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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
14 May 2020 (14.05.2020)

WIPO | PCT

(10) International Publication Number
WO 2020/094151 A1

(51) International Patent Classification:
H04N 19/10 (2014.01)

(21) International Application Number:
PCT/CN2019/117119

(22) International Filing Date:
11 November 2019 (11.11.2019)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
PCT/CN2018/114931
10 November 2018 (10.11.2018) 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,

(54) Title: ROUNDING IN PAIRWISE AVERAGE CANDIDATE CALCULATIONS

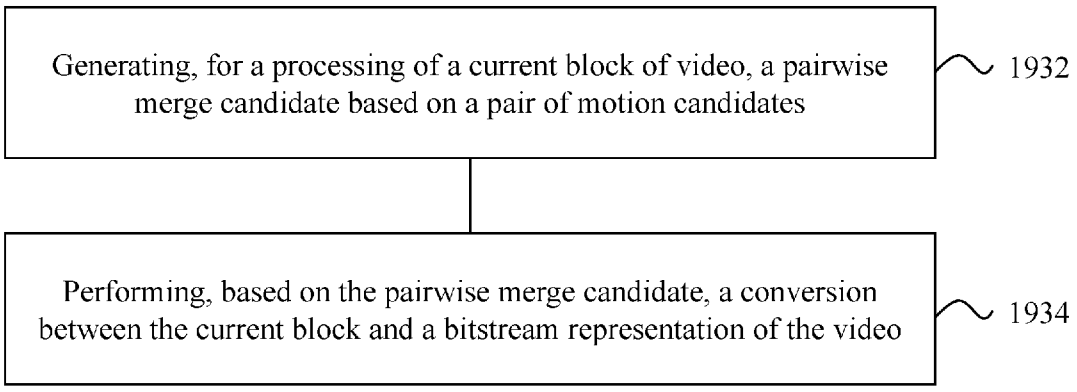

FIG. 19A

(57) Abstract: Devices, systems and methods for unified rounding in sub-block based prediction are described. In a representative aspect, a method of video processing includes generating, for a processing of a current block of video, a pairwise merge candidate based on a pair of motion candidates, and performing, based on the pairwise merge candidate, a conversion between the current block and a bitstream representation of the video. In another representative aspect, a method of video processing includes generating, for a current block of video coded using a geometry partition mode, a uni-prediction motion candidate based on a scaled motion vector and a List0 motion vector, and performing, based on the uni-prediction motion candidate, a conversion between the current block and a bitstream representation of the video.

WO 2020/094151 A1

[Continued on next page]

DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

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

Published:

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

ROUNDING IN PAIRWISE AVERAGE CANDIDATE CALCULATIONS

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/CN2018/114931, filed on November 10, 2018. For all purposes under the U.S. law, the entire disclosure of the aforementioned application is incorporated by reference as part of the disclosure of this application.

TECHNICAL FIELD

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

BACKGROUND

[0003] Motion compensation is a technique in video processing to predict a frame in a video, given the previous and/or future frames by accounting for motion of the camera and/or objects in the video. Motion compensation can be used in the encoding and decoding of video data for video compression.

SUMMARY

[0004] Devices, systems and methods related to unified rounding methods for sub-block based prediction for image and video coding are described.

[0005] In one representative aspect, the disclosed technology may be used to provide a method of video processing.

This method includes generating, for a processing of a current block of video, a pairwise merge candidate based on a pair of motion candidates; and performing, based on the pairwise merge candidate, a conversion between the current block and a bitstream representation of the video, wherein the pair of motion candidates comprises a first motion vector MV0 = (MV0x, MV0y) of a reference picture list L in a first motion candidate and a second motion vector MV1 = (MV1x, MV1y) of the reference picture list L in a second motion candidate, and wherein MV* = (MV*x, MV*y) is the pairwise merge candidate such that:

$$MV^*x = \text{Shift}(MV0x + MV1x, 1) \text{ and } MV^*y = \text{Shift}(MV0y + MV1y, 1),$$

wherein $\text{Shift}(x, s) = (x + off) \gg s$, wherein *off* and *s* are integers, and wherein >> represents a right shift operation.

[0006] In another representative aspect, the disclosed technology may be used to provide a method of video processing.

This method includes generating, for a processing of a current block of video, a pairwise merge candidate based on a pair of motion candidates; and performing, based on the pairwise merge candidate, a conversion between the current block and a bitstream representation of the video, wherein the pair of motion candidates comprises a first motion vector MV0 = (MV0x, MV0y) of a reference picture list L in a first motion candidate and a second motion vector MV1 = (MV1x, MV1y) of the reference picture list L in a second motion candidate, and wherein MV* = (MV*x, MV*y) is the pairwise merge candidate such that:

$$MV^*x = \text{SignShift}(MV0x + MV1x, 1) \text{ and } MV^*y = \text{SignShift}(MV0y + MV1y, 1),$$

wherein

$$\text{SignShift}(x, s) = \begin{cases} (x + off) \gg s & x \geq 0 \\ -((-x + off) \gg s) & x < 0 \end{cases},$$

wherein *off* and *s* are integers, and wherein >> represents a shift operation.

[0007] In yet another representative aspect, the disclosed technology may be used to provide a method of video processing.

This method includes generating, for a processing of a current block of video, a pairwise merge candidate based on a pair of motion candidates; and performing, based on the pairwise merge candidate, a conversion between the current block and a bitstream representation of the video, wherein the pair of motion candidates comprises a first motion vector MV0 = (MV0x, MV0y) of a reference picture list L in a first motion candidate and a second motion vector MV1 = (MV1x, MV1y) of the reference picture list L in a second motion candidate, and wherein MV* = (MV*x, MV*y) is the pairwise merge candidate such that:

$$MV^*x = \text{SignShift}(MV0x + MV1x, 1) \text{ and } MV^*y = \text{SignShift}(MV0y + MV1y, 1),$$

wherein

$$\text{SignShift}(x, s) = \begin{cases} (x + off) \gg s & x \geq 0 \\ (x + off + 1) \gg s & x < 0 \end{cases},$$

wherein *off* and *s* are integers, and wherein >> represents a shift operation.

[0008] In yet another representative aspect, the disclosed technology may be used to provide

a method of video processing.

This method includes generating, for a processing of a current block of video, a pairwise merge candidate, wherein the generating is based on a pair of motion candidates that refers to a current picture comprising the current block; and performing, based on the pairwise merge candidate, a conversion between the current block and a bitstream representation of the video, wherein the pair of motion candidates comprises a first motion vector MV0 = (MV0x, MV0y) of a reference picture list L in a first motion candidate and a second motion vector MV1 = (MV1x, MV1y) of the reference picture list L in a second motion candidate, and wherein MV* = (MV*x, MV*y) is the pairwise merge candidate such that:

$$MV^*x = \text{Shift}(MV0x + MV1x, 1) \text{ and } MV^*y = \text{Shift}(MV0y + MV1y, 1),$$

wherein $\text{Shift}(x, s) = (x + off) \gg s$, wherein *off* and *s* are integers, and wherein >> represents a right shift operation.

[0009] In yet another representative aspect, the disclosed technology may be used to provide a method of video processing.

This method includes generating, for a processing of a current block of video, a pairwise merge candidate, wherein the generating is based on a pair of motion candidates that refers to a current picture comprising the current block; and performing, based on the pairwise merge candidate, a conversion between the current block and a bitstream representation of the video, wherein the pair of motion candidates comprises a first motion vector MV0 = (MV0x, MV0y) of a reference picture list L in a first motion candidate and a second motion vector MV1 = (MV1x, MV1y) of the reference picture list L in a second motion candidate, and wherein MV* = (MV*x, MV*y) is the pairwise merge candidate such that:

$$MV^*x = \text{SignShift}(MV0x + MV1x, 1) \text{ and } MV^*y = \text{SignShift}(MV0y + MV1y, 1),$$

wherein

$$\text{SignShift}(x, s) = \begin{cases} (x + off) \gg s & x \geq 0 \\ -((-x + off) \gg s) & x < 0 \end{cases},$$

wherein *off* and *s* are integers, and wherein >> represents a shift operation.

[0010] In yet another representative aspect, the disclosed technology may be used to provide a method of video processing.

This method includes generating, for a processing of a current block of video, a pairwise merge candidate, wherein the generating is based on a pair of motion candidates that refers to a current

picture comprising the current block; and performing, based on the pairwise merge candidate, a conversion between the current block and a bitstream representation of the video, wherein the pair of motion candidates comprises a first motion vector MV0 = (MV0x, MV0y) of a reference picture list L in a first motion candidate and a second motion vector MV1 = (MV1x, MV1y) of the reference picture list L in a second motion candidate, and wherein MV* = (MV*x, MV*y) is the pairwise merge candidate such that:

$MV^*x = \text{SignShift}(MV0x + MV1x, 1)$ and $MV^*y = \text{SignShift}(MV0y + MV1y, 1)$,

wherein

$$\text{SignShift}(x, s) = \begin{cases} (x + off) \gg s & x \geq 0 \\ (x + off + 1) \gg s & x < 0 \end{cases},$$

wherein off and s are integers, and wherein >> represents a shift operation.

[0011] In yet another representative aspect, the disclosed technology may be used to provide a method of video processing.

[0012] This method includes generating, for a processing of a current block of video, a pairwise merge candidate, wherein the generating is based on a pair of motion candidates that refers to a current picture comprising the current block; and performing, based on the pairwise merge candidate, a conversion between the current block and a bitstream representation of the video, wherein the pair of motion candidates comprises a first motion vector MV0 = (MV0x, MV0y) of a reference picture list L in a first motion candidate and a second motion vector MV1 = (MV1x, MV1y) of the reference picture list L in a second motion candidate, and wherein MV* = (MV*x, MV*y) is the pairwise merge candidate such that:

[0013] $MV^*x = (\text{Shift}(MV0x + MV1x, (W + 1))) \ll W$ and

[0014] $MV^*y = (\text{Shift}(MV0y + MV1y, (W + 1))) \ll W$,

wherein $\text{Shift}(x, s) = (x + off) \gg s$, wherein W, off and s are integers, and wherein >> represents a right shift operation.

[0015] In yet another representative aspect, the disclosed technology may be used to provide a method of video processing.

This method includes generating, for a processing of a current block of video, a pairwise merge candidate, wherein the generating is based on a pair of motion candidates that refers to a current picture comprising the current block; and performing, based on the pairwise merge candidate, a conversion between the current block and a bitstream representation of the video, wherein the

pair of motion candidates comprises a first motion vector MV0 = (MV0x, MV0y) of a reference picture list L in a first motion candidate and a second motion vector MV1 = (MV1x, MV1y) of the reference picture list L in a second motion candidate, and wherein MV* = (MV*x, MV*y) is the pairwise merge candidate such that:

$$MV^*x = (\text{SignShift}(MV0x + MV1x, (W + 1))) \ll \text{W} \text{ and}$$

$$MV^*y = (\text{SignShift}(MV0y + MV1y, (W + 1))) \ll \text{W},$$

wherein

$$\text{SignShift}(x, s) = \begin{cases} (x + off) \gg s & x \geq 0 \\ -((-x + off) \gg s) & x < 0 \end{cases},$$

wherein *W*, *off* and *s* are integers, and wherein >> represents a shift operation.

[0016] In yet another representative aspect, the disclosed technology may be used to provide a method of video processing.

This method includes generating, for a processing of a current block of video, a pairwise merge candidate, wherein the generating is based on a pair of motion candidates that refers to a current picture comprising the current block; and performing, based on the pairwise merge candidate, a conversion between the current block and a bitstream representation of the video, wherein the pair of motion candidates comprises a first motion vector MV0 = (MV0x, MV0y) of a reference picture list L in a first motion candidate and a second motion vector MV1 = (MV1x, MV1y) of the reference picture list L in a second motion candidate, and wherein MV* = (MV*x, MV*y) is the pairwise merge candidate such that:

$$MV^*x = (\text{SignShift}(MV0x + MV1x, (W + 1))) \ll \text{W} \text{ and}$$

$$MV^*y = (\text{SignShift}(MV0y + MV1y, (W + 1))) \ll \text{W},$$

wherein

$$\text{SignShift}(x, s) = \begin{cases} (x + off) \gg s & x \geq 0 \\ (x + off + 1) \gg s & x < 0 \end{cases},$$

wherein *W*, *off* and *s* are integers, and wherein >> represents a shift operation.

[0017] In yet another representative aspect, the disclosed technology may be used to provide a method of video processing.

This method includes generating, for a current block of video coded using a geometry partition mode, a uni-prediction motion candidate based on a scaled motion vector and a List0 motion vector; and performing, based on the uni-prediction motion candidate, a conversion between the

current block and a bitstream representation of the video, wherein the scaled motion vector is MV1' = (MV1'x, MV1'y) and the List0 motion vector is MV0 = (MV0x, MV0y), and wherein MV* = (MV*x, MV*y) is the uni-prediction motion candidate such that:

$$MV^*x = \text{Shift}(MV0x + MV1'x, 1) \text{ and } MV^*y = \text{Shift}(MV0y + MV1'y, 1),$$

wherein $\text{Shift}(x, s) = (x + off) \gg s$, wherein *off* and *s* are integers, and wherein >> represents a right shift operation.

[0018] In yet another representative aspect, the disclosed technology may be used to provide a method of video processing.

This method includes generating, for a current block of video coded using a geometry partition mode, a uni-prediction motion candidate based on a scaled motion vector and a List0 motion vector; and performing, based on the uni-prediction motion candidate, a conversion between the current block and a bitstream representation of the video, wherein the scaled motion vector is MV1' = (MV1'x, MV1'y) and the List0 motion vector is MV0 = (MV0x, MV0y), and wherein MV* = (MV*x, MV*y) is the uni-prediction motion candidate such that:

$$MV^*x = \text{SignShift}(MV0x + MV1'x, 1) \text{ and } MV^*y = \text{SignShift}(MV0y + MV1'y, 1),$$

wherein

$$\text{SignShift}(x, s) = \begin{cases} (x + off) \gg s & x \geq 0 \\ -((-x + off) \gg s) & x < 0 \end{cases},$$

wherein *off* and *s* are integers, and wherein >> represents a shift operation.

[0019] In yet another representative aspect, the disclosed technology may be used to provide a method of video processing.

This method includes generating, for a current block of video coded using a geometry partition mode, a uni-prediction motion candidate based on a scaled motion vector and a List0 motion vector; and performing, based on the uni-prediction motion candidate, a conversion between the current block and a bitstream representation of the video, wherein the scaled motion vector is MV1' = (MV1'x, MV1'y) and the List0 motion vector is MV0 = (MV0x, MV0y), and wherein MV* = (MV*x, MV*y) is the uni-prediction motion candidate such that:

$$MV^*x = \text{SignShift}(MV0x + MV1'x, 1) \text{ and } MV^*y = \text{SignShift}(MV0y + MV1'y, 1),$$

wherein

$$\text{SignShift}(x, s) = \begin{cases} (x + off) \gg s & x \geq 0 \\ (x + off + 1) \gg s & x < 0 \end{cases},$$

wherein *off* and *s* are integers, and wherein >> represents a shift operation.

[0020] 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.

[0021] 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.

[0022] In yet another representative aspect, a video decoder apparatus may implement a method as described herein.

[0023] 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

[0024] FIG. 1 shows an example of sub-block based prediction.

[0025] FIG. 2 shows an example of a simplified affine motion model.

[0026] FIG. 3 shows an example of an affine motion vector field (MVF) per sub-block.

[0027] FIG. 4 shows an example of motion vector prediction (MVP) for the AF_INTER affine motion mode.

[0028] FIGS. 5A and 5B show example candidates for the AF_MERGE affine motion mode.

[0029] FIG. 6 shows an example of motion prediction using the alternative temporal motion vector prediction (ATMVP) algorithm for a coding unit (CU).

[0030] FIG. 7 shows an example of a coding unit (CU) with sub-blocks and neighboring blocks used by the spatial-temporal motion vector prediction (STMVP) algorithm.

[0031] FIG. 8 shows an example of an optical flow trajectory used by the bi-directional optical flow (BIO) algorithm.

[0032] FIGS. 9A and 9B show example snapshots of using of the bi-directional optical flow (BIO) algorithm without block extensions.

[0033] FIG. 10 shows an example of bilateral matching in the frame-rate up conversion (FRUC) algorithm.

[0034] FIG. 11 shows an example of template matching in the FRUC algorithm.

[0035] FIG. 12 shows an example of unilateral motion estimation in the FRUC algorithm.

[0036] FIG. 13 shows an example of sub-blocks for different components with the 4:2:0

format in JEM.

[0037] FIG. 14 shows an example of sub-blocks for different components with the 4:2:0 format in accordance with the disclosed technology.

[0038] FIG. 15 shows another example of sub-blocks for different components with the 4:2:0 format in accordance with the disclosed technology.

[0039] FIG. 16 shows an example of sub-block motion vector (MV) copying in accordance with the disclosed technology.

[0040] FIG. 17 shows another example of sub-block motion vector (MV) copying in accordance with the disclosed technology.

[0041] FIG. 18 shows an example of sub-block motion vector (MV) derivation.

[0042] FIGS. 19A–19C show flowcharts of example methods of video processing.

[0043] FIG. 20 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.

[0044] FIG. 21 is a block diagram of an example video processing system in which disclosed techniques may be implemented.

DETAILED DESCRIPTION

[0045] 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.

[0046] 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.

[0047] 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)

[0048] 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), Locally Adaptive Motion Vector Resolution (LAMVR), Overlapped Block Motion Compensation (OBMC), Local Illumination Compensation (LIC), and Decoder-side Motion Vector Refinement (DMVR).

1.1 Examples of affine prediction

[0049] In HEVC, only a translation motion model is applied for motion compensation prediction (MCP). However, the camera and objects may have many kinds of motion, e.g. zoom in/out, rotation, perspective motions, and/or other irregular motions. JEM, on the other hand, applies a simplified affine transform motion compensation prediction. FIG. 2 shows an example of an affine motion field of a block 200 described by two control point motion vectors V_0 and V_1. The motion vector field (MVF) of the block 200 can be described by the following equation:

[0050]
$$\begin{cases} v_x = \dfrac{(v_{1x} - v_{0x})}{w} x - \dfrac{(v_{1y} - v_{0y})}{w} y + v_{0x} \\ v_y = \dfrac{(v_{1y} - v_{0y})}{w} x + \dfrac{(v_{1x} - v_{0x})}{w} y + v_{0y} \end{cases} \quad \text{Eq. (1)}$$

[0051] As shown in FIG. 2, (v_{0x}, v_{0y}) is motion vector of the top-left corner control point, and (v_{1x}, v_{1y}) is motion vector of the top-right corner control point. To simplify the motion compensation prediction, sub-block based affine transform prediction can be applied. The sub-block size M×N is derived as follows:

[0052] $$\begin{cases} M = clip3(4, w, \frac{w \times MvPre}{\max\left(abs(v_{1x}-v_{0x}), abs(v_{1y}-v_{0y})\right)}) \\ N = clip3(4, h, \frac{h \times MvPre}{\max\left(abs(v_{2x}-v_{0x}), abs(v_{2y}-v_{0y})\right)}) \end{cases} \quad \text{Eq. (2)}$$

[0053] Here, *MvPre* is the motion vector fraction accuracy (e.g., 1/16 in JEM). (v_{2x}, v_{2y}) is motion vector of the bottom-left control point, calculated according to Eq. (1). M and N can be adjusted downward if necessary to make it a divisor of w and h, respectively.

