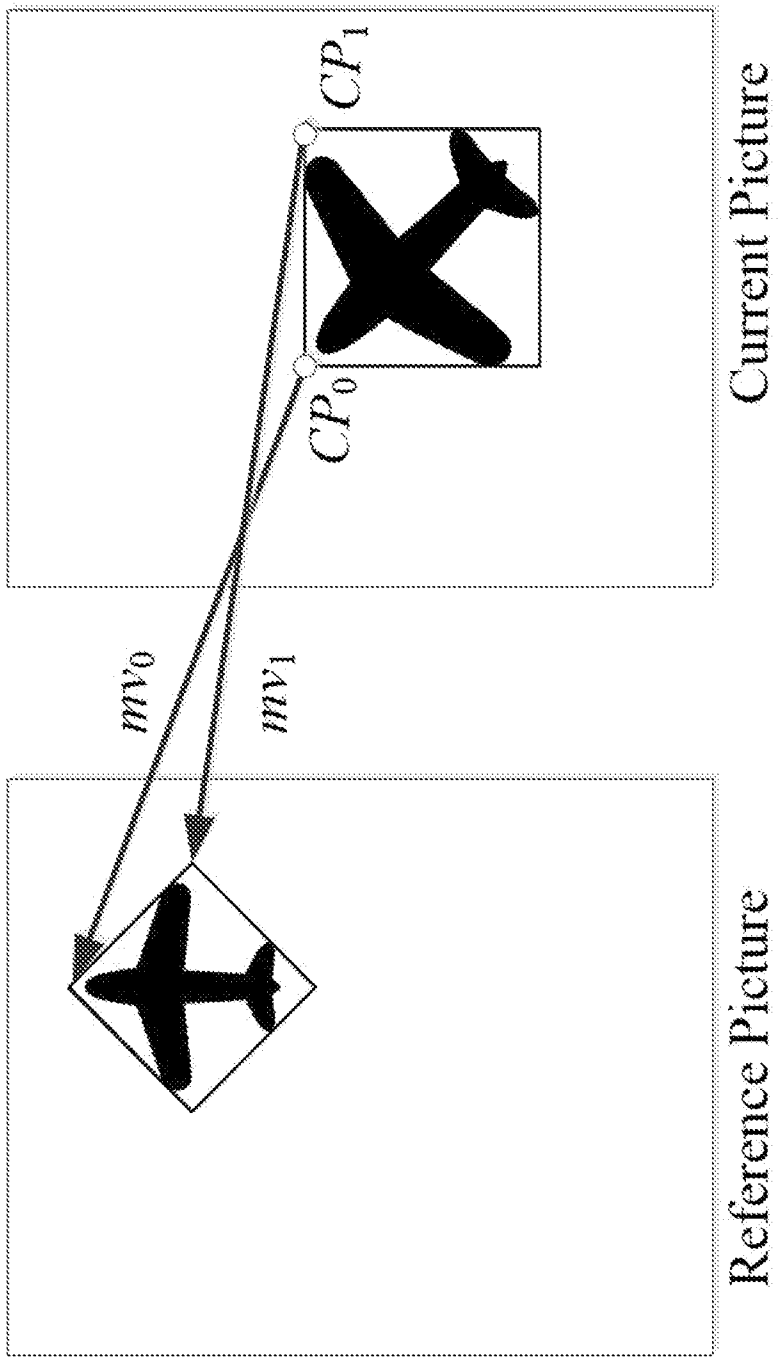

FIG. 2A

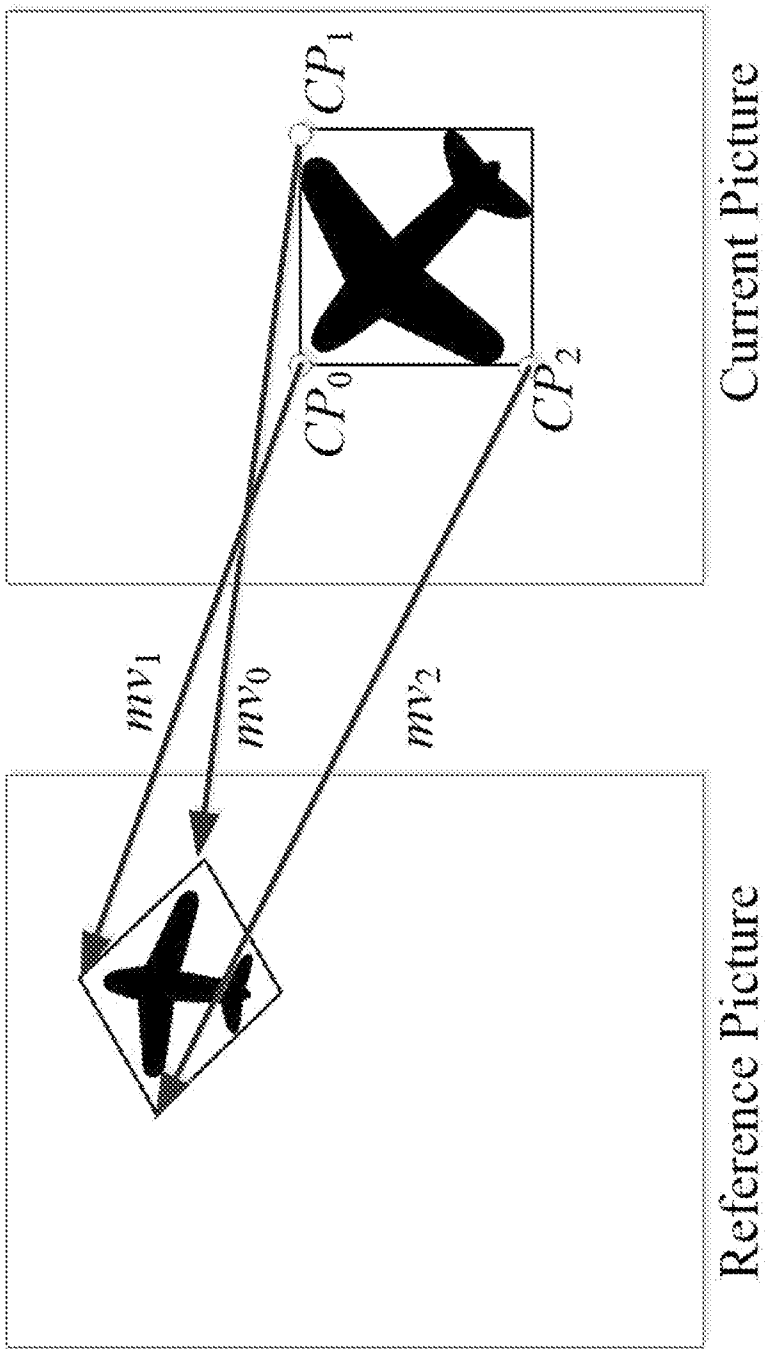

FIG. 2B

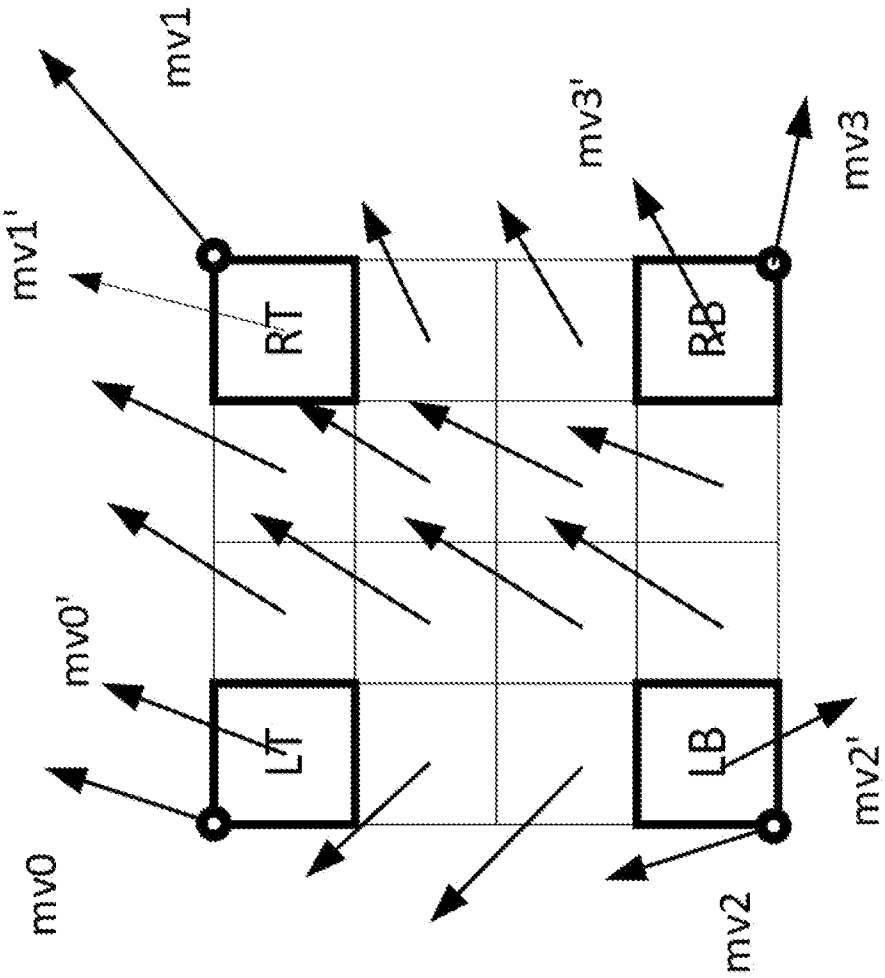

FIG. 3

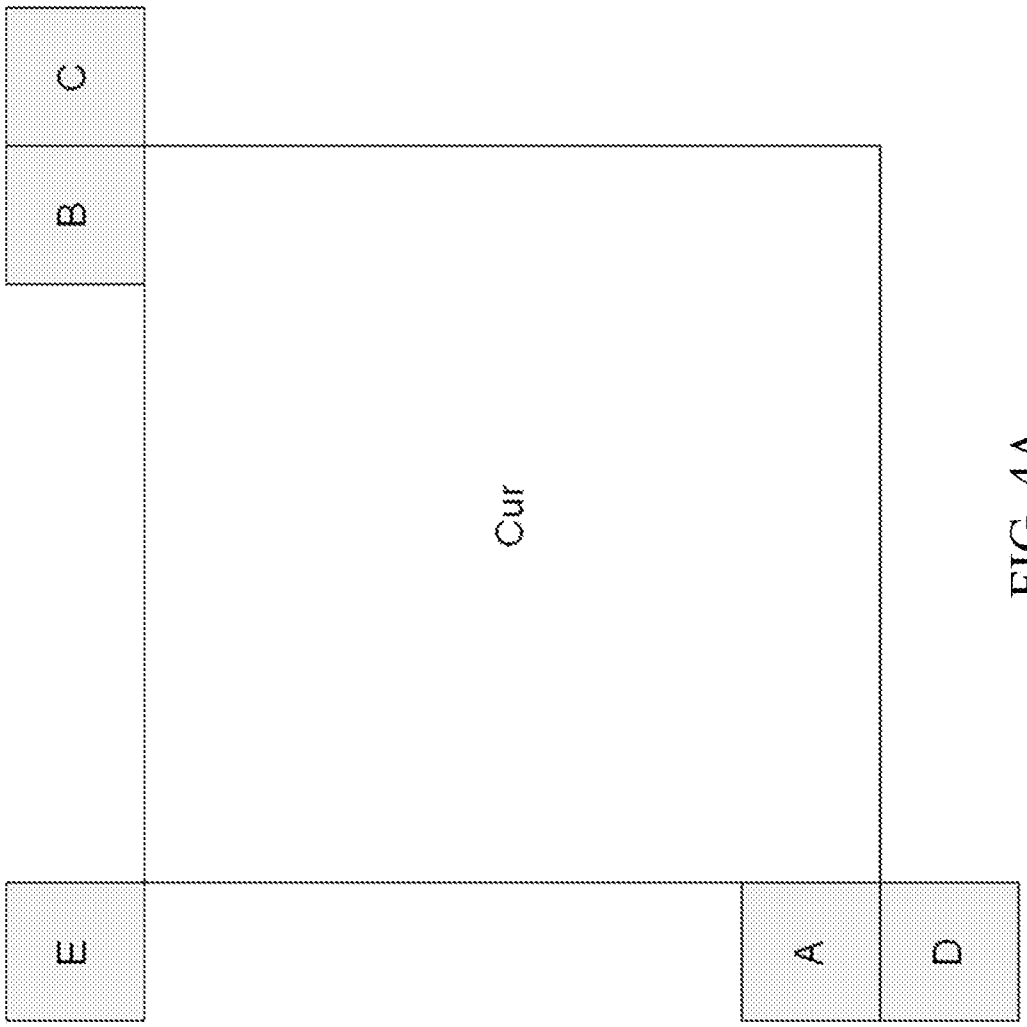