[0054] FIG. 3 shows an example of affine MVF per sub-block for a block 300. To derive motion vector of each M×N sub-block, the motion vector of the center sample of each sub-block can be calculated according to Eq. (1), and rounded to the motion vector fraction accuracy (e.g., 1/16 in JEM). Then the motion compensation interpolation filters can be applied to generate the prediction of each sub-block with derived motion vector. After the MCP, the high accuracy motion vector of each sub-block is rounded and saved as the same accuracy as the normal motion vector.

[0055] In the JEM, there are two affine motion modes: AF_INTER mode and AF_MERGE mode. For CUs with both width and height larger than 8, AF_INTER mode can be applied. An affine flag in CU level is signaled in the bitstream to indicate whether AF_INTER mode is used. In the AF_INTER mode, a candidate list with motion vector pair $\{(v_0, v_1)|v_0 = \{v_A, v_B, v_c\}, v_1 = \{v_D, v_E\}\}$ is constructed using the neighboring blocks.

[0056] FIG. 4 shows an example of motion vector prediction (MVP) for a block 400 in the AF_INTER mode. As shown in FIG. 4, v_0 is selected from the motion vectors of the sub-block A, B, or C. The motion vectors from the neighboring blocks can be scaled according to the reference list. The motion vectors can also be scaled according to the relationship among the Picture Order Count (POC) of the reference for the neighboring block, the POC of the reference for the current CU, and the POC of the current CU. The approach to select v_1 from the neighboring sub-block D and E is similar. If the number of candidate list is smaller than 2, the list is padded by the motion vector pair composed by duplicating each of the AMVP candidates. When the candidate list is larger than 2, the candidates can be firstly sorted according to the neighboring motion vectors (e.g., based on the similarity of the two motion vectors in a pair candidate). In some implementations, the first two candidates are kept. In some embodiments, a Rate Distortion (RD) cost check is used to determine which motion vector pair candidate is

selected as the control point motion vector prediction (CPMVP) of the current CU. An index indicating the position of the CPMVP in the candidate list can be signaled in the bitstream. After the CPMVP of the current affine CU is determined, affine motion estimation is applied and the control point motion vector (CPMV) is found. Then the difference of the CPMV and the CPMVP is signaled in the bitstream.

[0057] When a CU is applied in AF_MERGE mode, it gets the first block coded with an affine mode from the valid neighboring reconstructed blocks. FIG. 5A shows an example of the selection order of candidate blocks for a current CU 500. As shown in FIG. 5A, the selection order can be from left (501), above (502), above right (503), left bottom (504) to above left (505) of the current CU 500. FIG. 5B shows another example of candidate blocks for a current CU 500 in the AF_MERGE mode. If the neighboring left bottom block 501 is coded in affine mode, as shown in FIG. 5B, the motion vectors v_2, v_3 and v_4 of the top left corner, above right corner, and left bottom corner of the CU containing the sub-block 501 are derived. The motion vector v_0 of the top left corner on the current CU 500 is calculated based on v2, v3 and v4. The motion vector v1 of the above right of the current CU can be calculated accordingly.

[0058] After the CPMV of the current CU v0 and v1 are computed according to the affine motion model in Eq. (1), the MVF of the current CU can be generated. In order to identify whether the current CU is coded with AF_MERGE mode, an affine flag can be signaled in the bitstream when there is at least one neighboring block is coded in affine mode.

[0059] In JEM, the non-merge affine mode can be used only when the width and the height of the current block are both larger than 8; the merge affine mode can be used only when the area (i.e. width × height) of the current block is not smaller than 64.

1.2 Examples of alternative temporal motion vector prediction (ATMVP)

[0060] In the ATMVP method, the temporal motion vector prediction (TMVP) method is modified by fetching multiple sets of motion information (including motion vectors and reference indices) from blocks smaller than the current CU.

[0061] FIG. 6 shows an example of ATMVP motion prediction process for a CU 600. The ATMVP method predicts the motion vectors of the sub-CUs 601 within a CU 600 in two steps. The first step is to identify the corresponding block 651 in a reference picture 650 with a temporal vector. The reference picture 650 is also referred to as the motion source picture. The second step is to split the current CU 600 into sub-CUs 601 and obtain the motion vectors as well

as the reference indices of each sub-CU from the block corresponding to each sub-CU.

[0062] In the first step, a reference picture 650 and the corresponding block is determined by the motion information of the spatial neighboring blocks of the current CU 600. To avoid the repetitive scanning process of neighboring blocks, the first merge candidate in the merge candidate list of the current CU 600 is used. The first available motion vector as well as its associated reference index are set to be the temporal vector and the index to the motion source picture. This way, the corresponding block may be more accurately identified, compared with TMVP, wherein the corresponding block (sometimes called collocated block) is always in a bottom-right or center position relative to the current CU.

[0063] In the second step, a corresponding block of the sub-CU 651 is identified by the temporal vector in the motion source picture 650, by adding to the coordinate of the current CU the temporal vector. For each sub-CU, the motion information of its corresponding block (e.g., the smallest motion grid that covers the center sample) is used to derive the motion information for the sub-CU. After the motion information of a corresponding N×N block is identified, it is converted to the motion vectors and reference indices of the current sub-CU, in the same way as TMVP of HEVC, wherein motion scaling and other procedures apply. For example, the decoder checks whether the low-delay condition (e.g. the POCs of all reference pictures of the current picture are smaller than the POC of the current picture) is fulfilled and possibly uses motion vector MVx (e.g., the motion vector corresponding to reference picture list X) to predict motion vector MVy (e.g., with X being equal to 0 or 1 and Y being equal to 1−X) for each sub-CU.

1.3 Examples of spatial-temporal motion vector prediction (STMVP)

[0064] In the STMVP method, the motion vectors of the sub-CUs are derived recursively, following raster scan order. FIG. 7 shows an example of one CU with four sub-blocks and neighboring blocks. Consider an 8×8 CU 700 that includes four 4×4 sub-CUs A (701), B (702), C (703), and D (704). The neighboring 4×4 blocks in the current frame are labelled as a (711), b (712), c (713), and d (714).

[0065] The motion derivation for sub-CU A starts by identifying its two spatial neighbors. The first neighbor is the N×N block above sub-CU A 701 (block c 713). If this block c (713) is not available or is intra coded the other N×N blocks above sub-CU A (701) are checked (from left to right, starting at block c 713). The second neighbor is a block to the left of the sub-CU A 701 (block b 712). If block b (712) is not available or is intra coded other blocks to the left of

sub-CU A 701 are checked (from top to bottom, staring at block b 712). The motion information obtained from the neighboring blocks for each list is scaled to the first reference frame for a given list. Next, temporal motion vector predictor (TMVP) of sub-block A 701 is derived by following the same procedure of TMVP derivation as specified in HEVC. The motion information of the collocated block at block D 704 is fetched and scaled accordingly. Finally, after retrieving and scaling the motion information, all available motion vectors are averaged separately for each reference list. The averaged motion vector is assigned as the motion vector of the current sub-CU.

1.4 Examples of bi-directional optical flow (BIO)

[0066] The bi-directional optical flow (BIO) method is a sample-wise motion refinement performed on top of block-wise motion compensation for bi-prediction. In some implementations, the sample-level motion refinement does not use signaling.

[0067] Let $I^{(k)}$ be the luma value from reference k (k=0, 1) after block motion compensation, and $\partial I^{(k)}/\partial x$, $\partial I^{(k)}/\partial y$ are horizontal and vertical components of the $I^{(k)}$ gradient, respectively. Assuming the optical flow is valid, the motion vector field (v_x, v_y) is given by:

[0068] $$\partial I^{(k)}/\partial t + v_x \partial I^{(k)}/\partial x + v_y \partial I^{(k)}/\partial y = 0. \quad \text{Eq. (3)}$$

[0069] Combining this optical flow equation with Hermite interpolation for the motion trajectory of each sample results in a unique third-order polynomial that matches both the function values $I^{(k)}$ and $\partial I^{(k)}/\partial x$, $\partial I^{(k)}/\partial y$ derivatives at the ends. The value of this polynomial at t=0 is the BIO prediction:

[0070] $$pred_{BIO} = 1/2 \cdot \left(I^{(0)} + I^{(1)} + v_x/2 \cdot \left(\tau_1 \partial I^{(1)}/\partial x - \tau_0 \partial I^{(0)}/\partial x\right) + v_y/2 \cdot \left(\tau_1 \partial I^{(1)}/\partial y - \tau_0 \partial I^{(0)}/\partial y\right)\right). \quad \text{Eq. (4)}$$

[0071] FIG. 8 shows an example optical flow trajectory in the Bi-directional Optical flow (BIO) method. Here, τ_0 and τ_1 denote the distances to the reference frames. Distances τ_0 and τ_1 are calculated based on POC for Ref_0 and Ref_1: τ_0=POC(current) − POC(Ref_0), τ_1= POC(Ref_1) − POC(current). If both predictions come from the same time direction (either both from the past or both from the future) then the signs are different (e.g., $\tau_0 \cdot \tau_1 < 0$). In this case, BIO is applied if the prediction is not from the same time moment (e.g., $\tau_0 \neq \tau_1$). Both referenced regions have non-zero motion (e.g., $MVx_0, MVy_0, MVx_1, MVy_1 \neq 0$) and the block motion vectors are proportional to the time distance (e.g., $MVx_0/MVx_1 = MVy_0/MVy_1 = -\tau_0/\tau_1$).

[0072] The motion vector field (v_x, v_y) is determined by minimizing the difference Δ between values in points A and B. FIGS. 9A-9B show an example of intersection of motion trajectory and reference frame planes. Model uses only first linear term of a local Taylor expansion for Δ:

[0073] $$\Delta = \left(I^{(0)} - I^{(1)}{}_0 + v_x\left(\tau_1 \partial I^{(1)}/\partial x + \tau_0 \partial I^{(0)}/\partial x\right) + v_y\left(\tau_1 \partial I^{(1)}/\partial y + \tau_0 \partial I^{(0)}/\partial y\right)\right) \quad \text{Eq. (5)}$$

[0074] All values in the above equation depend on the sample location, denoted as (i', j'). Assuming the motion is consistent in the local surrounding area, Δ can be minimized inside the (2M+1)×(2M+1) square window Ω centered on the currently predicted point (i, j), where M is equal to 2:

[0075] $$\left(v_x, v_y\right) = \underset{v_x, v_y}{\operatorname{argmin}} \sum_{[i',j]\in\Omega} \Delta^2[i', j'] \quad \text{Eq. (6)}$$

[0076] For this optimization problem, the JEM uses a simplified approach making first a minimization in the vertical direction and then in the horizontal direction. This results in the following:

[0077] $$v_x = (s_1 + r) > m ? \, clip3\left(-thBIO, thBIO, -\frac{s_3}{(s_1 + r)}\right) : 0 \quad \text{Eq. (7)}$$

[0078] $$v_y = (s_5 + r) > m ? \, clip3\left(-thBIO, thBIO, -\frac{s_6 - v_x s_2/2}{(s_5 + r)}\right) : 0 \quad \text{Eq. (8)}$$

[0079] where,

[0080] $$\begin{aligned} s_1 &= \sum_{[i',j]\in\Omega} \left(\tau_1 \partial I^{(1)}/\partial x + \tau_0 \partial I^{(0)}/\partial x\right)^2; \; s_3 = \sum_{[i',j]\in\Omega} \left(I^{(1)} - I^{(0)}\right)\left(\tau_1 \partial I^{(1)}/\partial x + \tau_0 \partial I^{(0)}/\partial x\right); \\ s_2 &= \sum_{[i',j]\in\Omega} \left(\tau_1 \partial I^{(1)}/\partial x + \tau_0 \partial I^{(0)}/\partial x\right)\left(\tau_1 \partial I^{(1)}/\partial y + \tau_0 \partial I^{(0)}/\partial y\right); \\ s_5 &= \sum_{[i',j]\in\Omega} \left(\tau_1 \partial I^{(1)}/\partial y + \tau_0 \partial I^{(0)}/\partial y\right)^2; \; s_6 = \sum_{[i',j]\in\Omega} \left(I^{(1)} - I^{(0)}\right)\left(\tau_1 \partial I^{(1)}/\partial y + \tau_0 \partial I^{(0)}/\partial y\right) \end{aligned} \quad \text{Eq. (9)}$$

[0081] In order to avoid division by zero or a very small value, regularization parameters r and m can be introduced in Eq. (7) and Eq. (8), where:

[0082] $$r = 500 \cdot 4^{d-8} \quad \text{Eq. (10)}$$

[0083] $$m = 700 \cdot 4^{d-8} \quad \text{Eq. (11)}$$

[0084] Here, d is bit depth of the video samples.

[0085] In order to keep the memory access for BIO the same as for regular bi-predictive motion compensation, all prediction and gradients values, $I^{(k)}, \partial I^{(k)}/\partial x, \partial I^{(k)}/\partial y$, are

calculated for positions inside the current block. FIG. 9A shows an example of access positions outside of a block 900. As shown in FIG. 9A, in Eq. (9), (2M+1)×(2M+1) square window Ω centered in currently predicted point on a boundary of predicted block needs to accesses positions outside of the block. In the JEM, values of $I^{(k)}, \partial I^{(k)}/\partial x, \partial I^{(k)}/\partial y$ outside of the block are set to be equal to the nearest available value inside the block. For example, this can be implemented as a padding area 901, as shown in FIG. 9B.

[0086] With BIO, it is possible that the motion field can be refined for each sample. To reduce the computational complexity, a block-based design of BIO is used in the JEM. The motion refinement can be calculated based on a 4x4 block. In the block-based BIO, the values of s_n in Eq. (9) of all samples in a 4x4 block can be aggregated, and then the aggregated values of s_n in are used to derived BIO motion vectors offset for the 4×4 block. More specifically, the following formula can used for block-based BIO derivation:

[0087]

$$s_{1,b_k} = \sum_{(x,y)\in b_k} \sum_{[i',j]\in\Omega(x,y)} \left(\tau_1 \partial I^{(1)}/\partial x + \tau_0 \partial I^{(0)}/\partial x\right)^2; s_{3,b_k} = \sum_{(x,y)\in b_k} \sum_{[i',j]\in\Omega} \left(I^{(1)} - I^{(0)}\right)\left(\tau_1 \partial I^{(1)}/\partial x + \tau_0 \partial I^{(0)}/\partial x\right);$$
$$s_{2,b_k} = \sum_{(x,y)\in b_k} \sum_{[i',j]\in\Omega} \left(\tau_1 \partial I^{(1)}/\partial x + \tau_0 \partial I^{(0)}/\partial x\right)\left(\tau_1 \partial I^{(1)}/\partial y + \tau_0 \partial I^{(0)}/\partial y\right); \quad \text{Eq. (12)}$$
$$s_{5,b_k} = \sum_{(x,y)\in b_k} \sum_{[i',j]\in\Omega} \left(\tau_1 \partial I^{(1)}/\partial y + \tau_0 \partial I^{(0)}/\partial y\right)^2; s_{6,b_k} = \sum_{(x,y)\in b_k} \sum_{[i',j]\in\Omega} \left(I^{(1)} - I^{(0)}\right)\left(\tau_1 \partial I^{(1)}/\partial y + \tau_0 \partial I^{(0)}/\partial y\right)$$

[0088] Here, b_k denotes the set of samples belonging to the k-th 4x4 block of the predicted block. s_n in Eq (7) and Eq (8) are replaced by $((s_{n,bk}) >> 4)$ to derive the associated motion vector offsets.

[0089] In some scenarios, MV regiment of BIO may be unreliable due to noise or irregular motion. Therefore, in BIO, the magnitude of MV regiment is clipped to a threshold value. The threshold value is determined based on whether the reference pictures of the current picture are all from one direction. For example, if all the reference pictures of the current picture are from one direction, the value of the threshold is set to $12 \times 2^{14-d}$; otherwise, it is set to $12 \times 2^{13-d}$.

[0090] Gradients for BIO can be calculated at the same time with motion compensation interpolation using operations consistent with HEVC motion compensation process (e.g., 2D separable Finite Impulse Response (FIR)). In some embodiments, the input for the 2D separable FIR is the same reference frame sample as for motion compensation process and fractional position (*fracX*, *fracY*) according to the fractional part of block motion vector. For horizontal gradient $\partial I/\partial x$, a signal is first interpolated vertically using *BIOfilterS* corresponding to the

fractional position *fracY* with de-scaling shift $d-8$. Gradient filter *BIOfilterG* is then applied in horizontal direction corresponding to the fractional position *fracX* with de-scaling shift by $18-d$. For vertical gradient $\partial I/\partial y$, a gradient filter is applied vertically using *BIOfilterG* corresponding to the fractional position *fracY* with de-scaling shift $d-8$. The signal displacement is then performed using *BIOfilterS* in horizontal direction corresponding to the fractional position *fracX* with de-scaling shift by $18-d$. The length of interpolation filter for gradients calculation BIOfilterG and signal displacement BIOfilterF can be shorter (e.g., 6-tap) in order to maintain reasonable complexity. Table 2 shows example filters that can be used for gradients calculation of different fractional positions of block motion vector in BIO. Table 3 shows example interpolation filters that can be used for prediction signal generation in BIO.

Table 2: Example filters for gradient calculation in BIO

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

Table 3: Example interpolation filters for prediction signal generation in BIO

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

[0091] In the JEM, BIO can be applied to all bi-predicted blocks when the two predictions

are from different reference pictures. When Local Illumination Compensation (LIC) is enabled for a CU, BIO can be disabled.

[0092] In some embodiments, OBMC is applied for a block after normal MC process. To reduce the computational complexity, BIO may not be applied during the OBMC process. This means that BIO is applied in the MC process for a block when using its own MV and is not applied in the MC process when the MV of a neighboring block is used during the OBMC process.

[0093] In JEM, BIO is only invoked for the luma component.