FIG. 4A

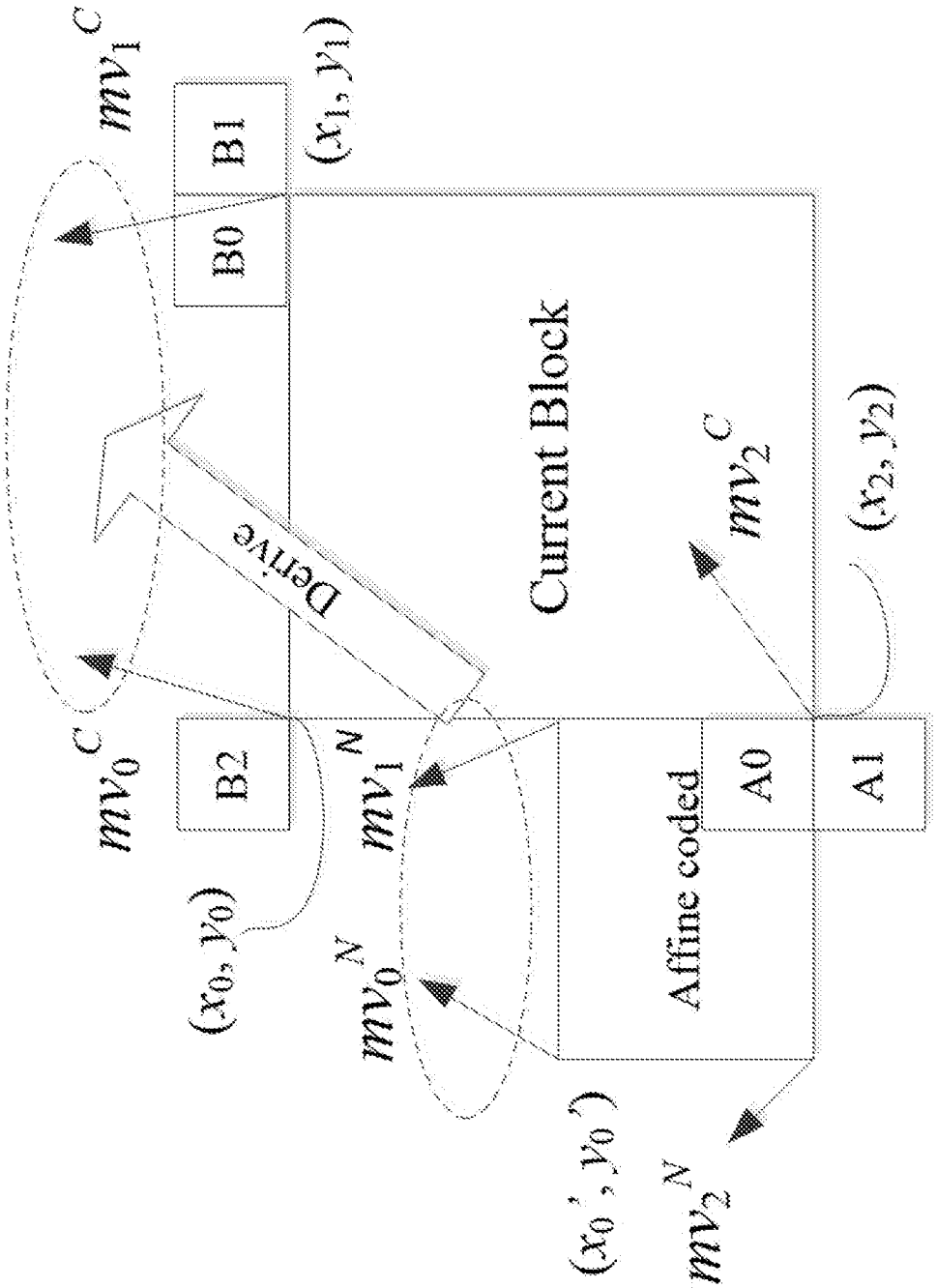
mv_1^C
B1
B0
(x_1, y_1)
Current Block
Derive
mv_2^C
(x_2, y_2)
mv_0^C
B2
mv_1^N
Affine coded
A0
A1
(x_0, y_0)
mv_0^N
(x_0', y_0')
mv_2^N