1.5 Examples of frame-rate up conversion (FRUC)

[0094] A FRUC flag can be signaled for a CU when its merge flag is true. When the FRUC flag is false, a merge index can be signaled and the regular merge mode is used. When the FRUC flag is true, an additional FRUC mode flag can be signaled to indicate which method (e.g., bilateral matching or template matching) is to be used to derive motion information for the block.

[0095] At the encoder side, the decision on whether using FRUC merge mode for a CU is based on RD cost selection as done for normal merge candidate. For example, multiple matching modes (e.g., bilateral matching and template matching) are checked for a CU by using RD cost selection. The one leading to the minimal cost is further compared to other CU modes. If a FRUC matching mode is the most efficient one, FRUC flag is set to true for the CU and the related matching mode is used.

[0096] Typically, motion derivation process in FRUC merge mode has two steps: a CU-level motion search is first performed, then followed by a Sub-CU level motion refinement. At CU level, an initial motion vector is derived for the whole CU based on bilateral matching or template matching. First, a list of MV candidates is generated and the candidate that leads to the minimum matching cost is selected as the starting point for further CU level refinement. Then a local search based on bilateral matching or template matching around the starting point is performed. The MV results in the minimum matching cost is taken as the MV for the whole CU. Subsequently, the motion information is further refined at sub-CU level with the derived CU motion vectors as the starting points.

[0097] For example, the following derivation process is performed for a $W \times H$ CU motion information derivation. At the first stage, MV for the whole $W \times H$ CU is derived. At the second

stage, the CU is further split into $M \times M$ sub-CUs. The value of M is calculated as in (16), D is a predefined splitting depth which is set to 3 by default in the JEM. Then the MV for each sub-CU is derived.

[0098] $$M = \max\{4, \min\left\{\frac{M}{2^D}, \frac{N}{2^D}\right\}\}$$ Eq. (13)

[0099] FIG. 10 shows an example of bilateral matching used in the Frame-Rate Up Conversion (FRUC) method. The bilateral matching is used to derive motion information of the current CU by finding the closest match between two blocks along the motion trajectory of the current CU (1000) in two different reference pictures (1010, 1011). Under the assumption of continuous motion trajectory, the motion vectors MV0 (1001) and MV1 (1002) pointing to the two reference blocks are proportional to the temporal distances, e.g., TD0 (1003) and TD1 (1004), between the current picture and the two reference pictures. In some embodiments, when the current picture 1000 is temporally between the two reference pictures (1010, 1011) and the temporal distance from the current picture to the two reference pictures is the same, the bilateral matching becomes mirror based bi-directional MV.

[00100] FIG. 11 shows an example of template matching used in the Frame-Rate Up Conversion (FRUC) method. Template matching can be used to derive motion information of the current CU 1100 by finding the closest match between a template (e.g., top and/or left neighboring blocks of the current CU) in the current picture and a block (e.g., same size to the template) in a reference picture 1110. Except the aforementioned FRUC merge mode, the template matching can also be applied to AMVP mode. In both JEM and HEVC, AMVP has two candidates. With the template matching method, a new candidate can be derived. If the newly derived candidate by template matching is different to the first existing AMVP candidate, it is inserted at the very beginning of the AMVP candidate list and then the list size is set to two (e.g., by removing the second existing AMVP candidate). When applied to AMVP mode, only CU level search is applied.

[00101] The MV candidate set at CU level can include the following: (1) original AMVP candidates if the current CU is in AMVP mode, (2) all merge candidates, (3) several MVs in the interpolated MV field (described later), and top and left neighboring motion vectors.

[00102] When using bilateral matching, each valid MV of a merge candidate can be used as an input to generate a MV pair with the assumption of bilateral matching. For example, one valid

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

[00103] In some implementations, four MVs from the interpolated MV field can also be added to the CU level candidate list. More specifically, the interpolated MVs at the position (0, 0), (W/2, 0), (0, H/2) and (W/2, H/2) of the current CU are added. When FRUC is applied in AMVP mode, the original AMVP candidates are also added to CU level MV candidate set. In some implementations, at the CU level, 15 MVs for AMVP CUs and 13 MVs for merge CUs can be added to the candidate list.

[00104] The MV candidate set at sub-CU level includes an MV determined from a CU-level search, (2) top, left, top-left and top-right neighboring MVs, (3) scaled versions of collocated MVs from reference pictures, (4) one or more ATMVP candidates (e.g., up to four), and (5) one or more STMVP candidates (e.g., up to four). The scaled MVs from reference pictures are derived as follows. The reference pictures in both lists are traversed. The MVs at a collocated position of the sub-CU in a reference picture are scaled to the reference of the starting CU-level MV. ATMVP and STMVP candidates can be the four first ones. At the sub-CU level, one or more MVs (e.g., up to 17) are added to the candidate list.

[00105] Generation of an interpolated MV field. Before coding a frame, interpolated motion field is generated for the whole picture based on unilateral ME. Then the motion field may be used later as CU level or sub-CU level MV candidates.

[00106] In some embodiments, the motion field of each reference pictures in both reference lists is traversed at 4×4 block level. FIG. 12 shows an example of unilateral Motion Estimation (ME) 1200 in the FRUC method. For each 4×4 block, if the motion associated to the block passing through a 4×4 block in the current picture and the block has not been assigned any interpolated motion, the motion of the reference block is scaled to the current picture according to the temporal distance TD0 and TD1 (the same way as that of MV scaling of TMVP in HEVC) and the scaled motion is assigned to the block in the current frame. If no scaled MV is assigned to a 4×4 block, the block's motion is marked as unavailable in the interpolated motion field.

[00107] Interpolation and matching cost. When a motion vector points to a fractional sample position, motion compensated interpolation is needed. To reduce complexity, bi-linear interpolation instead of regular 8-tap HEVC interpolation can be used for both bilateral matching and template matching.

[00108] The calculation of matching cost is a bit different at different steps. When selecting the candidate from the candidate set at the CU level, the matching cost can be the absolute sum difference (SAD) of bilateral matching or template matching. After the starting MV is determined, the matching cost C of bilateral matching at sub-CU level search is calculated as follows:

[00109] $$C = SAD + w \cdot \left(|MV_x - MV_x^s| + |MV_y - MV_y^s|\right) \quad \text{Eq. (14)}$$

[00110] Here, w is a weighting factor. In some embodiments, w can be empirically set to 4. MV and MV^s indicate the current MV and the starting MV, respectively. SAD may still be used as the matching cost of template matching at sub-CU level search.

[00111] In FRUC mode, MV is derived by using luma samples only. The derived motion will be used for both luma and chroma for MC inter prediction. After MV is decided, final MC is performed using 8-taps interpolation filter for luma and 4-taps interpolation filter for chroma.

[00112] MV refinement is a pattern based MV search with the criterion of bilateral matching cost or template matching cost. In the JEM, two search patterns are supported – an unrestricted center-biased diamond search (UCBDS) and an adaptive cross search for MV refinement at the CU level and sub-CU level, respectively. For both CU and sub-CU level MV refinement, the MV is directly searched at quarter luma sample MV accuracy, and this is followed by one-eighth luma sample MV refinement. The search range of MV refinement for the CU and sub-CU step are set equal to 8 luma samples.

[00113] In the bilateral matching merge mode, bi-prediction is applied because the motion information of a CU is derived based on the closest match between two blocks along the motion trajectory of the current CU in two different reference pictures. In the template matching merge mode, the encoder can choose among uni-prediction from list0, uni-prediction from list1, or bi-prediction for a CU. The selection ca based on a template matching cost as follows:

[00114] If costBi <= factor * min (cost0, cost1)

[00115] bi-prediction is used;

[00116] Otherwise, if cost0 <= cost1

[00117] uni-prediction from list0 is used;

[00118] Otherwise,

[00119] uni-prediction from list1 is used;

[00120] Here, cost0 is the SAD of list0 template matching, cost1 is the SAD of list1 template matching and costBi is the SAD of bi-prediction template matching. For example, when the value of factor is equal to 1.25, it means that the selection process is biased toward bi-prediction. The inter prediction direction selection can be applied to the CU-level template matching process.

1.6 Examples of MV derived for MC in chroma components

[00121] In an example, the HEVC standard defines how to derive the MV used for MC in chroma components (noted as *mvC*) from the MV used for MC in the luma component (noted as *mv*). Generally speaking, *mvC* is calculated as *mv* multiplying a factor, which relies on the color format, such as 4:2:0 or 4:2:2.

1.7 Examples of current picture referencing in VVC

[00122] Intra block copy (IBC, or intra picture block compensation), also named current picture referencing (CPR) was adopted in HEVC screen content coding extensions (SCC). This tool is very efficient for coding of screen content video in that repeated patterns in text and graphics rich content occur frequently within the same picture. Having a previously reconstructed block with equal or similar pattern as a predictor can effectively reduce the prediction error and therefore improve coding efficiency.

[00123] Similar to the design of CRP in HEVC SCC, In VVC, The use of the IBC mode is signaled at both sequence and picture level. When the IBC mode is enabled at sequence parameter set (SPS), it can be enabled at picture level. When the IBC mode is enabled at picture level, the current reconstructed picture is treated as a reference picture. Therefore, no syntax change on block level is needed on top of the existing VVC inter mode to signal the use of the IBC mode.

[00124] Features of this embodiment include:

[00125] ○ It is treated as a normal inter mode. Therefore, merge and skip modes are also available for the IBC mode. The merge candidate list construction is unified, containing merge candidates from the neighboring positions that are either coded in the IBC mode or the HEVC

inter mode. Depending on the selected merge index, the current block under merge or skip mode can merge into either an IBC mode coded neighbor or otherwise an normal inter mode coded one with different pictures as reference pictures.

[00126] ○ Block vector prediction and coding schemes for the IBC mode reuse the schemes used for motion vector prediction and coding in the HEVC inter mode (AMVP and MVD coding).

[00127] ○ The motion vector for the IBC mode, also referred as block vector, is coded with integer-pel precision, but stored in memory in 1/16-pel precision after decoding as quarter-pel precision is required in interpolation and deblocking stages. When used in motion vector prediction for the IBC mode, the stored vector predictor will be right shifted by 4.

[00128] ○ Search range: it is restricted to be within the current CTU.

[00129] ○ CPR is disallowed when affine mode/triangular mode/GBI/weighted prediction is enabled.

1.8 Examples of pairwise average candidates

[00130] 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. Suppose the MVs of two merge candidates are MV0 = (MV0x, MV0y) and MV1=(MV1x, MV1y), then the MV of the pairwise merge candidate denoted as MV*=(MV*x, MV*y) is derived as

[00131] MV*x=(MV0x+MV1x)/2, and

[00132] MV*y=(MV0y+MV1y)/2.

[00133] In addition, when MV0 and MV1 refer to the current picture (i.e., CPR mode), MV*x and MV*y are further rounded to remove the part with a higher precision than full pixel to make sure the integer MV is obtained:

[00134] MV*x = (MV*x /16)<<4, and

[00135] MV*y = (MV*y /16)<<4.

[00136] It is noted that for each pair, if one of the two is coded with CPR and the other is not, such pair is disallowed to generate the pairwise average candidate.

1.9 Examples of triangular prediction mode

[00137] The concept of the triangular prediction mode (TPM) is to introduce a new triangular partition for motion compensated prediction. It splits a CU into two triangular prediction units, in either diagonal or inverse diagonal direction. Each triangular prediction unit in the CU is inter-predicted using its own uni-prediction motion vector and reference frame index which are derived from a single uni-prediction candidate list. An adaptive weighting process is performed to the diagonal edge after predicting the triangular prediction units. Then, the transform and quantization process are applied to the whole CU. It is noted that this mode is only applied to merge mode (note: skip mode is treated as a special merge mode). Uni-prediction candidate list for TPM.

[00138] The uni-prediction candidate list, named TPM motion candidate list, consists of five uni-prediction motion vector candidates. It is derived from seven neighboring blocks including five spatial neighboring blocks and two temporal co-located blocks. The motion vectors of the seven neighboring blocks are collected and put into the uni-prediction candidate list according in the order of uni-prediction motion vectors, L0 motion vector of bi-prediction motion vectors, L1 motion vector of bi-prediction motion vectors, and averaged motion vector of the L0 and L1 motion vectors of bi-prediction motion vectors. If the number of candidates is less than five, zero motion vector is added to the list. Motion candidates added in this list for TPM are called TPM candidates, motion information derived from spatial/temporal blocks are called regular motion candidates.

[00139] More specifically, the following steps are involved:

[00140] (1) Obtain regular motion candidates from A1, B1, B0, A0, B2, Col and Col2 (similar as those in the regular merge mode) without any pruning operations.

[00141] (2) Set variable numCurrMergeCand = 0

[00142] (3) For each regular motion candidates derived from A1, B1, B0, A0, B2, Col and Col2 and numCurrMergeCand is less than 5, if the regular motion candidate is uni-prediction (either from List 0 or List 1), it is directly added to the merge list as an TPM candidate with numCurrMergeCand increased by 1. Such a TPM candidate is named “originally uni-predicted candidate”.

[00143] Full pruning is applied.

[00144] (4) For each motion candidates derived from A1, B1, B0, A0, B2, Col and Col2 and numCurrMergeCand is less than 5, if the regular motion candidate is bi-prediction, the motion information from List 0 is added to the TPM merge list (that is, modified to be uni-prediction from List 0) as a new TPM candidate and numCurrMergeCand increased by 1. Such a TPM candidate is named 'Truncated List0-predicted candidate'.

[00145] Full pruning is applied.

[00146] (5) For each motion candidates derived from A1, B1, B0, A0, B2, Col and Col2 and numCurrMergeCand is less than 5, if the regular motion candidate is bi-prediction, the motion information from List 1 is added to the TPM merge list (that is, modified to be uni-prediction from List 1) and numCurrMergeCand increased by 1. Such a TPM candidate is named 'Truncated List1-predicted candidate'.

[00147] Full pruning is applied.

[00148] (6) For each motion candidates derived from A1, B1, B0, A0, B2, Col and Col2 and numCurrMergeCand is less than 5, if the regular motion candidate is bi-prediction, the motion information of List 1 is firstly scaled to List 0 reference picture, and the average of the two MVs (one is from original List 0, and the other is the scaled MV from List 1) is added to the TPM merge list, such a candidate is called averaged uni-prediction from List 0 motion candidate and numCurrMergeCand increased by 1.

[00149] Full pruning is applied.

[00150] (7) If numCurrMergeCand is less than 5, zero motion vector candidates are added.

[00151] When inserting a candidate to the list, if it has to be compared to all previously added candidates to see whether it is identical to one of them, such a process is called full pruning.

[00152] Assume the scaled MV denoted by (MV1'x, MV1'y) and the List 0 MV by (MV0x, MV0y). The averaged uni-prediction from List 0 motion candidate denoted by (MV*x, MV*y) is defined as:

[00153] MV*x=(MV0x+MV1'x + 1) >> 1, and

[00154] MV*y=(MV0y+MV1'y +1) >>1.

2 Drawbacks of existing implementations

[00155] The existing sub-block based prediction technologies have the following problems:

[00156] (1) In some existing implementations, the size of sub-blocks (such as 4×4 in JEM)

is majorly designed for the luma component. In JEM, the size of sub-blocks for is 2×2 the chroma components with the 4:2:0 format, and 2×4 the chroma components with the 4:2:2 format. The small size of sub-blocks imposes a higher band-width requirement. FIG. 13 shows an example of sub-blocks of a 16×16 block (8×8 for Cb/Cr) for different components with the 4:2:0 format in JEM.

[00157] (2) In some existing sub-block based tools such as the affine prediction in JEM, MVs of each sub-block are calculated with the affine model as shown in equation (1) independently for each component, which results in misalignment of motion vectors between luma and chroma components.

[00158] (3) In some existing sub-block based tools such as affine prediction, the usage constrains are different for the merge mode and the non-merge inter mode (a.k.a. AMVP mode, or normal inter-mode), which need to be unified.

[00159] (4) In some existing implementations, the motion vector averaging operation to derive the pairwise merge candidate/ averaged uni-prediction from List 0 motion candidate should be aligned with the rounding method used in the sub-block prediction. So the hardware can have a unified design.

3 Exemplary methods for unified rounding in video coding

[00160] Example 1. One block is divided into sub-blocks in different ways depending on the color component and/or the color format and/or block shape/size.

[00161] (a) For one luma and its corresponding chroma blocks (e.g., for 4:2:0 color format, one MxN luma block corresponds to one M/2 x N/2 Cb block and one M/2 x N/2 Cr block), the number of divided sub-blocks may be different for different components. The block is divided into more sub-blocks for a component if the block has a larger size for the component.

[00162] (b) For each color component, the size of sub-blocks is the same, e.g., $w \times h$ for Y, Cb and Cr component, where $w \times h$ can be 8×8, 8×4, 4×8, 4×4, etc. For example, w and h are both equal to 4. Fig. 14 shows an example of proposed sub-blocks of a 16×16 block (8×8 for Cb/Cr) for different components with the 4:2:0 format. In one example, if $w>W$ or $h>H$, the block is not divided into sub-blocks for the component.

[00163] (c) Different sizes of sub-block within one block may be allowed. In one example, if W (or H) is not an integer multiple of w (or h), it is proposed to merge the sub-blocks with width/height smaller than w/h into its adjacent left or above sub-block. FIG. 15 shows an

example, where the 12×16 block (6×8 for Cb/Cr) is divided into sub-blocks. The sub-block size is 4×4. For Cb/Cr components, the size of last two sub-blocks is 2×4, so they are merged to their left neighboring blocks.

[00164] (d) Sub-block sizes may further depend on the block shape.

[00165] (i) In one example, for square blocks (with W equal to H), w may be set equal to h.

[00166] (ii) Alternatively, for non-square blocks (with W unequal to H), w may be unequal to h. For example, when W > H, w may be set larger than h, and versa vice.

[00167] (e) Sub-block sizes of different color components or different block shapes may be signaled in sequence parameter set (SPS), picture parameter set (PPS), slice header.

[00168] (f) Sub-block sizes of Y component and Cb/Cr components may be determined in different ways. For example, the sub-block size of Y component is adaptively determined from some candidates such as 4×4 8×8 4×8 and 8×4 but the sub-block size of Cb/Cr components are fixed to be 4×4.

[00169] (g) Sub-block sizes of Y component and Cb/Cr components may depend on the color formats, such as 4:2:0, 4:2:2 and 4:4:4.

[00170] Example 2. The MV of a sub-block of one color component can be derived from the MV(s) of one or more sub-blocks of another color component, which has (have) already been derived with the affine model. In this case, there is no need to derive motion information of one color component based on the information of the same color component three times.

[00171] (a) In one example, the MV of a sub-block for one component can be derived from the MV of one corresponding sub-block for another component. FIG. 16 shows an example for the 4:2:0 format. The blocks for each component are divided following the example in FIG. 13. A first derived MV* is copied from the MV of a corresponding 4×4 sub-block in the Y component, then the MV of the 2×2 sub-block in the Cb or Cr component is derived from MV*, following the rule specified in Section 1.6.

[00172] (b) In another example, the MV of a sub-block for one component can be derived from the MVs of several corresponding sub-blocks for another component. FIG. 17 shows an example for the 4:2:0 format, where the a 4×4 sub-block in the Cb or Cr component corresponds to four 4×4 sub-block in the Y component. The blocks for each component are divided following the example in FIG. 14. A first derived MV* is calculated from the MVs of several

corresponding sub-blocks for another component, then the MV of the 4×4 sub-block in the Cb or Cr component is derived from MV*, following the rule specified in Section 1.6.

[00173] (i) In one embodiment, MV* is calculated as the average of all corresponding sub-block MVs in the Y component. In an example, MV* = (MV0 + MV1 + MV2 + MV3) / 4. Suppose MV*=(MV*x, MV*y), MV0=(MV0x, MV0y), MV1=(MV1x, MV1y), MV2=(MV2x, MV2y) and MV3=(MV3x, MV3y),

[00174] (1) In one embodiment, MV*x=Shift(MV0x+MV1x+MV2x +MV3x, 2), MV*y=Shift(MV0y+MV1y+MV2y+MV3y, 2).

[00175] (2) In another embodiment, MVx*=SignShift(MV0x+MV1x+ MV2x+MV3x, 2), MVy*=SignShift(MV0y+MV1y+MV2y+MV3y, 2).

[00176] (3) In another embodiment, the calculation of MV* may be done with the following steps in order:

[00177] MV'x=Shift(MV0x+MV1x, 1),

[00178] MV'y=Shift(MV0y+MV1y, 1),

[00179] MV''x= Shift(MV2x+MV3x, 1),

[00180] MV''y=Shift(MV2y+MV3y, 1),

[00181] MV*x=Shift(MV'x+MV''x, 1), and

[00182] MV*y=Shift(MV'y+MV''y, 1).

[00183] (4) In another embodiment, the calculation of MV* may be done with the following steps in order:

[00184] MV'x=Shift(MV0x+MV2x, 1),

[00185] MV'y=Shift(MV0y+MV2y, 1),

[00186] MV''x= Shift(MV1x+MV3x, 1),

[00187] MV''y=Shift(MV1y+MV3y, 1),

[00188] MV*x=Shift(MV'x+MV''x, 1), and

[00189] MV*y=Shift(MV'y+MV''y, 1).

[00190] (5) In another embodiment, the calculation of MV* may be done with the following steps in order:

[00191] MV'x=SignShift(MV0x+MV1x, 1),

[00192] MV'y=SignShift(MV0y+MV1y, 1),

[00193] MV''x= SignShift(MV2x+MV3x, 1),

[00194] MV''y=SignShift(MV2y+MV3y, 1),

[00195] MV*x=SignShift(MV'x+MV''x, 1), and

[00196] MV*y=SignShift(MV'y+MV''y, 1).

[00197] (6) In another embodiment, the calculation of MV* may be done with the following steps in order:

[00198] MV'x=SignShift(MV0x+MV2x, 1),

[00199] MV'y=SignShift(MV0y+MV2y, 1),

[00200] MV''x = SignShift(MV1x+MV3x, 1),

[00201] MV''y = SignShift(MV1y+MV3y, 1),

[00202] MV*x = SignShift(MV'x+MV''x, 1), and

[00203] MV*y = SignShift(MV'y+MV''y, 1).

[00204] (ii) In one embodiment, MV* is calculated as the MV of top-left corresponding sub-block in the Y component. In the example, MV*= MV0.

[00205] (iii) In one embodiment, MV* is calculated as the MV of center corresponding sub-block in the Y component.

[00206] (iv) In one embodiment, MV* is calculated as the median of all corresponding sub-block MVs in the Y component. In an example, MV* =median(MV0,MV1,MV2,MV3).

[00207] (v) How to derive MVs of sub-blocks for one component from another component may be determined based on the color formats, such as 4:2:0, 4:2:2 and 4:4:4.

[00208] (1) In an example when the color format is 4:4:4, the sub-block sizes are the same for all components, and the MV of a sub-block for each sub-block is the same.

[00209] (2) In an example when the color format is 4:2:2. the sub-block sizes are the same for all components. A first derived MV* is calculated from the MVs of several corresponding sub-blocks for another component, then the MV of the sub-block in the Cb or Cr component is derived from MV*, following the rule specified in section 2.6. In the example shown in Fig. 18, one Cb/Cr sub-block corresponds to two Y sub-block with motion vectors MV0 and MV1.

[00210] (a) MV* =(MV0+MV1)/2; Suppose MV*=(MV*x, MV*y), MV0=(MV0x, MV0y), MV1=(MV1x, MV1y),

[00211] (i) In one embodiment, MV*x=Shift(MV0x+MV1x, 1),

MV*y=Shift(MV0y+MV1y, 1).

[00212] (ii) In another embodiment, MVx*=SignShift(MV0x+MV1x, 1), MVy*=SignShift(MV0y+MV1y, 1).

[00213] *Examples of unified constraints for merge and non-merge affine modes*

[00214] Example 3. The merge affine mode and the non-merge affine mode are allowed or disallowed with the same block size constraint.

[00215] (a) The block size constrain depends on the width and height compared to one or two thresholds. For example, both the Merge affine mode and the non-merge affine mode are allowed if the width and the height of the current block are both larger than M (e.g., M equal to 8), or width is larger than M0 and height is larger than M1 (e.g., M0 equal to 8 and M1 equal to 4); Otherwise, both the Merge affine mode and the non-merge affine mode are not allowed.

[00216] (b) The block size constrain depends on the total number of samples within one block (i.e., the area width x height). In one example, both the Merge affine mode and the non-merge affine mode are allowed if the area (i.e. width × height) of the current block is not smaller than N (e.g., N equal to 64); Otherwise, both the Merge affine mode and the non-merge affine mode are not allowed.

[00217] (c) For the merge affine mode, it can be an explicit mode with signaling a flag as in JEM, or it can be an implicit mode without signaling a flag as in other implementations. In the latter case, the affine merge candidate is not put into the unified merge candidate list if the merge affine mode is not allowed.

[00218] (d) For the non-merge affine mode, when affine is disallowed according to above rules, the signaling of indications of affine mode is skipped.

[00219] *Examples of the unified rounding process*

[00220] Example 4. Suppose one merge candidate has a motion vector denoted by MV0 = (MV0x, MV0y) and another merge candidate has a motion vector denoted by MV1=(MV1x, MV1y), then one MV of the pairwise merge candidate derived from these two merge candidates, denoted as MV*=(MV*x, MV*y), is derived as

[00221] MV*x=Shift(MV0x+MV1x, 1), and

[00222] MV*y=Shift(MV0y+MV1y, 1).

[00223] Alternatively,

[00224] MV*x=SignShift(MV0x+MV1x, 1), and

[00225] MV*y =SignShift(MV0y+MV1y, 1).

[00226] Example 5. Suppose one merge candidate has a motion vector denoted by MV0 = (MV0x, MV0y) and another merge candidate has a motion vector denoted by MV1=(MV1x, MV1y), and MV0 and/or MV1 refer to the current picture, then one MV of the pairwise merge candidate derived from these two merge candidates, denoted as MV*=(MV*x, MV*y), is derived as

[00227] (a) MV*x=Shift(MV0x+MV1x, 1); MV*y=Shift(MV0y+MV1y, 1); then MV*x=(Shift(MV*x, 4))<<4; MV*y=(Shift(MV*y, 4))<<4

[00228] (b) MV*x=SignShift(MV0x+MV1x, 1); MV*y= SignShift (MV0y+MV1y, 1); then MV*x=(SignShift (MV*x, 4))<<4; MV*y=(SignShift (MV*y, 4))<<4

[00229] (c) MV*x=(Shift(MV0x+MV1x, (W+1)))<<W; MV*y=(Shift(MV0y+MV1y, (W+1)))<<W, where W is an integer such as 2 or 4. W is an integer so that 1<<W is equal to the MV representation precision.

[00230] (d) MV*x=(SignShift(MV0x+MV1x, (W+1)))<<W; MV*y=(SignShift(MV0y+MV1y, (W+1))) <<W, where W is an integer such as 2 or 4. W is an integer so that 1<<W is equal to the MV representation precision.

[00231] Example 6. Assume the scaled MV denoted by (MV1'x, MV1'y) and the List 0 MV by (MV0x, MV0y). The averaged uni-prediction from List 0 motion candidate denoted by (MV*x, MV*y) is defined as:

[00232] MV*x=Shift(MV0x+MV1'x, 1), and

[00233] MV*y=Shift(MV0y+MV1'y, 1).

[00234] Alternatively,

[00235] MV*x=SignShift(MV0x+MV1'x, 1), and

[00236] MV*y =SignShift(MV0y+MV1'y, 1).

[00237] In some embodiments, SignShift(x, s) can right shift an integer toward 0, defined as:

$$\text{SignShift}(x, s) = \begin{cases} (x + off) \gg s & x \geq 0 \\ -((-x + off) \gg s) & x < 0 \end{cases}.$$

[00238] Herein, *off* = (1 << (s-1)) - 1.

[00239] In other embodiments, SignShift(x, s) can be alternatively defined as:

$$\text{SignShift}(x, s) = \begin{cases} (x + off) \gg s & x \geq 0 \\ (x + off + 1) \gg s & x < 0 \end{cases}.$$

[00240] Herein, *off* = (1 << (s-1)) - 1.

[00241] The examples described above may be incorporated in the context of the methods described below, e.g., methods 1930, 1960 and 1990 which may be implemented at a video decoder and/or video encoder.

[00242] FIG. 19A shows a flowchart of an exemplary method 1930 of video processing. The method 1930 includes, at step 1932, generating, for a processing of a current block of video, a pairwise merge candidate based on a pair of motion candidates.

[00243] The method 1930 includes, at step 1934, performing, based on the pairwise merge candidate, a conversion between the current block and a bitstream representation of the video.

[00244] FIG. 19B shows a flowchart of an exemplary method 1960 of video processing. The method 1960 includes, at step 1962, generating, for a processing of a current block of video, a pairwise merge candidate, the generating being based on a pair of motion candidates that refers to a current picture comprising the current block.

[00245] The method 1960 includes, at step 1964, performing, based on the pairwise merge candidate, a conversion between the current block and a bitstream representation of the video.

[00246] FIG. 19C shows a flowchart of an exemplary method 1990 of video processing. The method 1990 includes, at step 1992, generating, for a current block of video coded using a geometry partition mode, a uni-prediction motion candidate based on a scaled motion vector and a List0 motion vector.

[00247] The method 1990 includes, at step 1994, performing, based on the uni-prediction motion candidate, a conversion between the current block and a bitstream representation of the video.

[00248] In some embodiments, the following technical solutions are implemented:

A1. A method of video processing, comprising: generating, for a processing of a current block of video, a pairwise merge candidate based on a pair of motion candidates; and performing, based on the pairwise merge candidate, a conversion between the current block and a bitstream representation of the video, wherein the pair of motion candidates comprises a first motion vector MV0 = (MV0x, MV0y) of a reference picture list L in a first motion candidate and a second motion vector MV1 = (MV1x, MV1y) of the reference picture list L in a second motion candidate, and wherein MV* = (MV*x, MV*y) is the pairwise merge candidate such that:

$$MV^*x = \text{Shift}(MV0x + MV1x, 1) \text{ and } MV^*y = \text{Shift}(MV0y + MV1y, 1),$$

wherein $\text{Shift}(x, s) = (x + off) \gg s$, wherein *off* and *s* are integers, and wherein >> represents a right shift operation.

A2. A method of video processing, comprising: generating, for a processing of a current block of video, a pairwise merge candidate based on a pair of motion candidates; and performing, based on the pairwise merge candidate, a conversion between the current block and a bitstream representation of the video, wherein the pair of motion candidates comprises a first motion vector MV0 = (MV0x, MV0y) of a reference picture list L in a first motion candidate and a second motion vector MV1 = (MV1x, MV1y) of the reference picture list L in a second motion candidate, and wherein MV* = (MV*x, MV*y) is the pairwise merge candidate such that:

$$MV^*x = \text{SignShift}(MV0x + MV1x, 1) \quad \text{and} \quad MV^*y = \text{SignShift}(MV0y + MV1y, 1),$$

wherein

$$\text{SignShift}(x, s) = \begin{cases} (x + off) \gg s & x \geq 0 \\ -((-x + off) \gg s) & x < 0 \end{cases},$$

wherein *off* and *s* are integers, and wherein >> represents a shift operation.

A3. A method of video processing, comprising: generating, for a processing of a current block of video, a pairwise merge candidate based on a pair of motion candidates; and performing, based on the pairwise merge candidate, a conversion between the current block and a bitstream representation of the video, wherein the pair of motion candidates comprises a first motion vector MV0 = (MV0x, MV0y) of a reference picture list L in a first motion candidate and a second motion vector MV1 = (MV1x, MV1y) of the reference picture list L in a second motion candidate, and wherein MV* = (MV*x, MV*y) is the pairwise merge candidate such that:

$$MV^*x = \text{SignShift}(MV0x + MV1x, 1) \quad \text{and} \quad MV^*y = \text{SignShift}(MV0y + MV1y, 1),$$

wherein

$$\text{SignShift}(x, s) = \begin{cases} (x + off) \gg s & x \geq 0 \\ (x + off + 1) \gg s & x < 0 \end{cases},$$

wherein *off* and *s* are integers, and wherein >> represents a shift operation.

A4. The method of any of solutions A1 to A3, wherein the pair of merge candidates is from a merge candidate list.

A5. The method of any of solutions A1 to A3, wherein the reference picture list L comprises reference picture list 0 (L0) or reference picture list 1 (L1).

A6. The method of any of solutions A1 to A3, further comprising: inserting the pairwise merge candidate into the merge candidate list.

A7. The method of any of solutions A1 to A3, further comprising: updating, subsequent to performing the conversion, the merge candidate list.

A8. The method of any of solutions A1 to A7, wherein $off = 0$.

A9. The method of any of solutions A1 to A7, wherein $off = 0$ upon a determination that s = 0.

A10. The method of any of solutions A1 to A7, wherein $off = 1 \ll (s - 1)$.

A11. The method of any of solutions A1 to A7, wherein $off = 1 \ll (s - 1)$ upon a determination that $s \neq 0$.

A12. The method of any of solutions A1 to A7, wherein $off = \big(1 \ll (s - 1)\big) - 1$.

A13. The method of any of solutions A1 to A12, wherein the current video block is a coding unit.

A14. The method of any of solutions A1 to A13, wherein performing the conversion comprises generating the bitstream representation from the current video block.

A15. The method of any of solutions A1 to A13, wherein performing the conversion comprises generating the current video block from the bitstream representation.

A16. A video decoding apparatus comprising a processor configured to implement a method recited in any one of solutions A1 to A15.

A17. 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 solutions A1 to A15.

[00249] In some embodiments, the following technical solutions are implemented:

B1. A method of video processing, comprising: generating, for a processing of a current block of video, a pairwise merge candidate, wherein the generating is based on a pair of motion candidates that refers to a current picture comprising the current block; and performing, based on the pairwise merge candidate, a conversion between the current block and a bitstream representation of the video, wherein the pair of motion candidates comprises a first motion vector MV0 = (MV0x, MV0y) of a reference picture list L in a first motion candidate and a second motion vector MV1 = (MV1x, MV1y) of the reference picture list L in a second motion candidate, and wherein MV* = (MV*x, MV*y) is the pairwise merge candidate such that:

$$MV^*x = \text{Shift}(MV0x + MV1x, 1) \text{ and } MV^*y = \text{Shift}(MV0y + MV1y, 1),$$

wherein $\text{Shift}(x, s) = (x + off) \gg s$, wherein *off* and *s* are integers, and wherein >> represents a right shift operation.

B2. The method of solution B1, wherein generating the pairwise merge candidate further comprises performing the operations:

$$MV^*x = (\text{Shift}(MV^*x, 4)) \ll 4 \text{ and } MV^*y = \big(\text{Shift}(MV^*y, 4)\big) \ll 4.$$

B3. A method of video processing, comprising: generating, for a processing of a current block of video, a pairwise merge candidate, wherein the generating is based on a pair of motion candidates that refers to a current picture comprising the current block; and performing, based on the pairwise merge candidate, a conversion between the current block and a bitstream representation of the video, wherein the pair of motion candidates comprises a first motion vector MV0 = (MV0x, MV0y) of a reference picture list L in a first motion candidate and a second motion vector MV1 = (MV1x, MV1y) of the reference picture list L in a second motion candidate, and wherein MV* = (MV*x, MV*y) is the pairwise merge candidate such that:

$$MV^*x = \text{SignShift}(MV0x + MV1x, 1) \text{ and } MV^*y = \text{SignShift}(MV0y + MV1y, 1),$$

wherein

$$\text{SignShift}(x, s) = \begin{cases} (x + off) \gg s & x \geq 0 \\ -((-x + off) \gg s) & x < 0 \end{cases},$$

wherein *off* and *s* are integers, and wherein >> represents a shift operation.

B4. The method of solution B3, wherein generating the pairwise merge candidate further comprises performing the operations:

$$MV^*x = (\text{SignShift}(MV^*x, 4)) \ll 4 \text{ and } MV^*y = \big(\text{SignShift}(MV^*y, 4)\big) \ll 4.$$

B5. A method of video processing, comprising: generating, for a processing of a current block of video, a pairwise merge candidate, wherein the generating is based on a pair of motion candidates that refers to a current picture comprising the current block; and performing, based on the pairwise merge candidate, a conversion between the current block and a bitstream representation of the video, wherein the pair of motion candidates comprises a first motion vector MV0 = (MV0x, MV0y) of a reference picture list L in a first motion candidate and a second motion vector MV1 = (MV1x, MV1y) of the reference picture list L in a second motion candidate, and wherein MV* = (MV*x, MV*y) is the pairwise merge candidate such that:

$$MV^*x = \text{SignShift}(MV0x + MV1x, 1) \text{ and } MV^*y = \text{SignShift}(MV0y + MV1y, 1),$$

wherein

$$\text{SignShift}(x, s) = \begin{cases} (x + off) \gg s & x \geq 0 \\ (x + off + 1) \gg s & x < 0 \end{cases},$$

wherein off and s are integers, and wherein >> represents a shift operation.