FIG. 4B

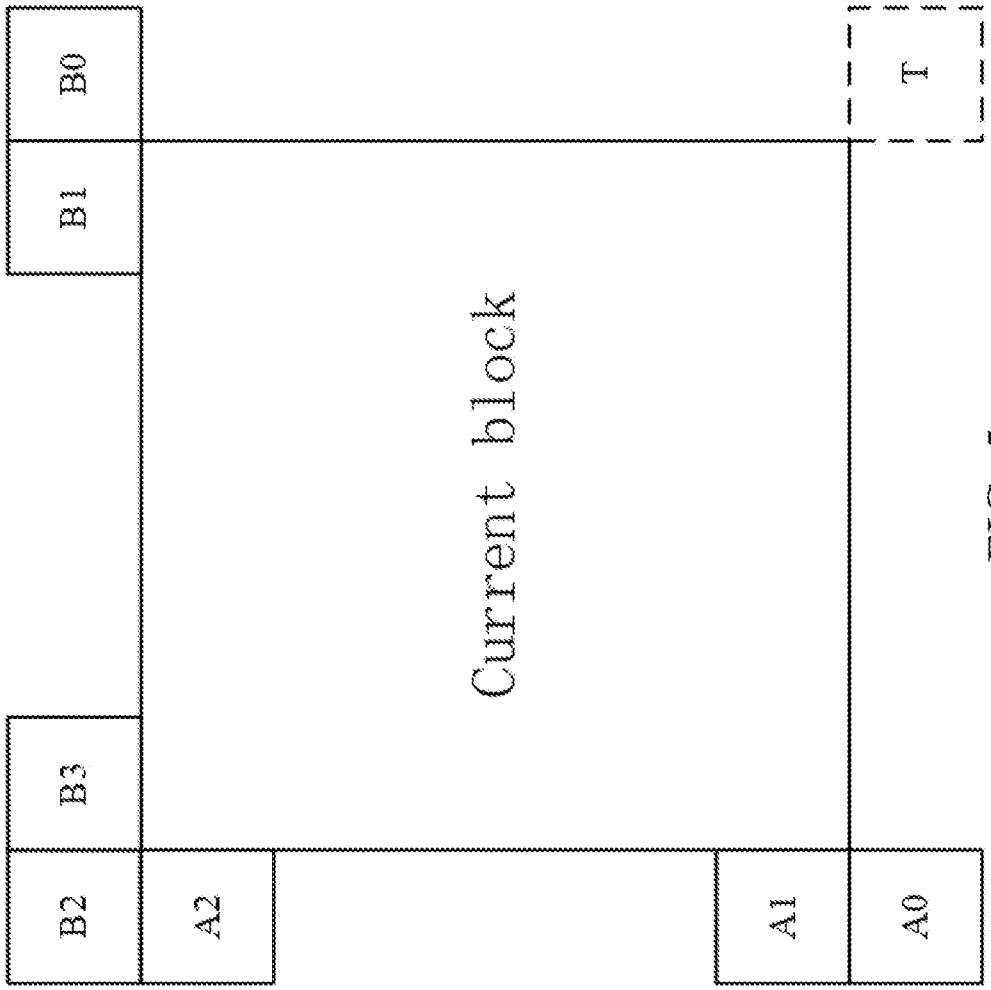

FIG. 5

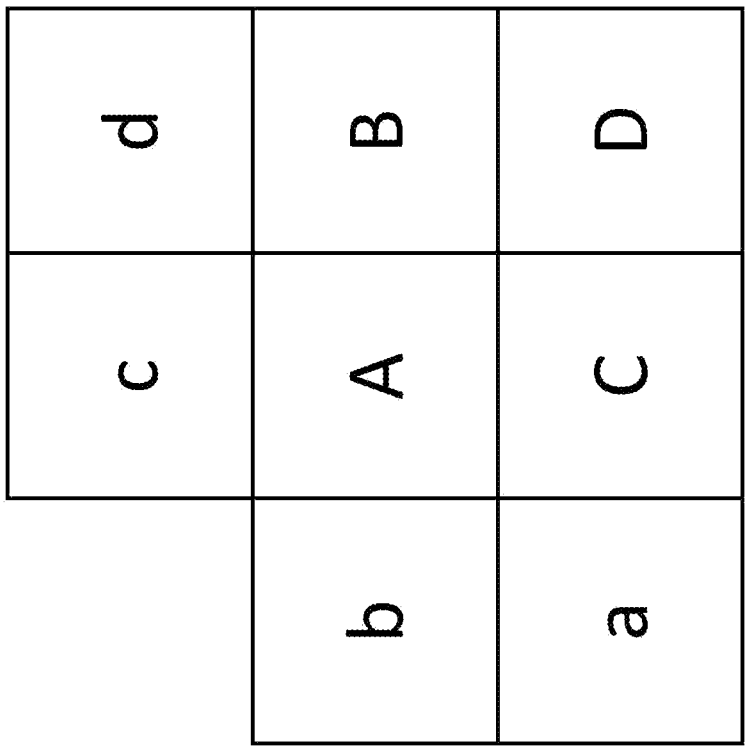

FIG. 6

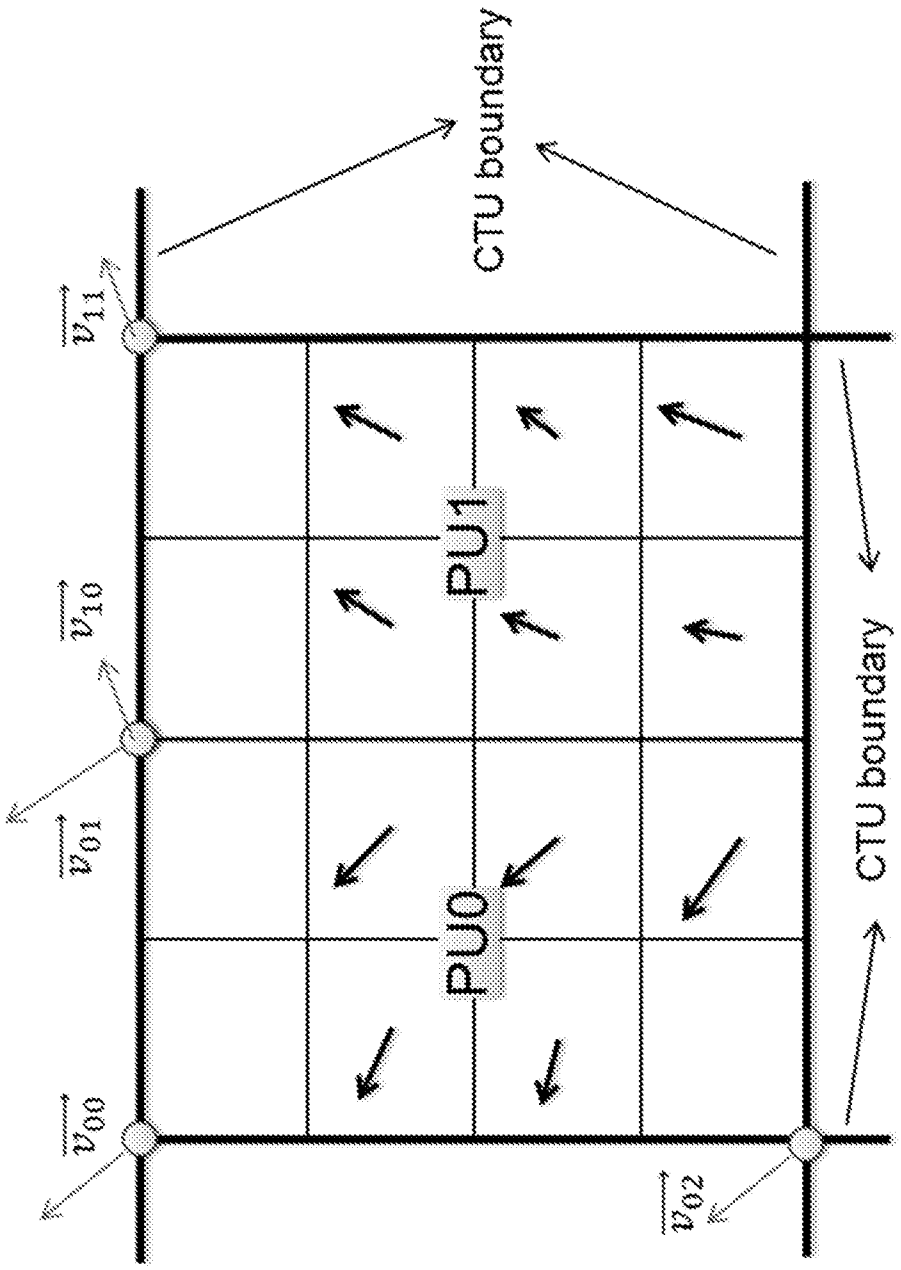
CTU boundary
PU1
PU0
CTU boundary
$\overrightarrow{v_{11}}$
$\overrightarrow{v_{10}}$
$\overrightarrow{v_{01}}$
$\overrightarrow{v_{00}}$
$\overrightarrow{v_{02}}$

FIG. 7A

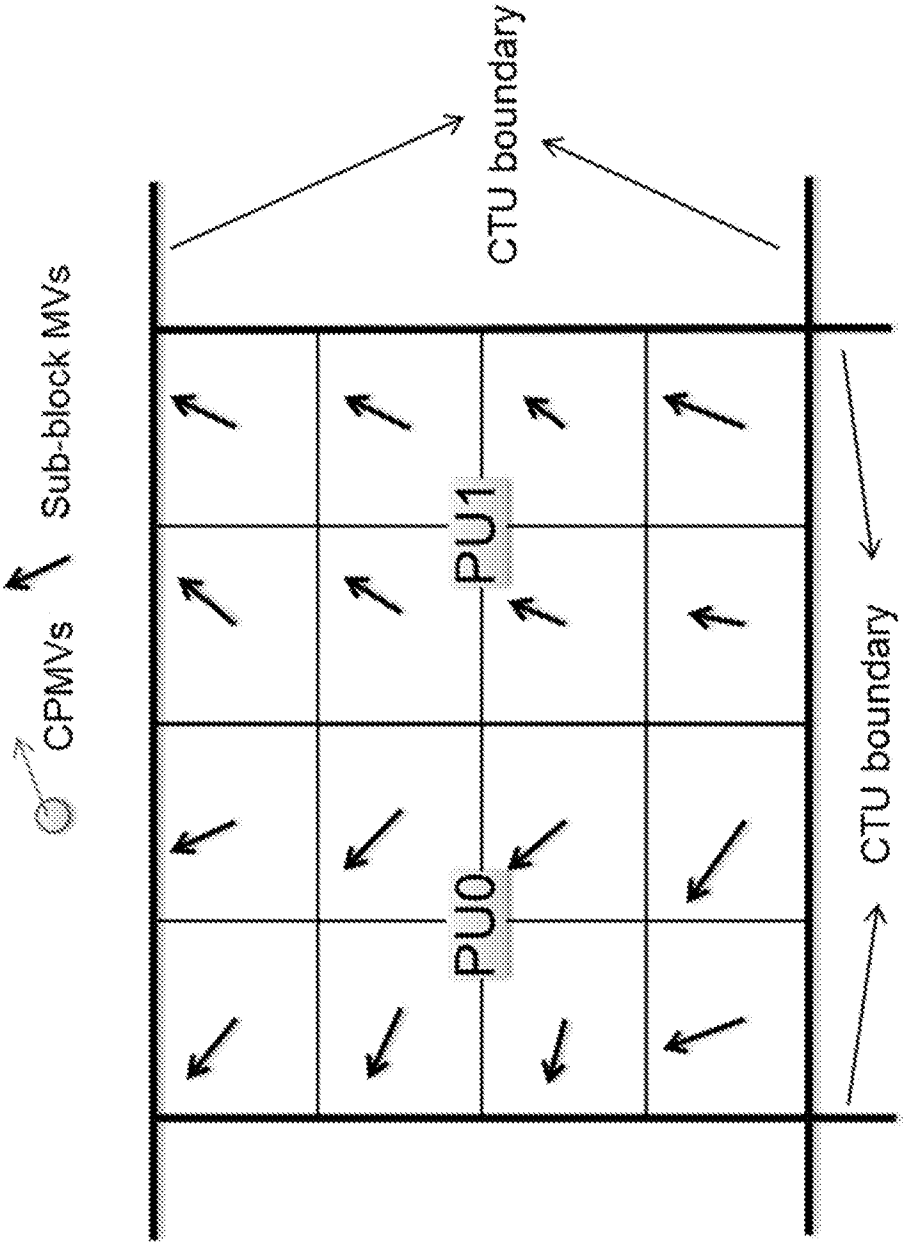

FIG. 7B

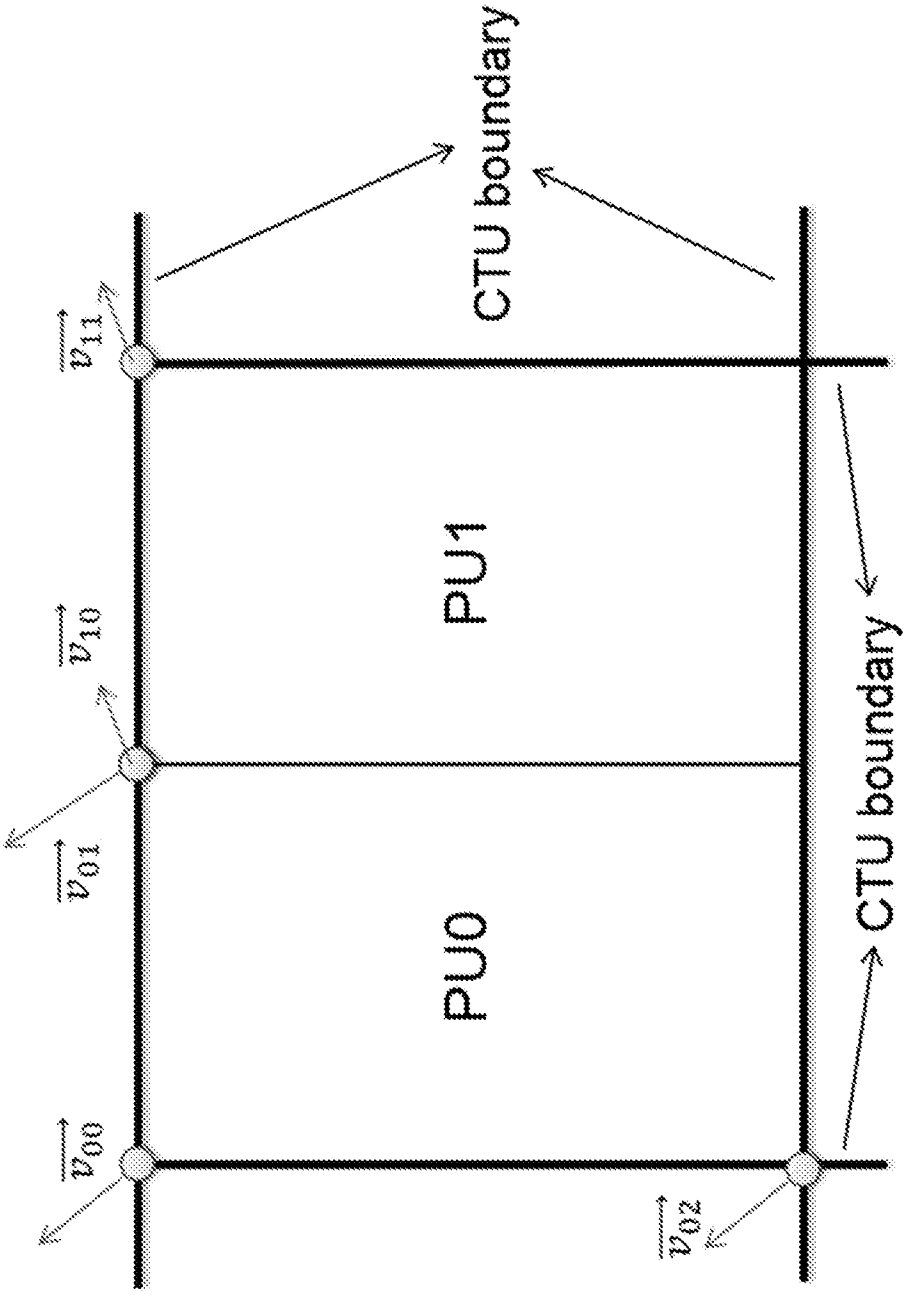
CTU boundary
PU1
PU0
CTU boundary
$\overrightarrow{v_{11}}$
$\overrightarrow{v_{10}}$
$\overrightarrow{v_{01}}$
$\overrightarrow{v_{00}}$
$\overrightarrow{v_{02}}$