B6. The method of solution B5, wherein generating the pairwise merge candidate further comprises performing the operations:

$$MV^*x = (\text{SignShift}(MV^*x, 4)) \ll 4 \text{ and } MV^*y = \left(\text{SignShift}(MV^*y, 4)\right) \ll 4.$$

B7. A method of video processing, comprising: generating, for a processing of a current block of video, a pairwise merge candidate, wherein the generating is based on a pair of motion candidates that refers to a current picture comprising the current block; and performing, based on the pairwise merge candidate, a conversion between the current block and a bitstream representation of the video, wherein the pair of motion candidates comprises a first motion vector MV0 = (MV0x, MV0y) of a reference picture list L in a first motion candidate and a second motion vector MV1 = (MV1x, MV1y) of the reference picture list L in a second motion candidate, and wherein MV* = (MV*x, MV*y) is the pairwise merge candidate such that:

$$MV^*x = (\text{Shift}(MV0x + MV1x, (W + 1))) \ll \text{W and}$$

$$MV^*y = (\text{Shift}(MV0y + MV1y, (W + 1))) \ll \text{W},$$

wherein $\text{Shift}(x, s) = (x + off) \gg s$, wherein *W*, *off* and *s* are integers, and wherein >> represents a right shift operation.

B8. A method of video processing, comprising: generating, for a processing of a current block of video, a pairwise merge candidate, wherein the generating is based on a pair of motion candidates that refers to a current picture comprising the current block; and performing, based on the pairwise merge candidate, a conversion between the current block and a bitstream representation of the video, wherein the pair of motion candidates comprises a first motion vector MV0 = (MV0x, MV0y) of a reference picture list L in a first motion candidate and a second motion vector MV1 = (MV1x, MV1y) of the reference picture list L in a second motion candidate, and wherein MV* = (MV*x, MV*y) is the pairwise merge candidate such that:

$$MV^*x = (\text{SignShift}(MV0x + MV1x, (W + 1))) \ll \text{W and}$$

$$MV^*y = (\text{SignShift}(MV0y + MV1y, (W + 1))) \ll \text{W},$$

wherein

$$\text{SignShift}(x, s) = \begin{cases} (x + off) \gg s & x \geq 0 \\ -((-x + off) \gg s) & x < 0 \end{cases},$$

wherein *W*, *off* and *s* are integers, and wherein >> represents a shift operation.

B9. A method of video processing, comprising: generating, for a processing of a current block of video, a pairwise merge candidate, wherein the generating is based on a pair of motion candidates that refers to a current picture comprising the current block; and performing, based on the pairwise merge candidate, a conversion between the current block and a bitstream representation of the video, wherein the pair of motion candidates comprises a first motion vector MV0 = (MV0x, MV0y) of a reference picture list L in a first motion candidate and a second motion vector MV1 = (MV1x, MV1y) of the reference picture list L in a second motion candidate, and wherein MV* = (MV*x, MV*y) is the pairwise merge candidate such that:

$MV^*x = (\text{SignShift}(MV0x + MV1x, (W + 1))) \ll \text{W}$ and

$MV^*y = (\text{SignShift}(MV0y + MV1y, (W + 1))) \ll \text{W},$

wherein

$$\text{SignShift}(x, s) = \begin{cases} (x + off) \gg s & x \geq 0 \\ (x + off + 1) \gg s & x < 0 \end{cases},$$

wherein *W*, *off* and *s* are integers, and wherein >> represents a shift operation.

B10. The method of any of solutions B7 to B9, wherein W = 2 or W = 4.

B11. The method of solution B10, wherein $(1 \ll W)$ is a motion vector representation precision.

B12. The method of any of solutions B1 to B11, wherein the pair of merge candidates is from a merge candidate list.

B13. The method of any of solutions B1 to B11, wherein the reference picture list L comprises reference picture list 0 (L0) or reference picture list 1 (L1).

B14. The method of any of solutions B1 to B11, further comprising: inserting the pairwise merge candidate into the merge candidate list.

B15. The method of any of solutions B1 to B11, further comprising: updating, subsequent to performing the conversion, the merge candidate list.

B16. The method of any of solutions B1 to B15, wherein $off = 0$.

B17. The method of any of solutions B1 to B15, wherein $off = 0$ upon a determination that s = 0.

B18. The method of any of solutions B1 to B15, wherein $off = 1 \ll (s - 1)$.

B19. The method of any of solutions B1 to B15, wherein $off = 1 \ll (s - 1)$ upon a determination that s $\neq$ 0.

B20. The method of any of solutions B1 to B15, wherein $off = \big(1 \ll (s - 1)\big) - 1$.

B21. The method of any of solutions B1 to B20, wherein the current video block is a coding unit.

B22. The method of any of solutions B1 to B21, wherein performing the conversion comprises generating the bitstream representation from the current block.

B23. The method of any of solutions B1 to B21, wherein performing the conversion comprises generating the current block from the bitstream representation.

B24. A video decoding apparatus comprising a processor configured to implement a method recited in any one of solutions B1 to B23.

B25. 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 solutions B1 to B23.

[00250] In some embodiments, the following technical solutions are implemented:

C1. A method of video processing, comprising: generating, for a current block of video coded using a geometry partition mode, a uni-prediction motion candidate based on a scaled motion vector and a List0 motion vector; and performing, based on the uni-prediction motion candidate, a conversion between the current block and a bitstream representation of the video, wherein the scaled motion vector is MV1' = (MV1'x, MV1'y) and the List0 motion vector is MV0 = (MV0x, MV0y), and wherein MV* = (MV*x, MV*y) is the uni-prediction motion candidate such that:

$$MV^*x = \text{Shift}(MV0x + MV1'x, 1) \text{ and } MV^*y = \text{Shift}(MV0y + MV1'y, 1),$$

wherein $\text{Shift}(x, s) = (x + off) \gg s$, wherein *off* and *s* are integers, and wherein >> represents a right shift operation.

C2. A method of video processing, comprising: generating, for a current block of video coded using a geometry partition mode, a uni-prediction motion candidate based on a scaled motion vector and a List0 motion vector; and performing, based on the uni-prediction motion candidate, a conversion between the current block and a bitstream representation of the video, wherein the scaled motion vector is MV1' = (MV1'x, MV1'y) and the List0 motion vector is MV0 = (MV0x, MV0y), and wherein MV* = (MV*x, MV*y) is the uni-prediction motion candidate such that:

$$MV^*x = \text{SignShift}(MV0x + MV1'x, 1) \text{ and } MV^*y = \text{SignShift}(MV0y + MV1'y, 1),$$

wherein

$$\text{SignShift}(x, s) = \begin{cases} (x + off) \gg s & x \geq 0 \\ -((-x + off) \gg s) & x < 0 \end{cases},$$

wherein *off* and *s* are integers, and wherein >> represents a shift operation.

C3. A method of video processing, comprising: generating, for a current block of video coded using a geometry partition mode, a uni-prediction motion candidate based on a scaled motion vector and a List0 motion vector; and performing, based on the uni-prediction motion candidate, a conversion between the current block and a bitstream representation of the video, wherein the scaled motion vector is MV1' = (MV1'x, MV1'y) and the List0 motion vector is MV0 = (MV0x, MV0y), and wherein MV* = (MV*x, MV*y) is the uni-prediction motion candidate such that:

$$MV^*x = \text{SignShift}(MV0x + MV1'x, 1) \text{ and } MV^*y = \text{SignShift}(MV0y + MV1'y, 1),$$

wherein

$$\text{SignShift}(x, s) = \begin{cases} (x + off) \gg s & x \geq 0 \\ (x + off + 1) \gg s & x < 0 \end{cases},$$

wherein *off* and *s* are integers, and wherein >> represents a shift operation.

C4. The method of any of claims 1 to 3, wherein MV* uses a reference picture from List0.

C5. The method of any of solutions C1 to C3, wherein the scaled motion vector and the List0 motion vector are from a motion candidate list derived from blocks coded using the geometry partition mode.

C6. The method of any of solutions C1 to C3, further comprising: inserting the uni-prediction motion candidate into the motion candidate list.

C7. The method of any of solutions C1 to C3, further comprising: updating, subsequent to performing the conversion, the motion candidate list.

C8. The method of any of solutions C1 to C7, wherein $off = 0$.

C9. The method of any of solutions C1 to C7, wherein $off = 0$ upon a determination that s = 0.

C10. The method of any of solutions C1 to C7, wherein $off = 1 \ll (s - 1)$.

C11. The method of any of solutions C1 to C7, wherein $off = 1 \ll (s - 1)$ upon a determination that $s \neq 0$.

C12. The method of any of solutions C1 to C7, wherein $off = \big(1 \ll (s - 1)\big) - 1$.

C13. The method of any of solutions C1 to C12, wherein the geometry partition mode comprises a triangular prediction mode (TPM).

C14. The method of any of solutions C1 to C12, wherein the current block is partitioned into multiple partitions with at least one partition being non-square and non-rectangular.
C15. The method of any of solutions C1 to C14, wherein the current video block is a coding unit.
C16. The method of any of solutions C1 to C15, wherein performing the conversion comprises generating the bitstream representation from the current video block.
C17. The method of any of solutions C1 to C15, wherein performing the conversion comprises generating the current video block from the bitstream representation.
C18. A video decoding apparatus comprising a processor configured to implement a method recited in any one of solutions C1 to C17.
C19. 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 solutions C1 to C17.

4 Example implementations of the disclosed technology

[00251] FIG. 20 is a block diagram of a video processing apparatus 2000. The apparatus 2000 may be used to implement one or more of the methods described herein. The apparatus 2000 may be embodied in a smartphone, tablet, computer, Internet of Things (IoT) receiver, and so on. The apparatus 2000 may include one or more processors 2002, one or more memories 2004 and video processing hardware 2006. The processor(s) 2002 may be configured to implement one or more methods (including, but not limited to, methods 1930, 1960 and 1990) described in the present document. The memory (memories) 2004 may be used for storing data and code used for implementing the methods and techniques described herein. The video processing hardware 2006 may be used to implement, in hardware circuitry, some techniques described in the present document.

[00252] In some embodiments, the video decoding methods may be implemented using an apparatus that is implemented on a hardware platform as described with respect to FIG. 20.

[00253] FIG. 21 is a block diagram showing an example video processing system 2100 in which various techniques disclosed herein may be implemented. Various implementations may include some or all of the components of the system 2100. The system 2100 may include input 2102 for receiving video content. The video content may be received in a raw or uncompressed format, e.g., 8 or 10 bit multi-component pixel values, or may be in a compressed or encoded format. The input 2102 may represent a network interface, a peripheral bus interface, or a storage

interface. Examples of network interface include wired interfaces such as Ethernet, passive optical network (PON), etc. and wireless interfaces such as Wi-Fi or cellular interfaces.

[00254] The system 2100 may include a coding component 2104 that may implement the various coding or encoding methods described in the present document. The coding component 2104 may reduce the average bitrate of video from the input 2102 to the output of the coding component 2104 to produce a coded representation of the video. The coding techniques are therefore sometimes called video compression or video transcoding techniques. The output of the coding component 2104 may be either stored, or transmitted via a communication connected, as represented by the component 2106. The stored or communicated bitstream (or coded) representation of the video received at the input 2102 may be used by the component 2108 for generating pixel values or displayable video that is sent to a display interface 2110. The process of generating user-viewable video from the bitstream representation is sometimes called video decompression. Furthermore, while certain video processing operations are referred to as "coding" operations or tools, it will be appreciated that the coding tools or operations are used at an encoder and corresponding decoding tools or operations that reverse the results of the coding will be performed by a decoder.

[00255] Examples of a peripheral bus interface or a display interface may include universal serial bus (USB) or high definition multimedia interface (HDMI) or Displayport, and so on. Examples of storage interfaces include SATA (serial advanced technology attachment), PCI, IDE interface, and the like. The techniques described in the present document may be embodied in various electronic devices such as mobile phones, laptops, smartphones or other devices that are capable of performing digital data processing and/or video display.

[00256] 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.

[00257] 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.

[00258] 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.

[00259] 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).

[00260] 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.

[00261] It is intended that the specification, together with the drawings, be considered exemplary only, where exemplary means an example. As used herein, the singular forms "a", "an" and "the" are intended to include the plural forms as well, unless the context clearly indicates otherwise. Additionally, the use of "or" is intended to include "and/or", unless the context clearly indicates otherwise.

[00262] 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.

[00263] 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.

[00264] Only a few implementations and examples are described and other implementations, enhancements and variations can be made based on what is described and illustrated in this patent document.

CLAIMS

What is claimed is:

1. A method of video processing, comprising:

generating, for a processing of a current block of video, a pairwise merge candidate based on a pair of motion candidates; and

performing, based on the pairwise merge candidate, a conversion between the current block and a bitstream representation of the video,

wherein the pair of motion candidates comprises a first motion vector MV0 = (MV0x, MV0y) of a reference picture list L in a first motion candidate and a second motion vector MV1 = (MV1x, MV1y) of the reference picture list L in a second motion candidate, and wherein MV* = (MV*x, MV*y) is the pairwise merge candidate such that:

$MV^*x = \text{Shift}(MV0x + MV1x, 1)$ and $MV^*y = \text{Shift}(MV0y + MV1y, 1)$,

wherein $\text{Shift}(x, s) = (x + off) \gg s$, wherein *off* and *s* are integers, and wherein >> represents a right shift operation.

2. A method of video processing, comprising:

generating, for a processing of a current block of video, a pairwise merge candidate based on a pair of motion candidates; and

performing, based on the pairwise merge candidate, a conversion between the current block and a bitstream representation of the video,

wherein the pair of motion candidates comprises a first motion vector MV0 = (MV0x, MV0y) of a reference picture list L in a first motion candidate and a second motion vector MV1 = (MV1x, MV1y) of the reference picture list L in a second motion candidate, and wherein MV* = (MV*x, MV*y) is the pairwise merge candidate such that:

$MV^*x = \text{SignShift}(MV0x + MV1x, 1)$ and $MV^*y = \text{SignShift}(MV0y + MV1y, 1)$,

wherein

$$\text{SignShift}(x, s) = \begin{cases} (x + off) \gg s & x \geq 0 \\ -((-x + off) \gg s) & x < 0 \end{cases},$$

wherein *off* and *s* are integers, and wherein >> represents a shift operation.

3. A method of video processing, comprising:

generating, for a processing of a current block of video, a pairwise merge candidate based on a pair of motion candidates; and

performing, based on the pairwise merge candidate, a conversion between the current block and a bitstream representation of the video,

wherein the pair of motion candidates comprises a first motion vector MV0 = (MV0x, MV0y) of a reference picture list L in a first motion candidate and a second motion vector MV1 = (MV1x, MV1y) of the reference picture list L in a second motion candidate, and wherein MV* = (MV*x, MV*y) is the pairwise merge candidate such that:

$MV^*x = \text{SignShift}(MV0x + MV1x, 1)$ and $MV^*y = \text{SignShift}(MV0y + MV1y, 1)$,

wherein

$$\text{SignShift}(x, s) = \begin{cases} (x + off) \gg s & x \geq 0 \\ (x + off + 1) \gg s & x < 0 \end{cases},$$

wherein *off* and *s* are integers, and wherein >> represents a shift operation.

4. The method of any of claims 1 to 3, wherein the pair of merge candidates is from a merge candidate list.

5. The method of any of claims 1 to 3, wherein the reference picture list L comprises reference picture list 0 (L0) or reference picture list 1 (L1).

6. The method of any of claims 1 to 3, further comprising:

inserting the pairwise merge candidate into the merge candidate list.

7. The method of any of claims 1 to 3, further comprising:

updating, subsequent to performing the conversion, the merge candidate list.

8. The method of any of claims 1 to 7, wherein $off = 0$.

9. The method of any of claims 1 to 7, wherein $off = 0$ upon a determination that s = 0.

10. The method of any of claims 1 to 7, wherein $off = 1 \ll (s - 1)$.

11. The method of any of claims 1 to 7, wherein $off = 1 \ll (s - 1)$ upon a determination that s $\neq$ 0.

12. The method of any of claims 1 to 7, wherein $off = \left(1 \ll (s - 1)\right) - 1$.

13. The method of any of claims 1 to 12, wherein the current video block is a coding unit.

14. The method of any of claims 1 to 13, wherein performing the conversion comprises generating the bitstream representation from the current video block.

15. The method of any of claims 1 to 13, wherein performing the conversion comprises generating the current video block from the bitstream representation.

16. A video decoding apparatus comprising a processor configured to implement a method recited in one or more of claims 1 to 15.

17. A computer program product stored on a non-transitory computer readable media, the computer program product including program code for carrying out the method recited in one or more of claims 1 to 15.

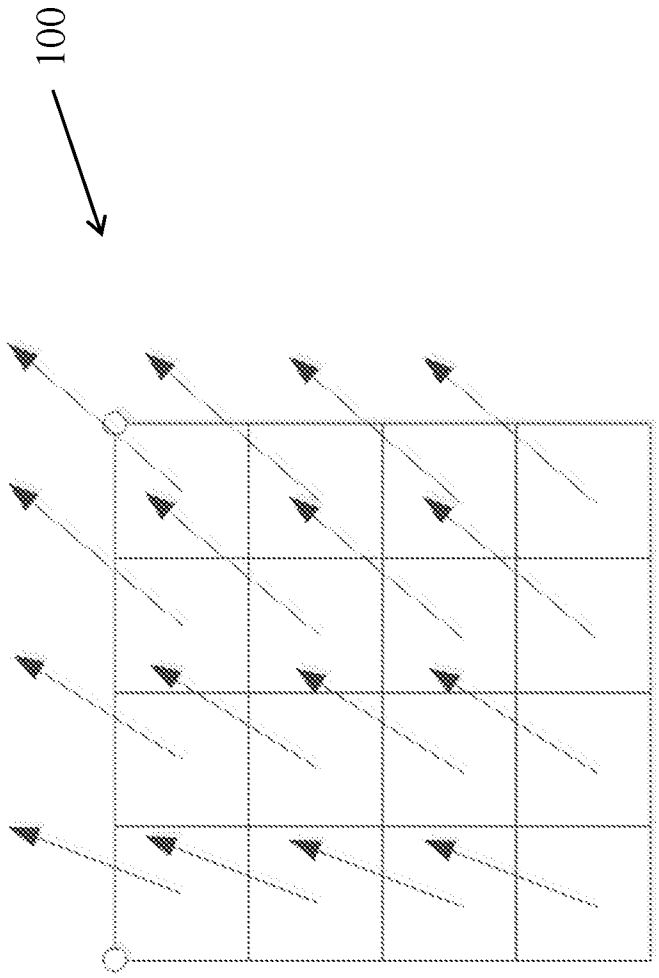

FIG. 1

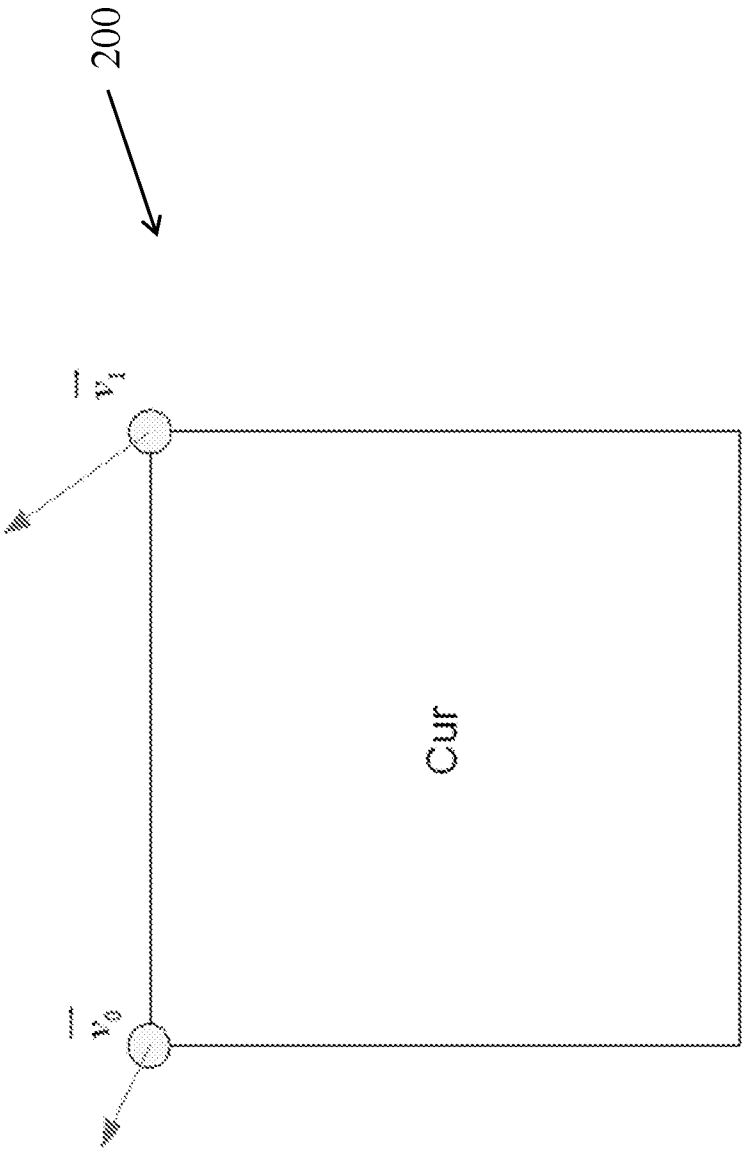

FIG. 2

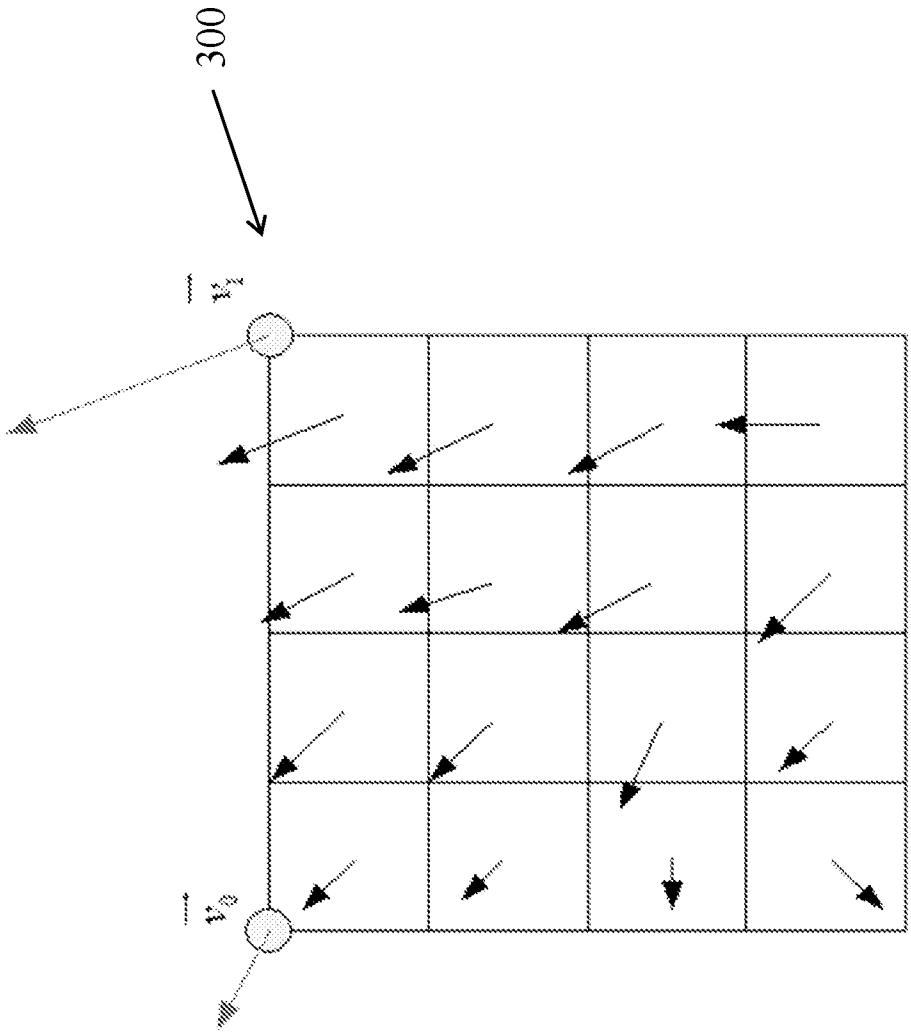

FIG. 3

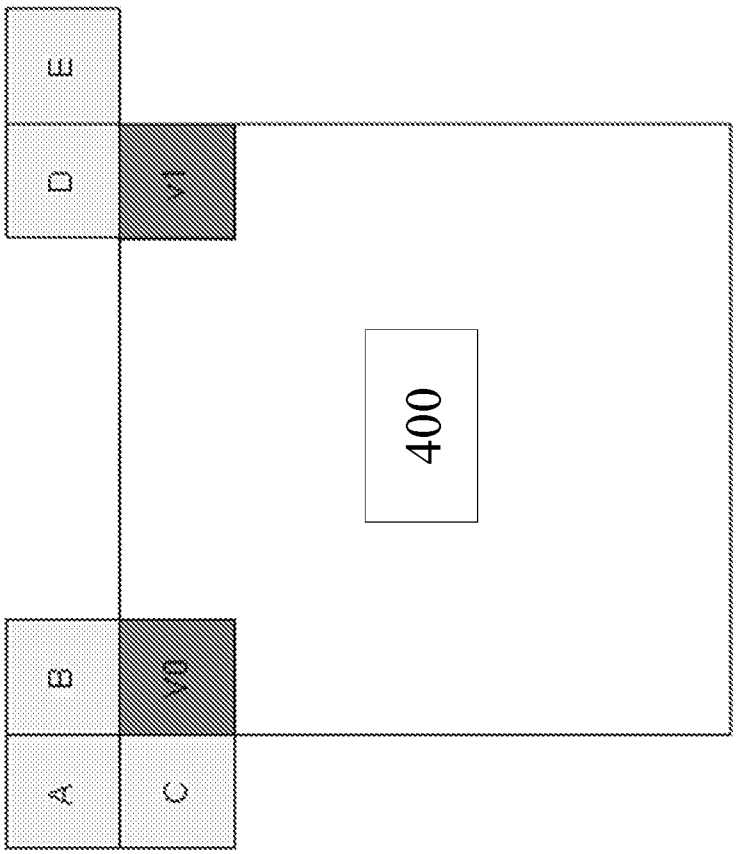

FIG. 4

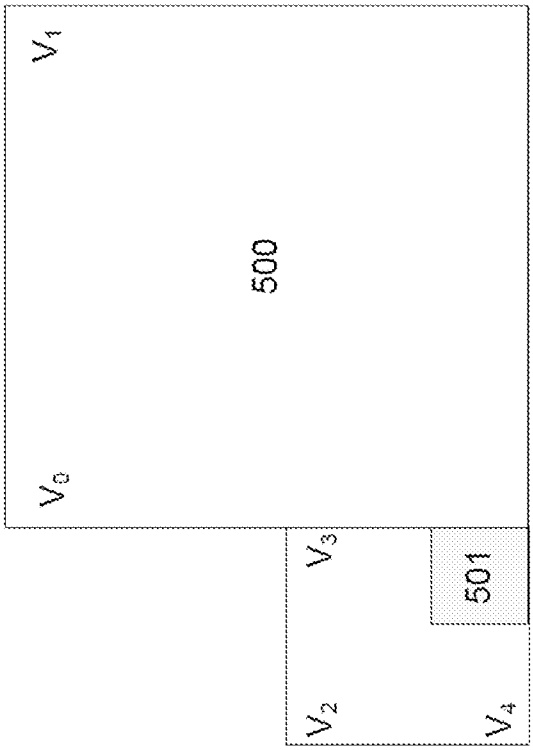

FIG. 5B

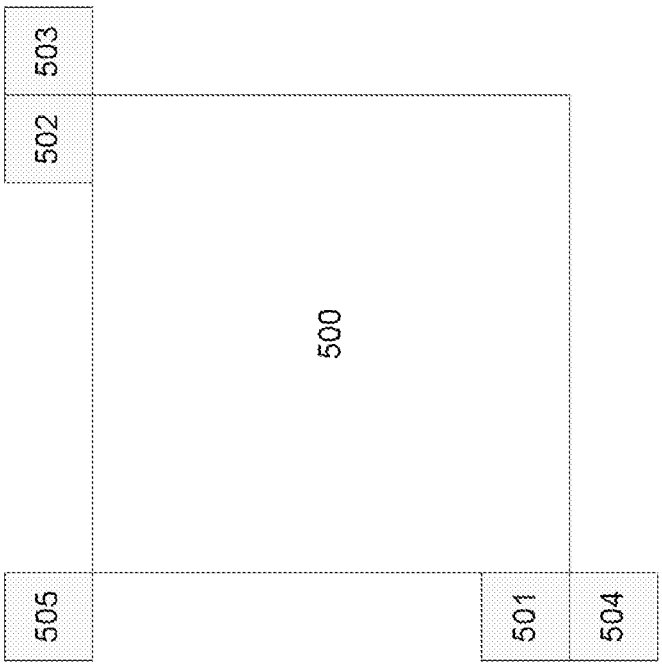

FIG. 5A

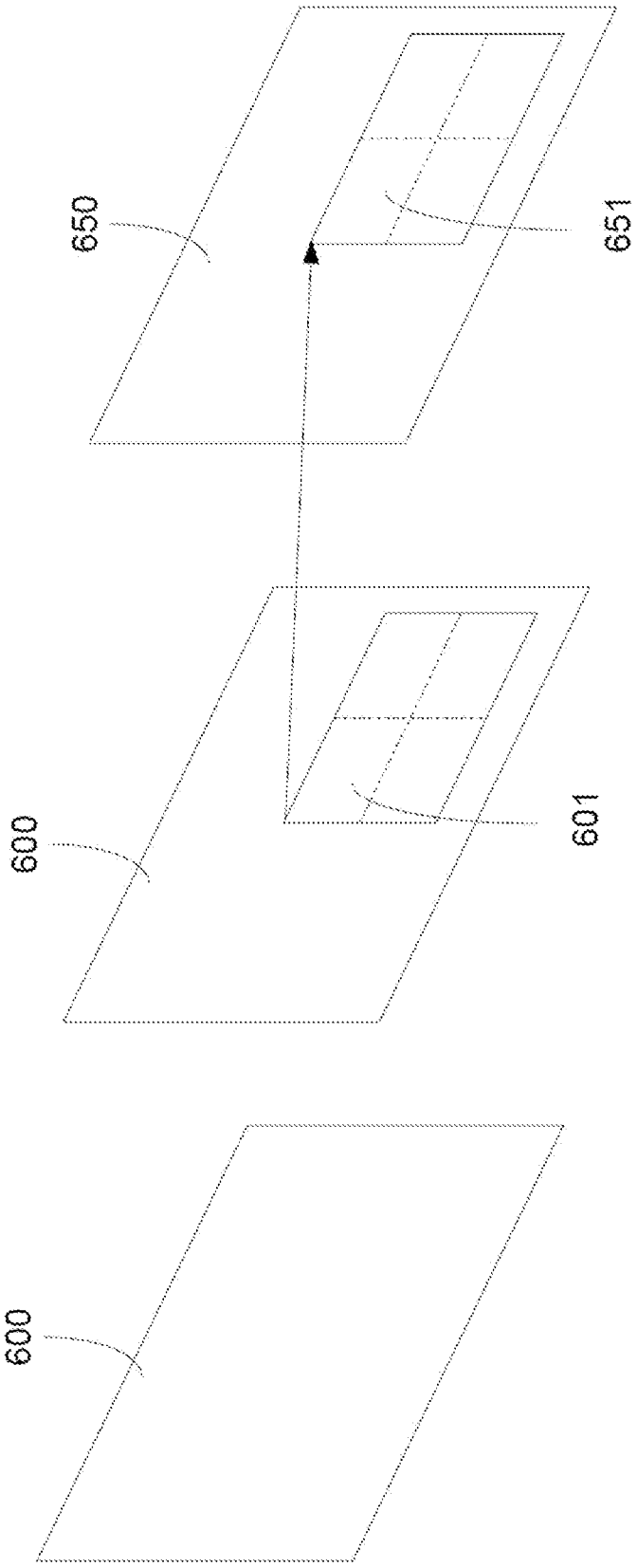

FIG. 6

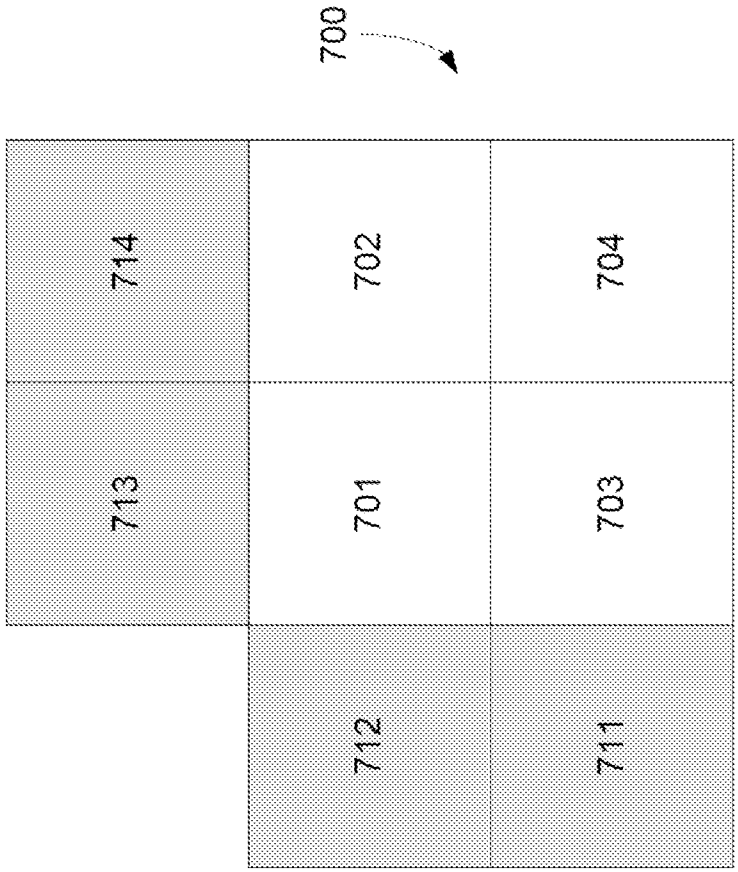

FIG. 7

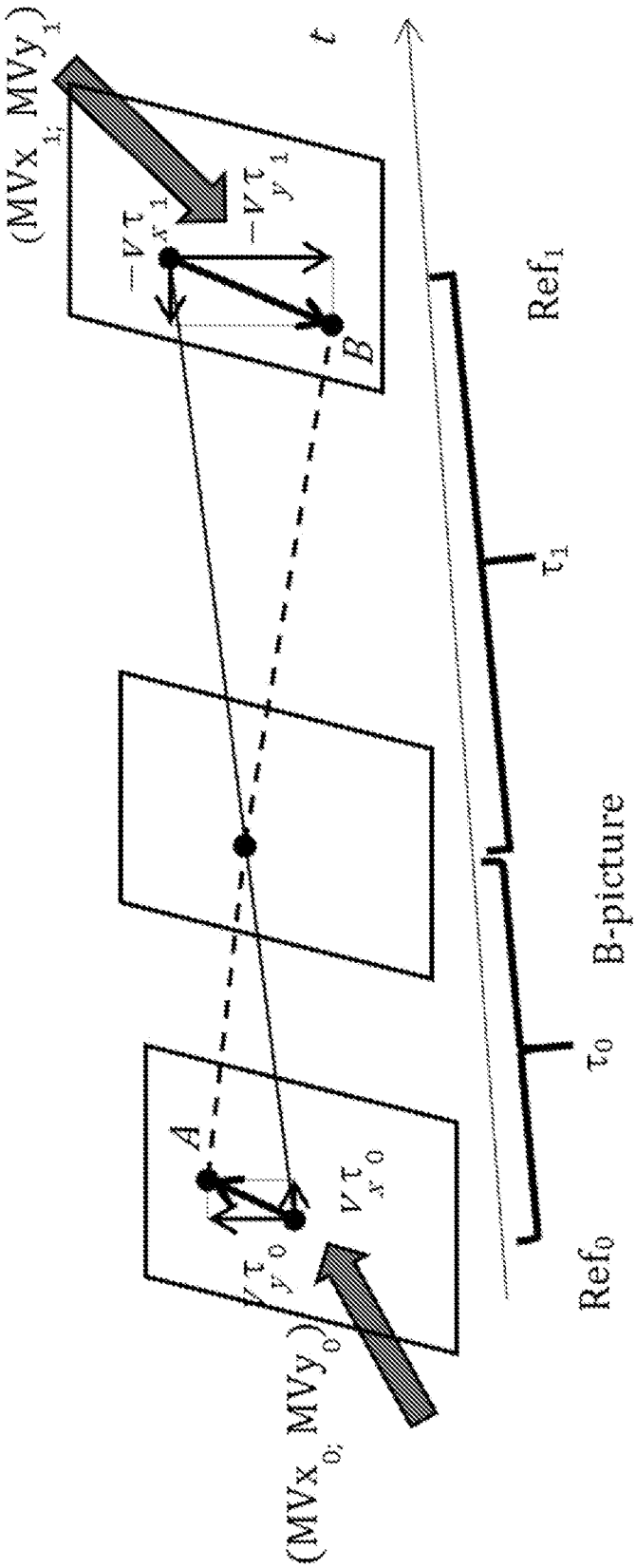

FIG. 8

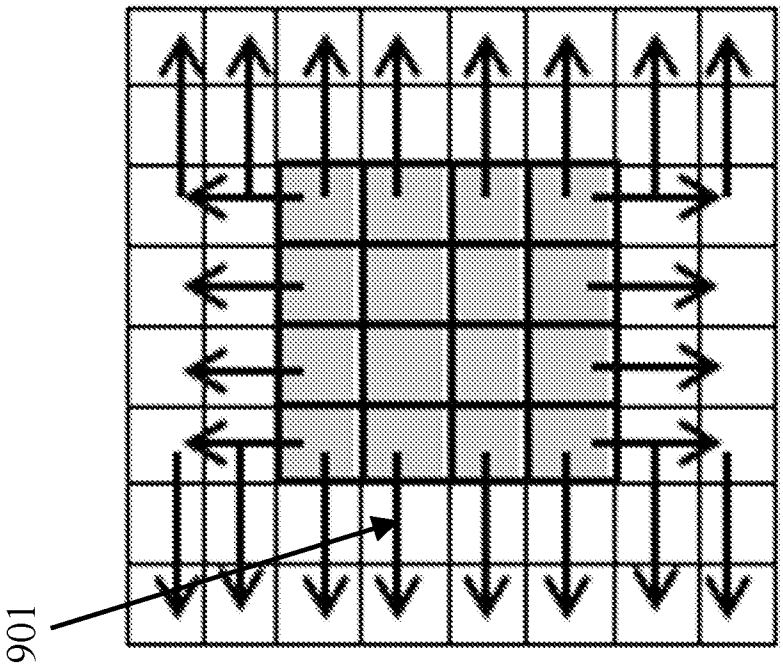

FIG. 9B

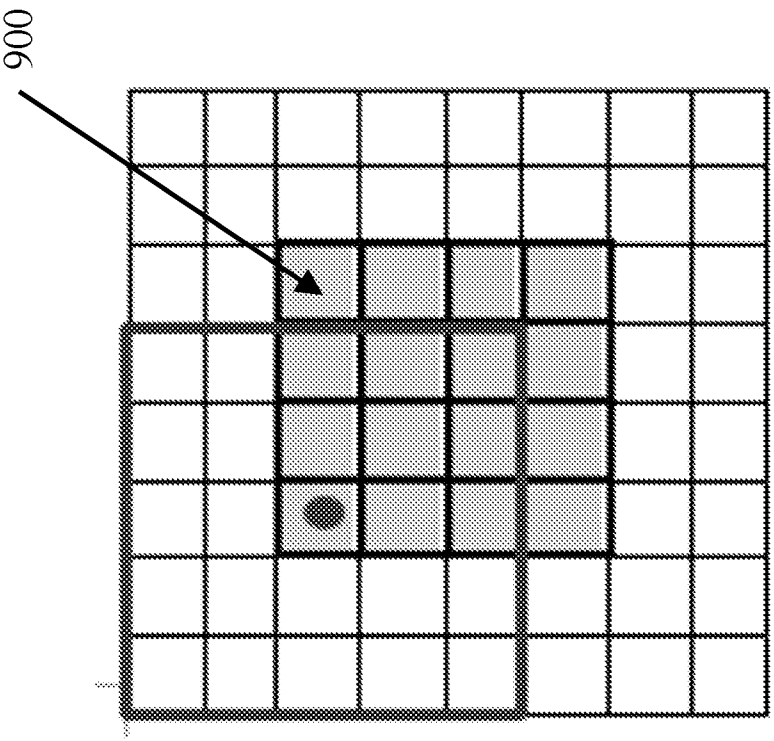

FIG. 9A

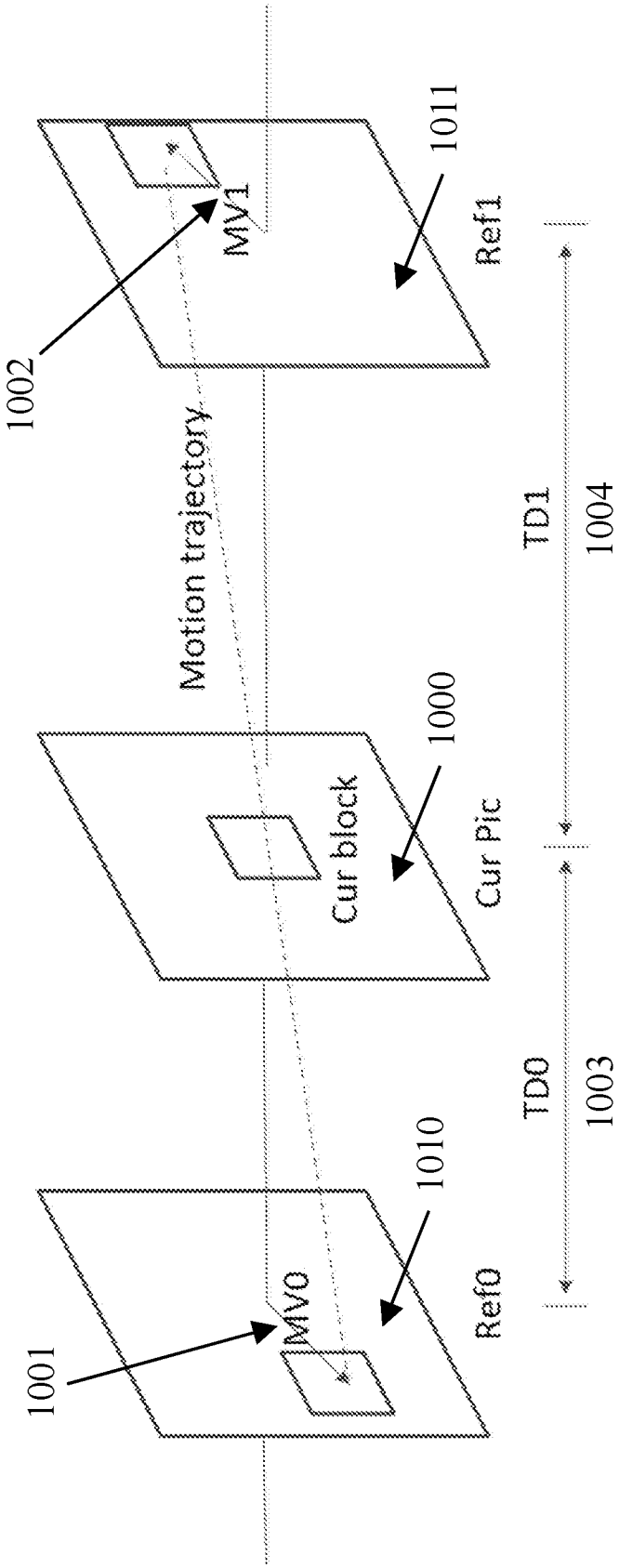
1002
MV1
1011
Ref1
Motion trajectory
TD1
1004
1000
Cur block
Cur Pic
TD0
1003
1001
MV0
1010
Ref0