FIG. 8A

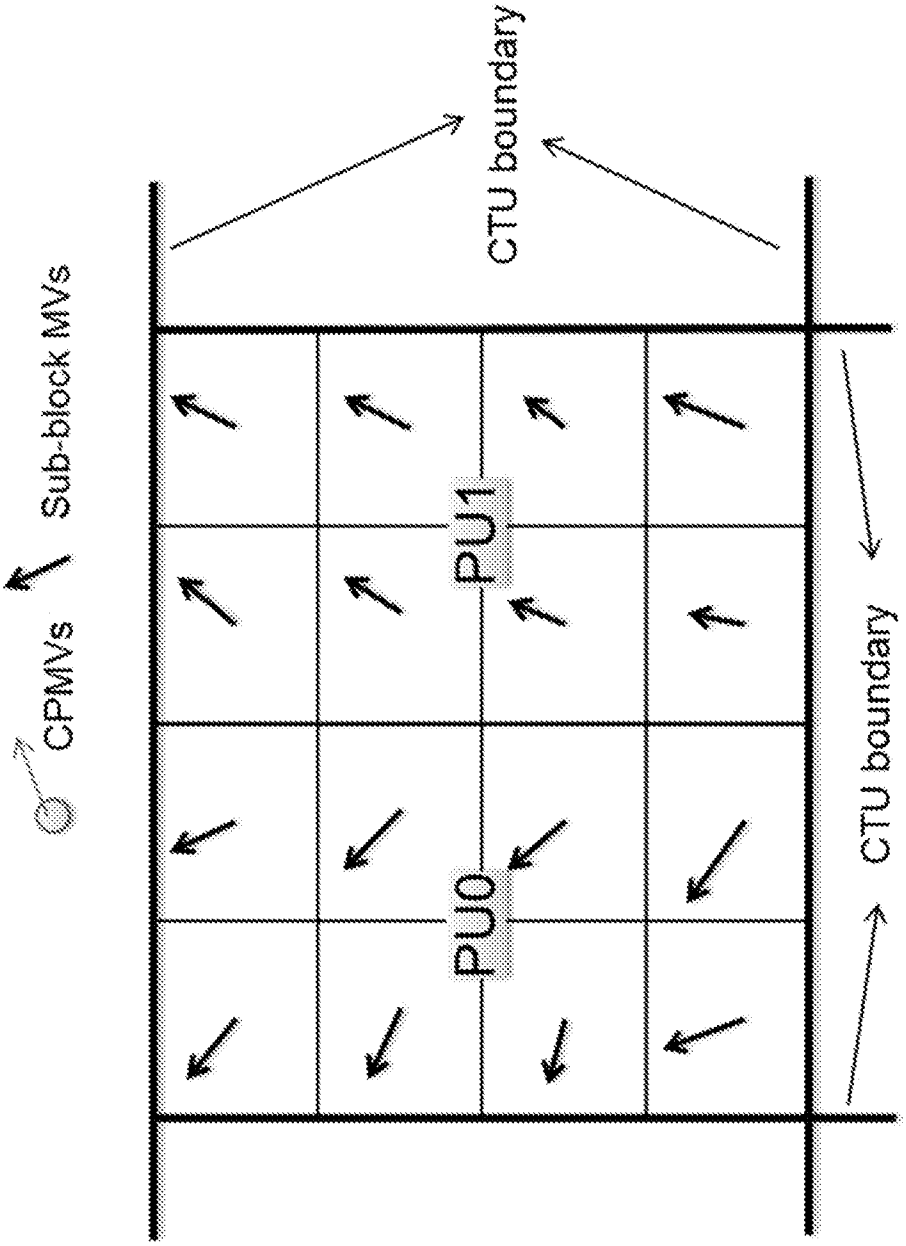

FIG. 8B

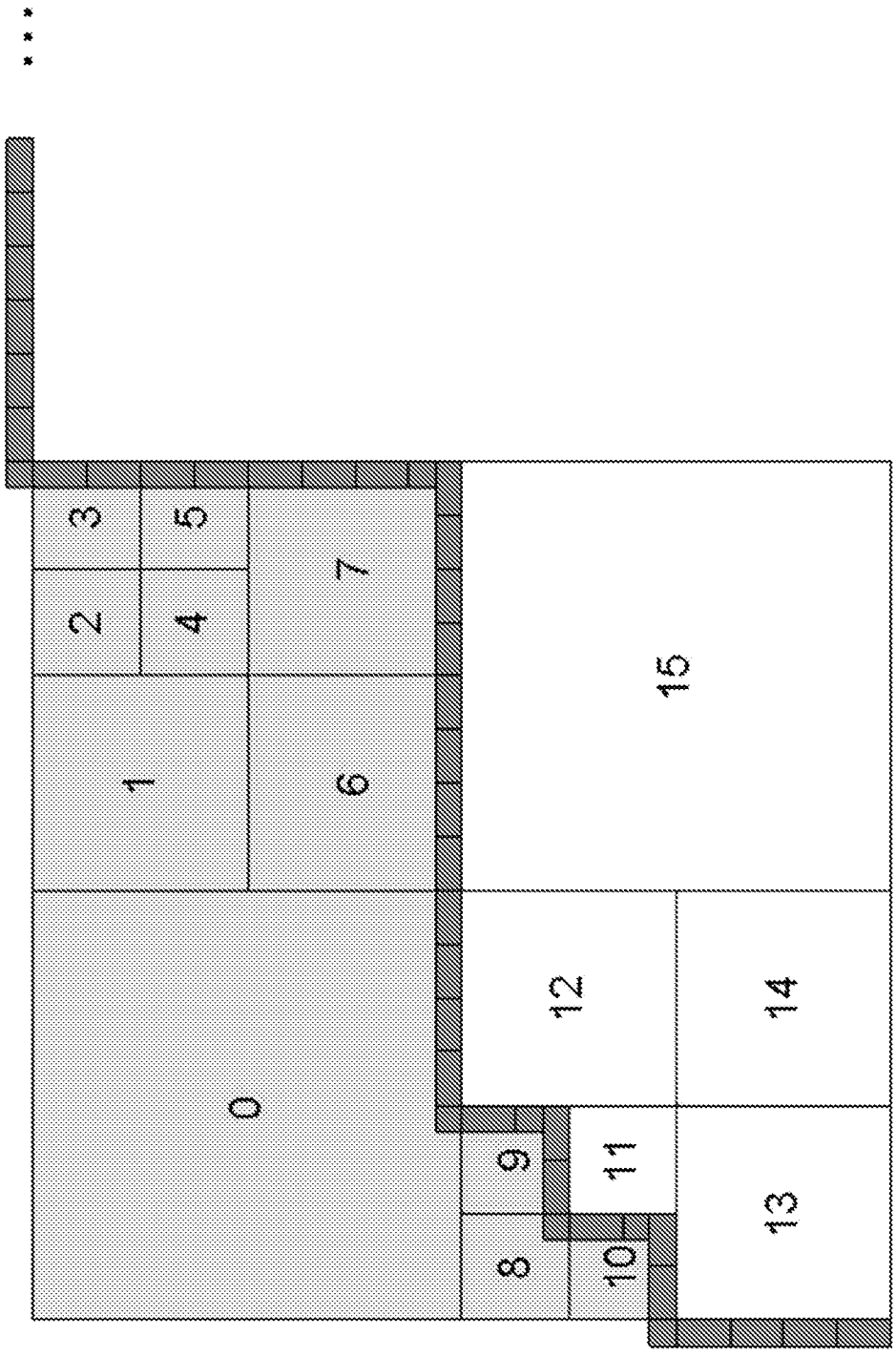

FIG. 9A

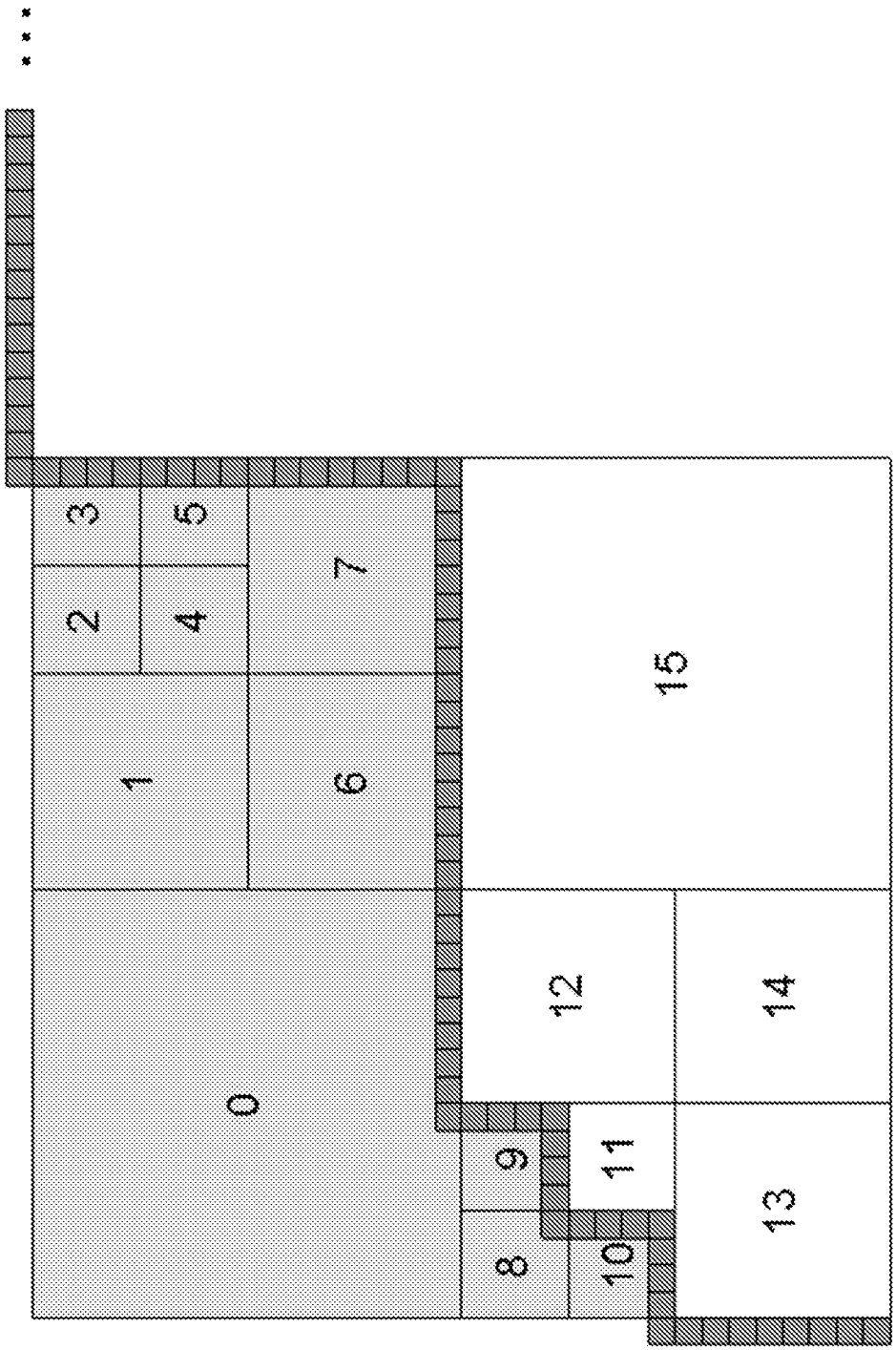

FIG. 9B

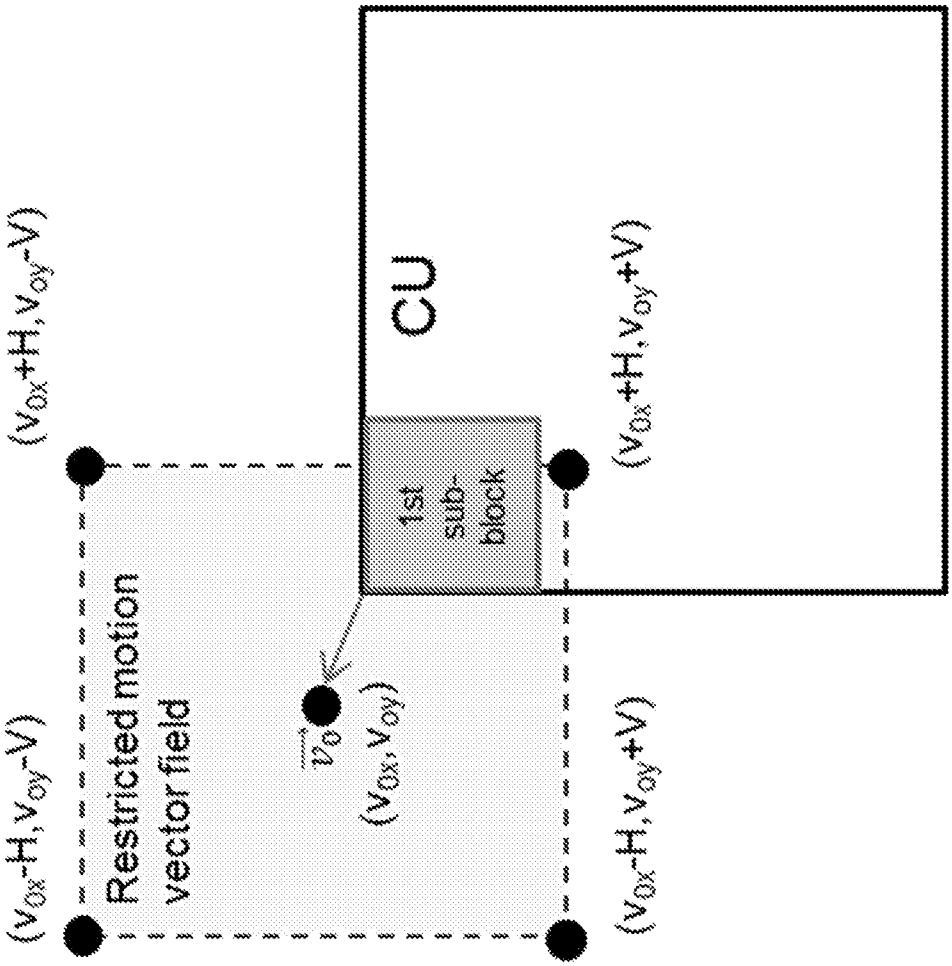
CU
1st sub-block
Restricted motion vector field
$(v_{0x}+H, v_{0y}+V)$
$(v_{0x}-H, v_{0y}+V)$
$(v_{0x}+H, v_{0y}-V)$
$(v_{0x}-H, v_{0y}-V)$
$\vec{v}_0$
(v_{0x}, v_{0y})