FIG. 10

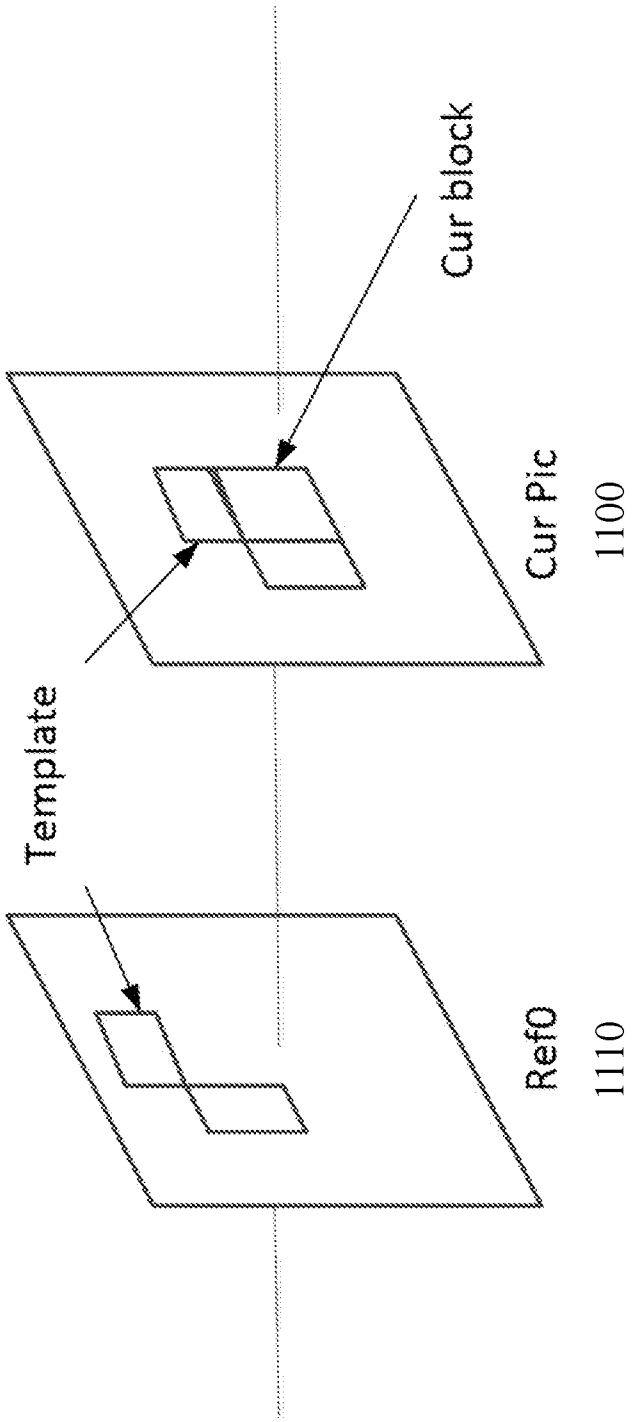

FIG. 11

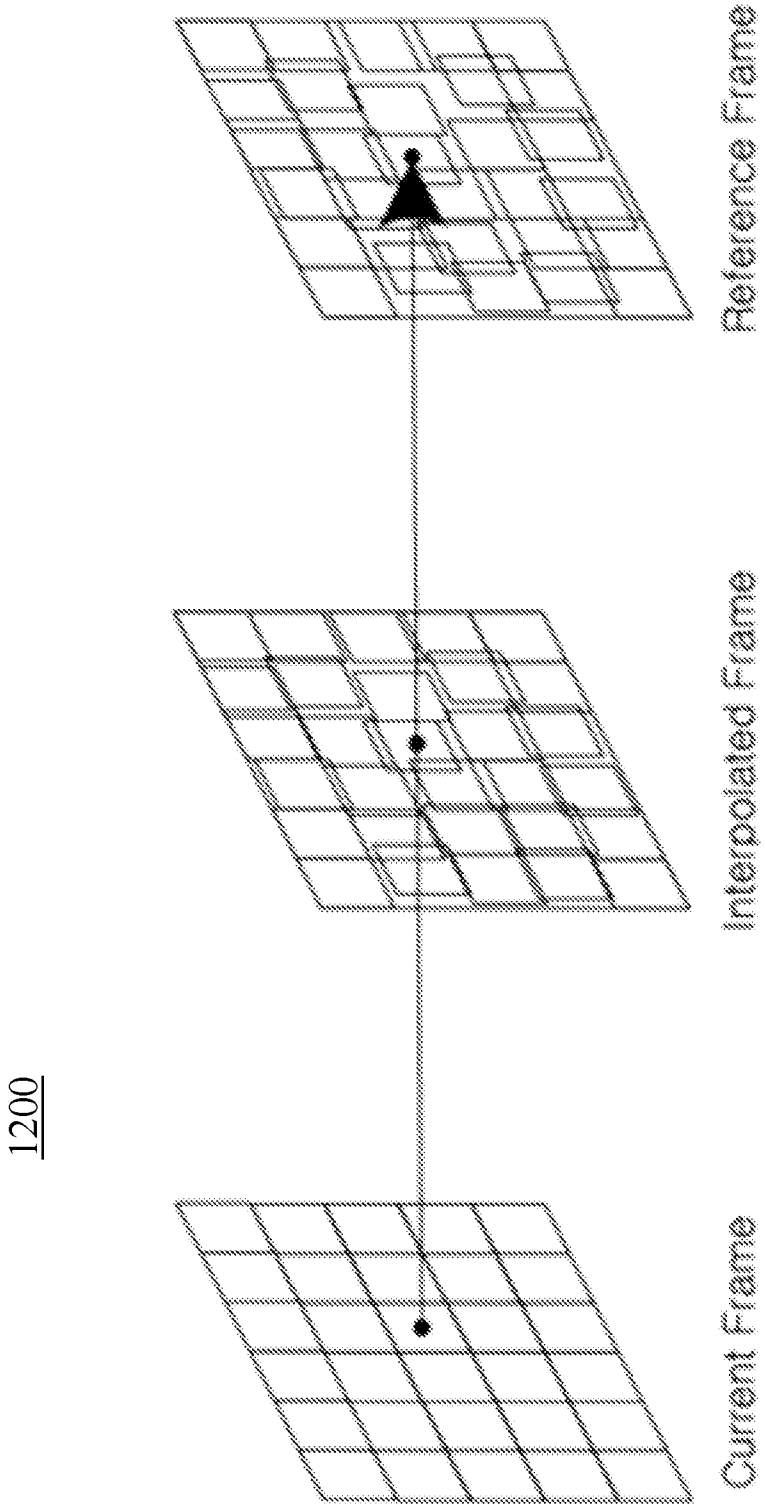

FIG. 12

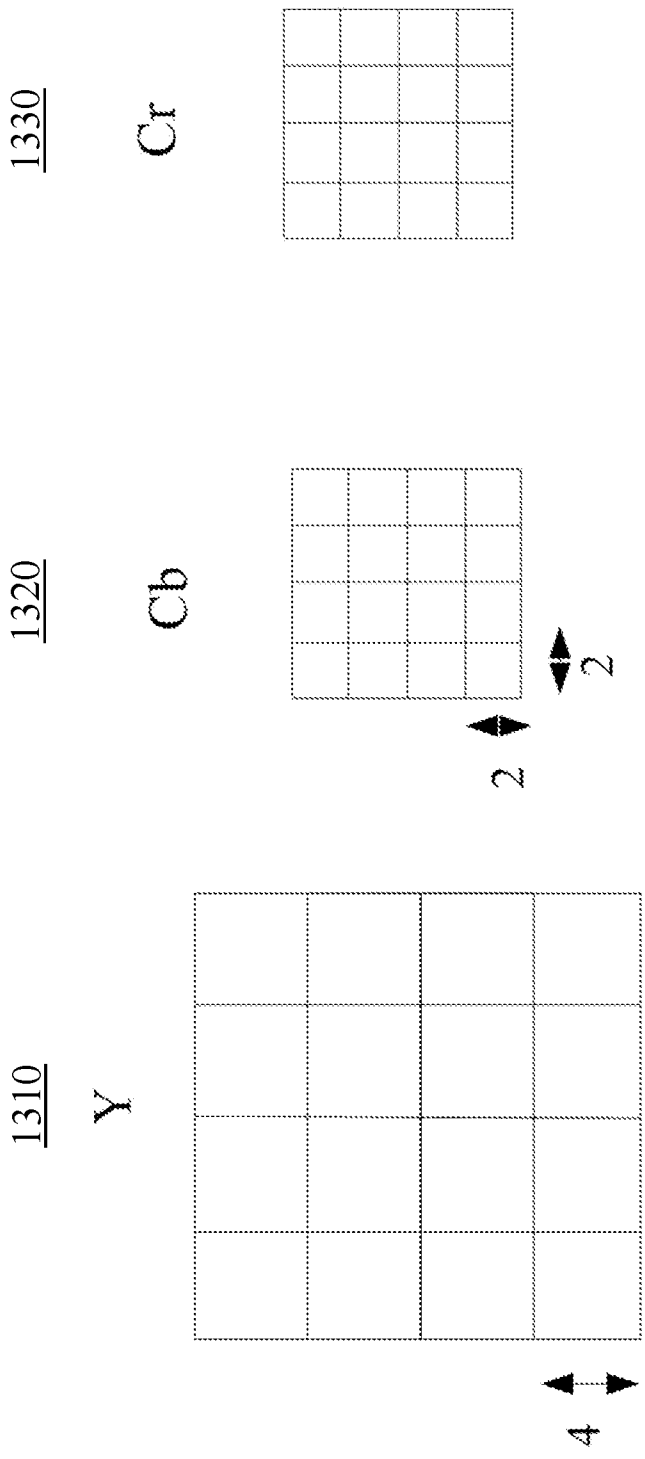

FIG. 13

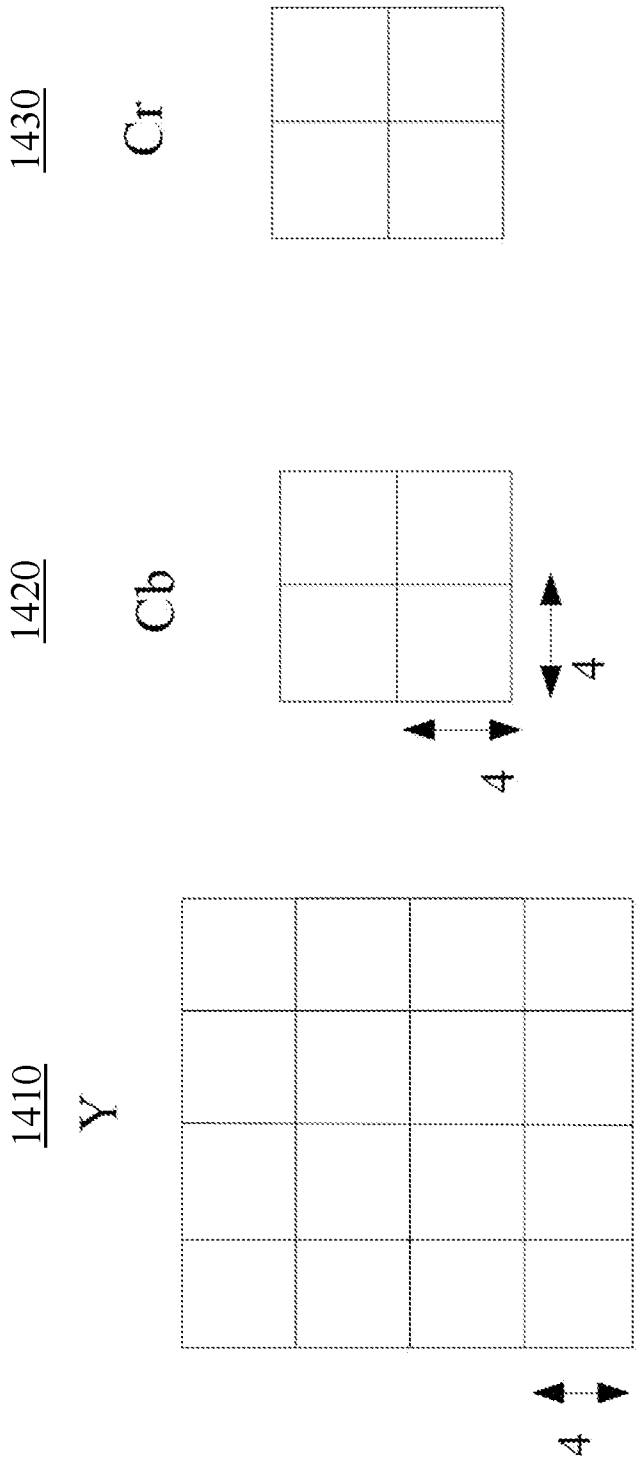

FIG. 14

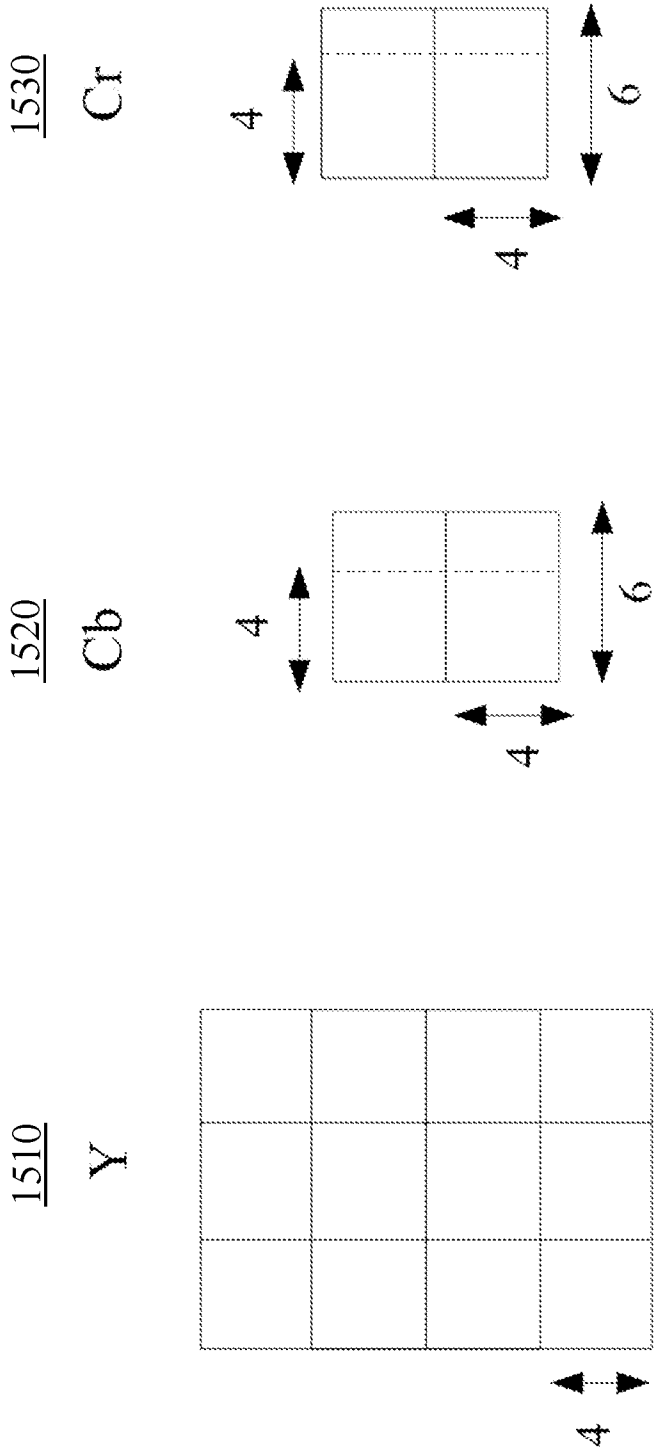

FIG. 15

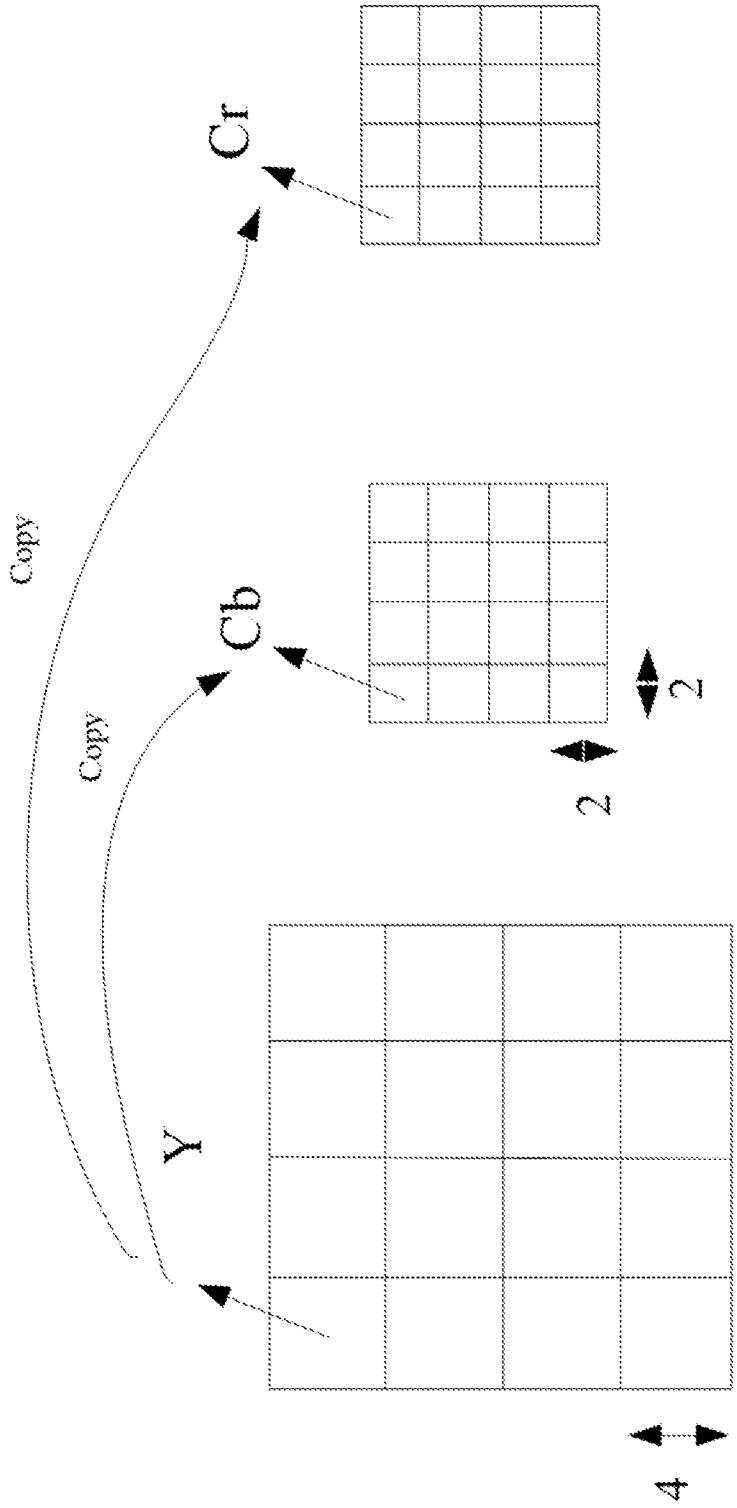

FIG. 16

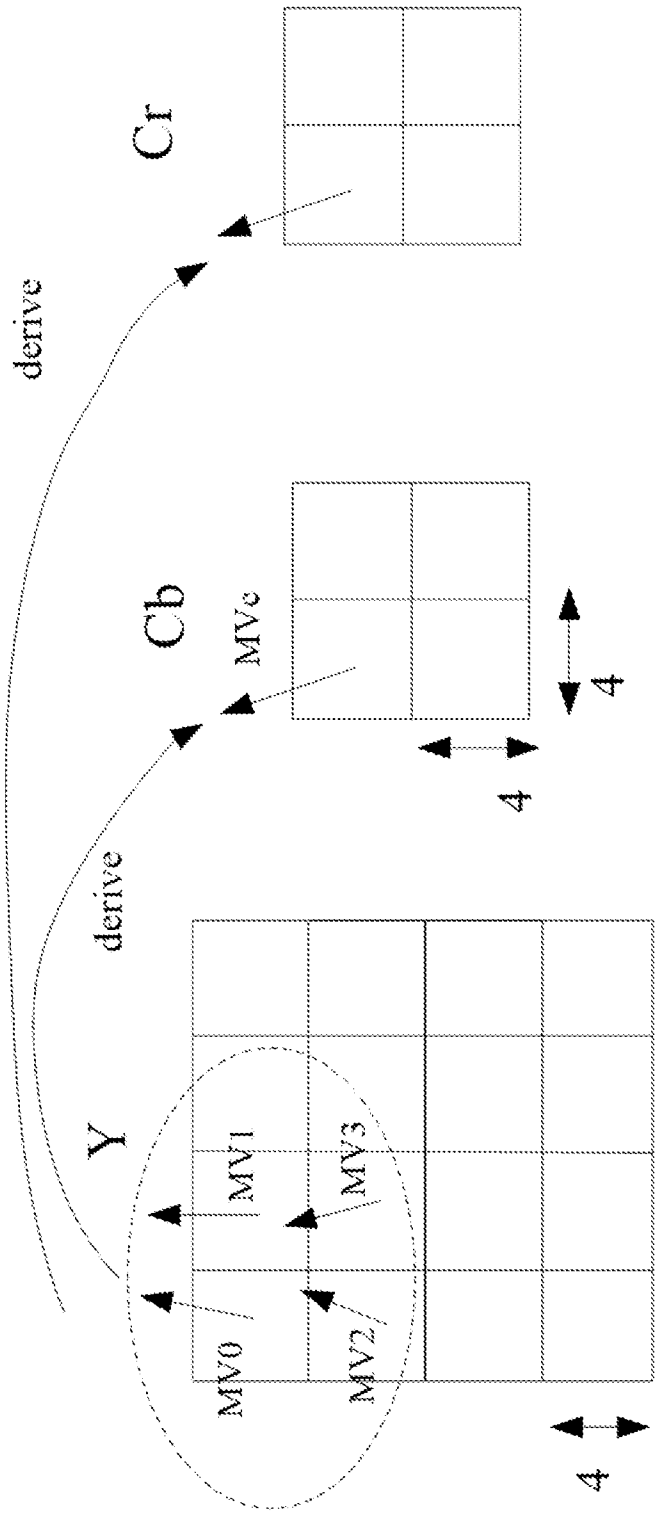

FIG. 17