FIG. 10

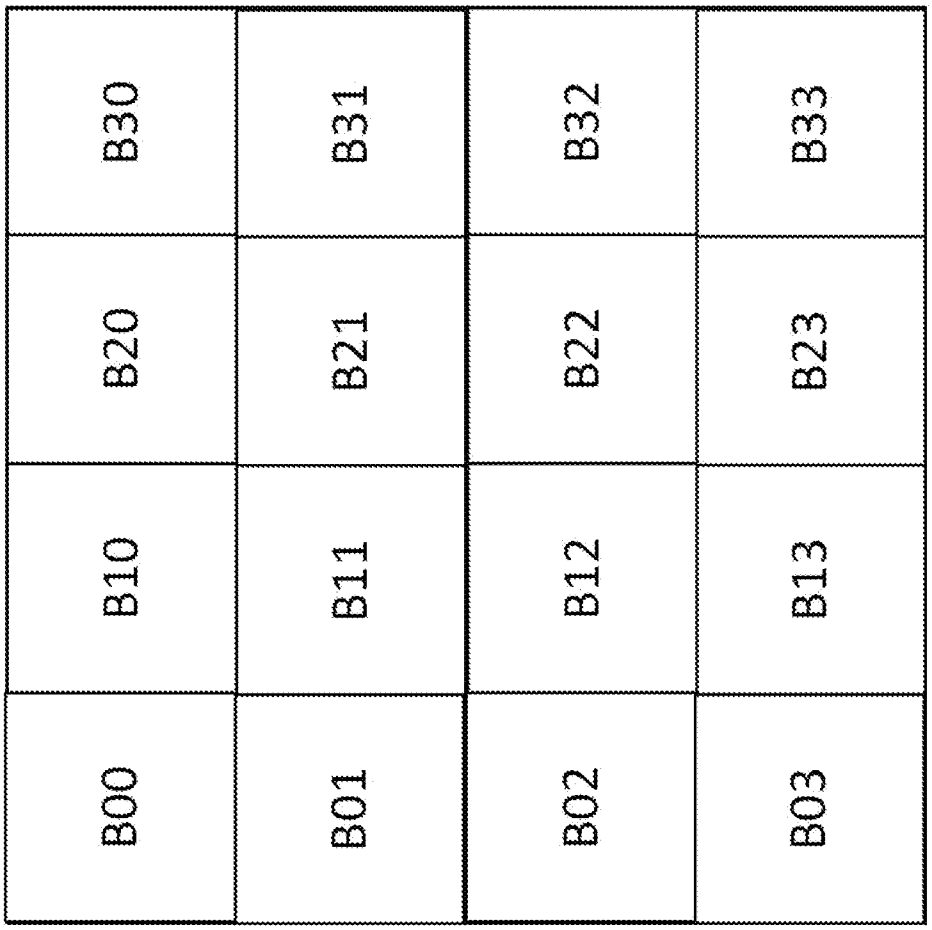

FIG. 11

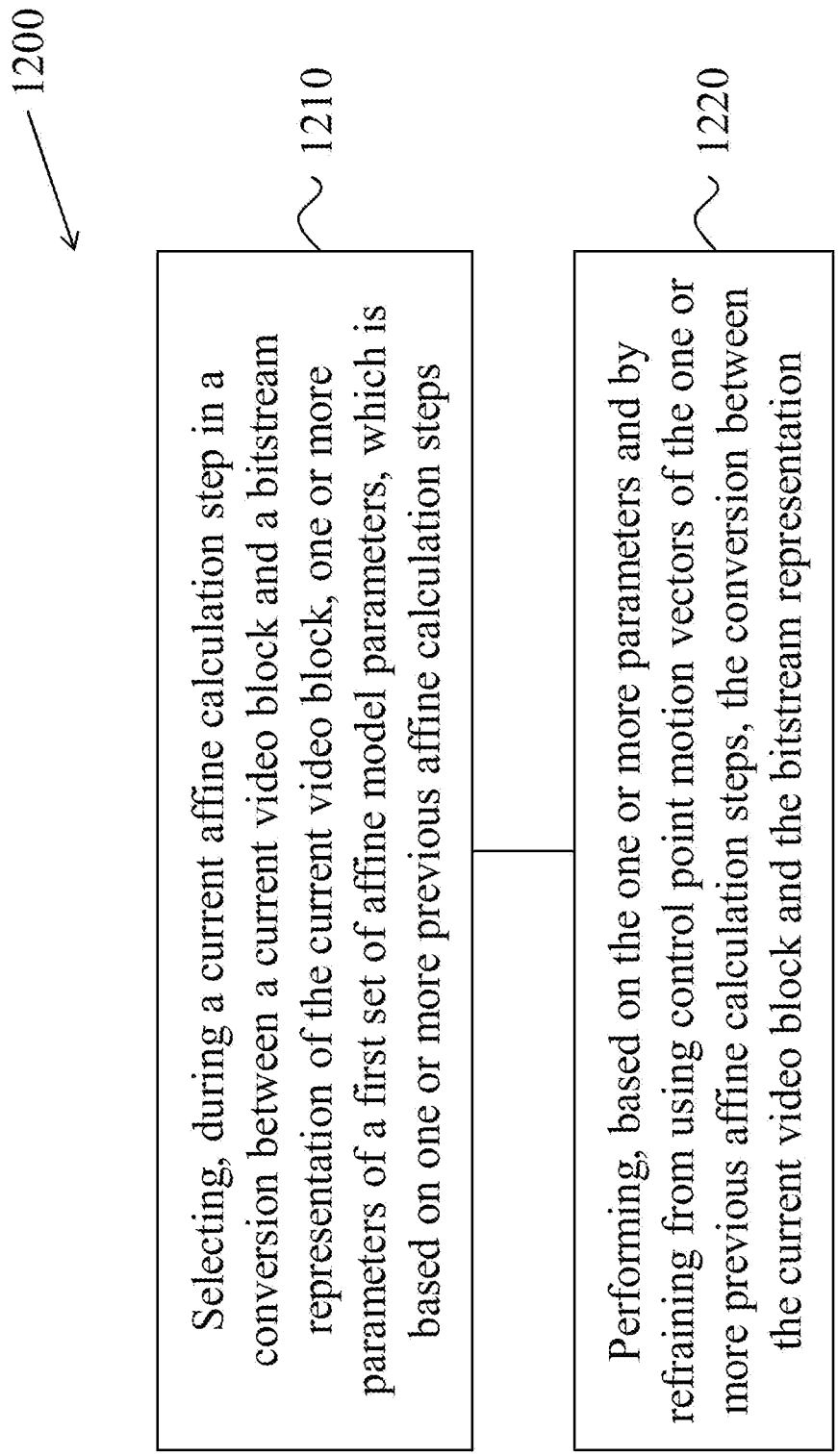
1200
1210
Selecting, during a current affine calculation step in a conversion between a current video block and a bitstream representation of the current video block, one or more parameters of a first set of affine model parameters, which is based on one or more previous affine calculation steps
1220
Performing, based on the one or more parameters and by refraining from using control point motion vectors of the one or more previous affine calculation steps, the conversion between the current video block and the bitstream representation

FIG. 12

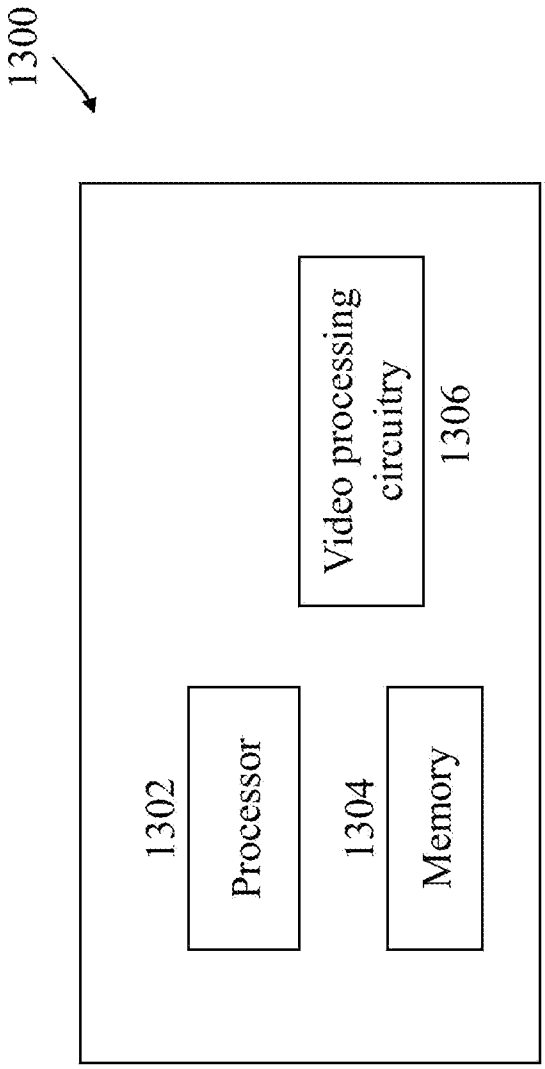
1300
Video processing circuitry
1306
1302
Processor
1304
Memory

FIG. 13

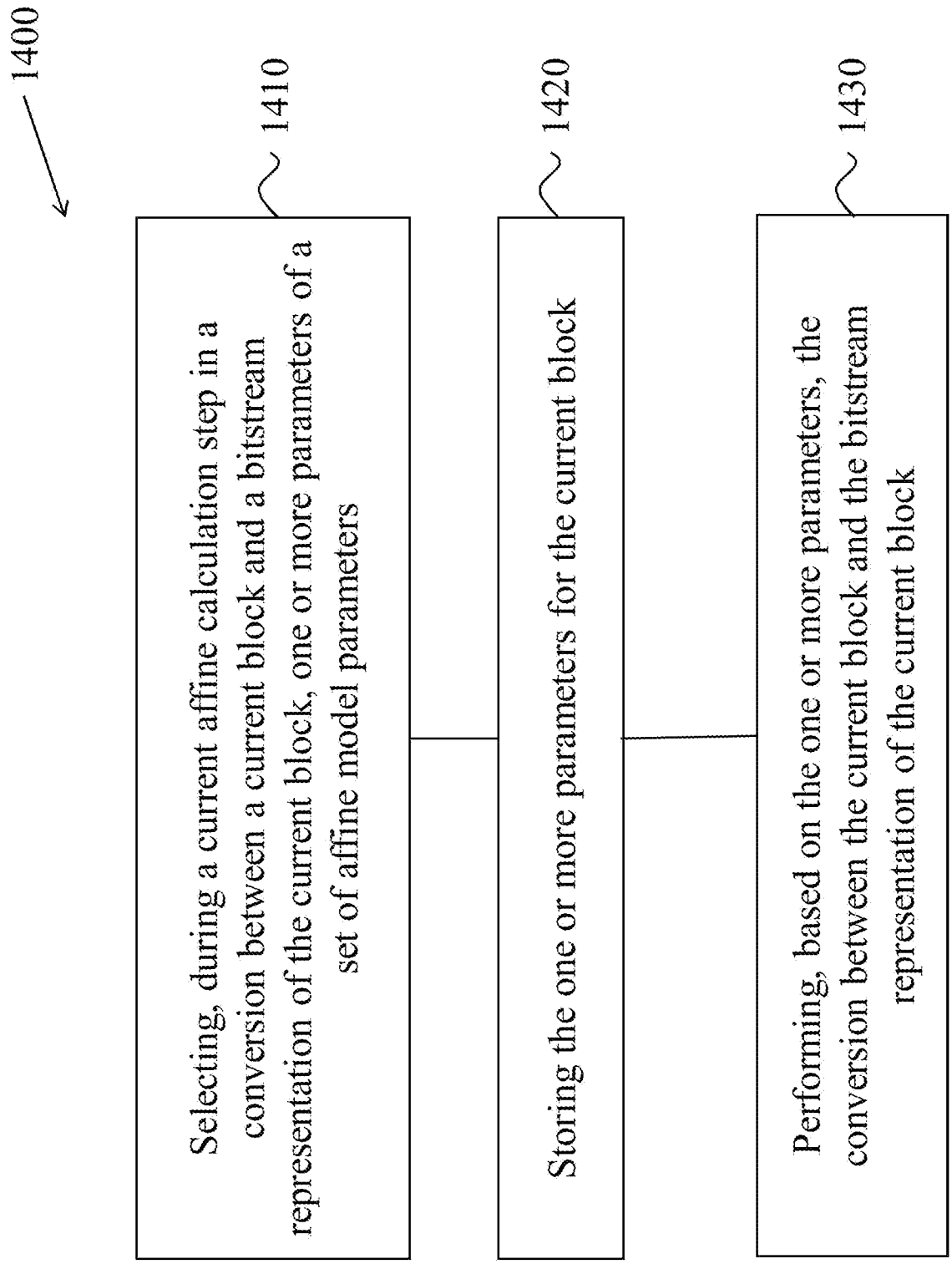
1400
1410
Selecting, during a current affine calculation step in a conversion between a current block and a bitstream representation of the current block, one or more parameters of a set of affine model parameters
1420
Storing the one or more parameters for the current block
1430
Performing, based on the one or more parameters, the conversion between the current block and the bitstream representation of the current block

FIG. 14

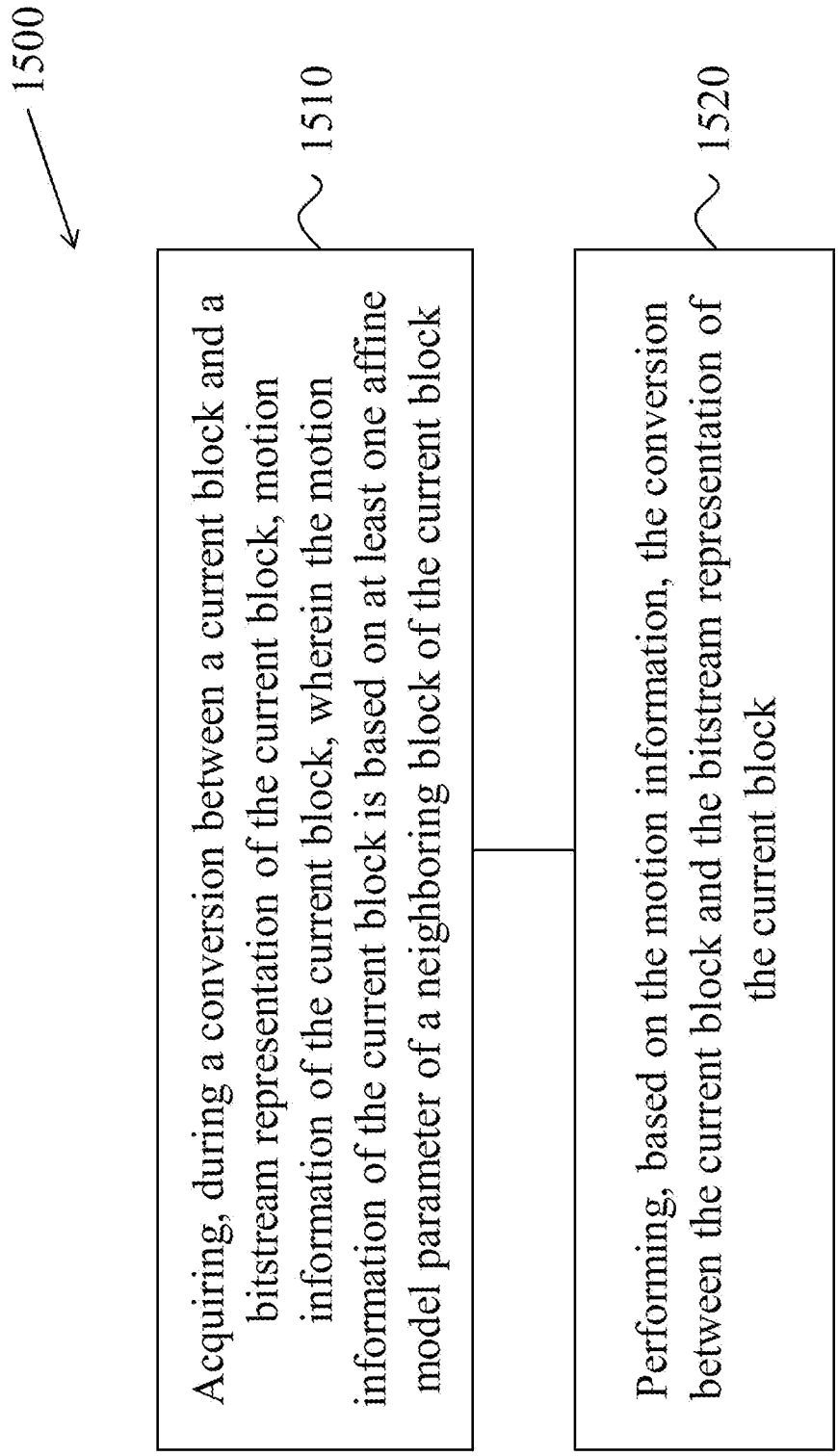
1500
1510
Acquiring, during a conversion between a current block and a bitstream representation of the current block, motion information of the current block, wherein the motion information of the current block is based on at least one affine model parameter of a neighboring block of the current block
1520
Performing, based on the motion information, the conversion between the current block and the bitstream representation of the current block

FIG. 15

INTERNATIONAL SEARCH REPORT

International application No
PCT/IB2019/058991

A. CLASSIFICATION OF SUBJECT MATTER
INV. H04N19/52 H04N19/537 H04N19/119 H04N19/139 H04N19/176
H04N19/54 H04N19/186
ADD.
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

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

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

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	HUANG HAN ET AL: "Control-Point Representation and Differential Coding Affine-Motion Compensation", IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, US, vol. 23, no. 10, 1 October 2013 (2013-10-01), pages 1651-1660, XP011528531, ISSN: 1051-8215, DOI: 10.1109/TCSVT.2013.2254977 [retrieved on 2013-09-30]	1-11, 20-29, 44-47
Y	the whole document	12-19, 30-43
	----- -/--	

[X] Further documents are listed in the continuation of Box C.

[X] See patent family annex.

* Special categories of cited documents :

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

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

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

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

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

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

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

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

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
2 January 2020	23/01/2020

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	McGrath, Simon

2

Form PCT/ISA/210 (second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

International application No
PCT/IB2019/058991

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WO 2017/148345 A1 (MEDIATEK INC [CN]) 8 September 2017 (2017-09-08)	1-11, 20-29, 44-47
Y	paragraphs [0005], [0006], [0030] - [0034]; figures 1A,1B	12-19, 30-43
Y	LI (PANASONIC) J ET AL: "AHG5: Reduction of worst case memory bandwidth", 12. JVET MEETING; 20181003 - 20181012; MACAO; (THE JOINT VIDEO EXPLORATION TEAM OF ISO/IEC JTC1/SC29/WG11 AND ITU-T SG.16), , no. JVET-L0122 5 October 2018 (2018-10-05), XP030194934, Retrieved from the Internet: URL:http://phenix.int-evry.fr/jvet/doc_end _user/documents/12_Macao/wg11/JVET-L0122-v 2.zip JVET-L0122-v2.doc [retrieved on 2018-10-05] the whole document	12-19, 30-43
Y	GAO (TENCENT) M ET AL: "CE4-related: Sub-block MV clipping in affine prediction", 124. MPEG MEETING; 20181008 - 20181012; MACAO; (MOTION PICTURE EXPERT GROUP OR ISO/IEC JTC1/SC29/WG11), , no. m44341 5 October 2018 (2018-10-05), XP030192618, Retrieved from the Internet: URL:http://phenix.int-evry.fr/mpeg/doc_end _user/documents/124_Macao/wg11/m44341-JVET -L0317-v3-JVET-L0317-v3.zip JVET-L0317-r1.docx [retrieved on 2018-10-05] the whole document	12-19, 30-43
A	US 2018/098063 A1 (CHEN YI-WEN [US] ET AL) 5 April 2018 (2018-04-05) paragraphs [0102] - [0110]; figures 4-11	1-47

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/IB2019/058991

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 2017148345 A1	08-09-2017	BR 112018067475 A2	02-01-2019
		CN 108605137 A	28-09-2018
		EP 3414905 A1	19-12-2018
		TW 201803351 A	16-01-2018
		US 2019058896 A1	21-02-2019
		WO 2017147765 A1	08-09-2017
		WO 2017148345 A1	08-09-2017
US 2018098063 A1	05-04-2018	AU 2017340631 A1	14-03-2019
		BR 112019006838 A2	25-06-2019
		CA 3035587 A1	12-04-2018
		CN 109792533 A	21-05-2019
		EP 3523972 A1	14-08-2019
		JP 2019535192 A	05-12-2019
		KR 20190058503 A	29-05-2019
		PH 12019500417 A1	03-06-2019
		SG 112019016320 A	29-04-2019
		TW 201817237 A	01-05-2018
		US 2018098063 A1	05-04-2018
		WO 2018067823 A1	12-04-2018