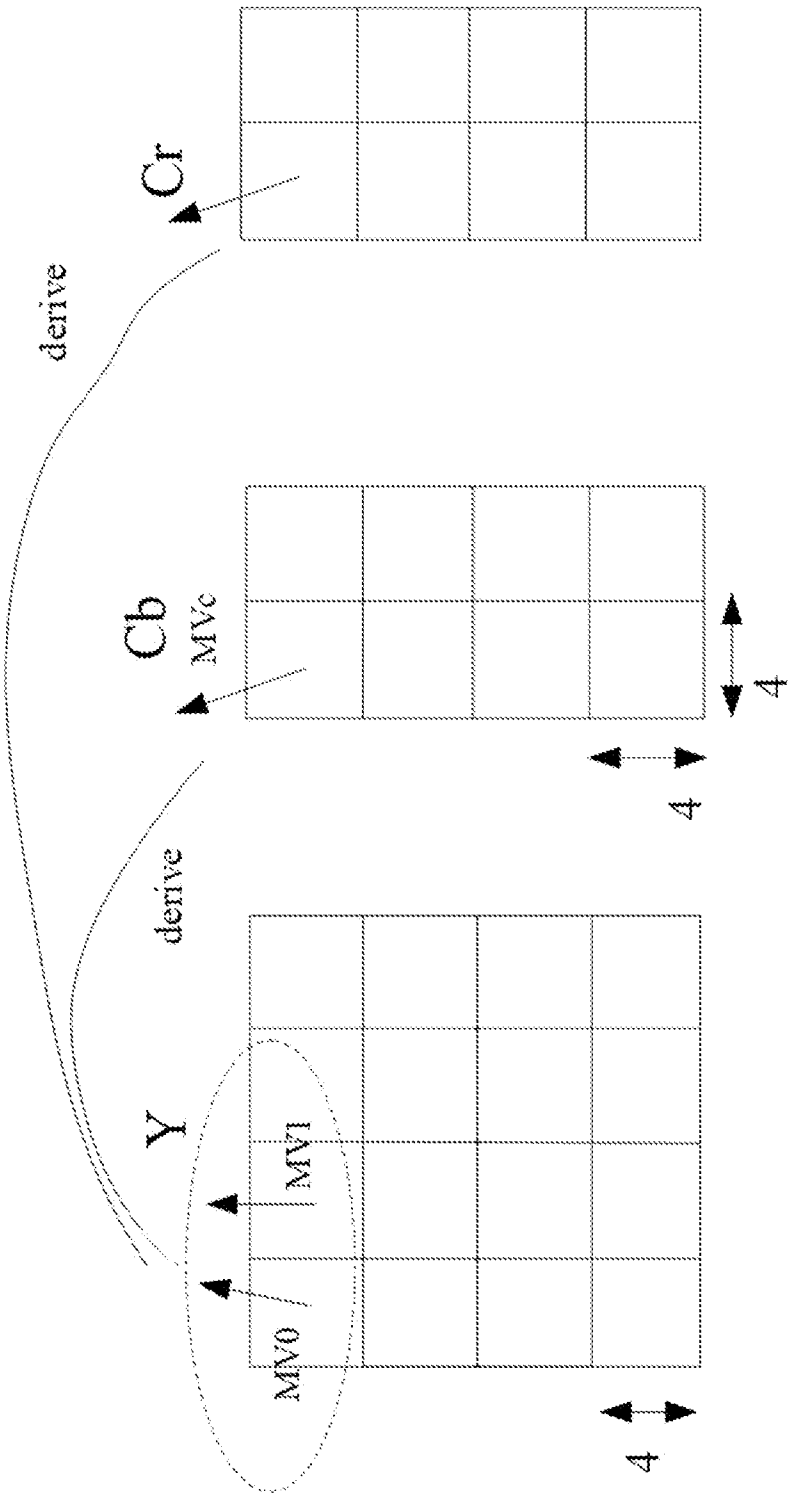

FIG. 18

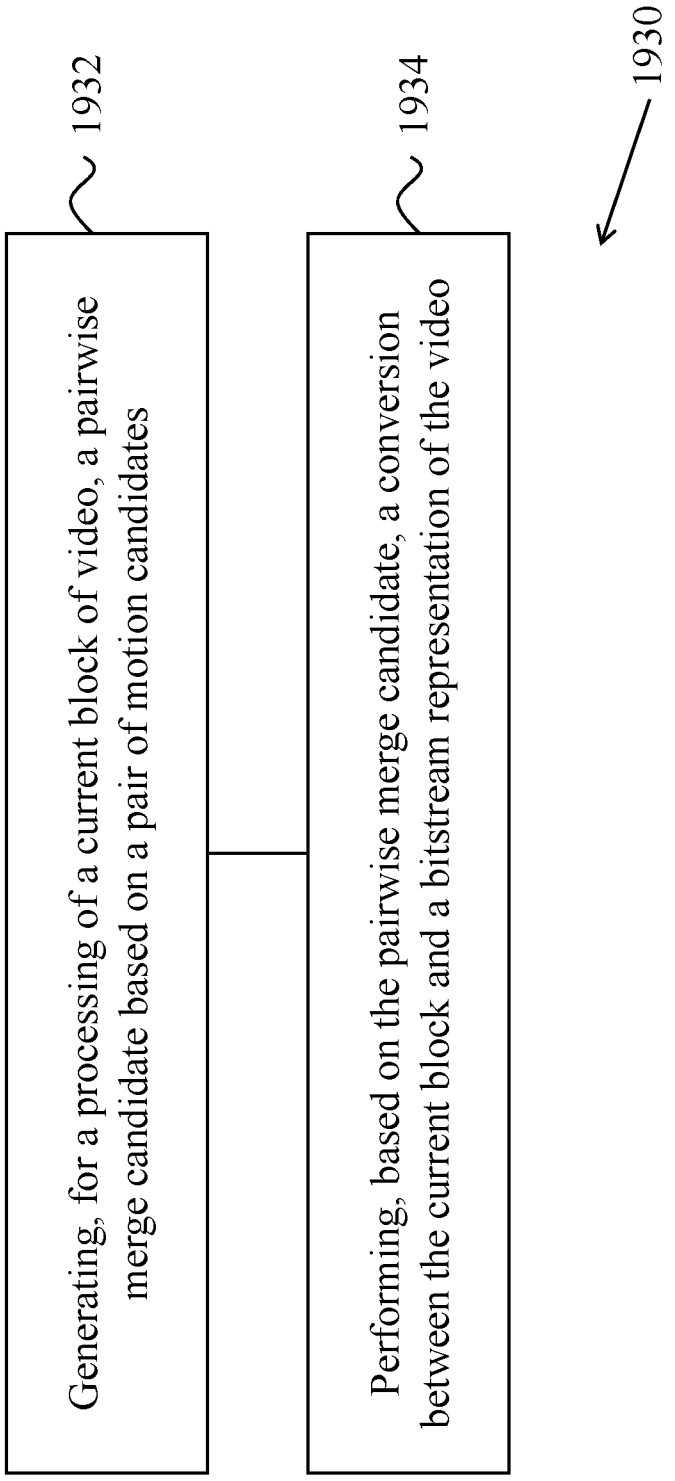
1930
1932
Generating, for a processing of a current block of video, a pairwise merge candidate based on a pair of motion candidates
1934
Performing, based on the pairwise merge candidate, a conversion between the current block and a bitstream representation of the video

FIG. 19A

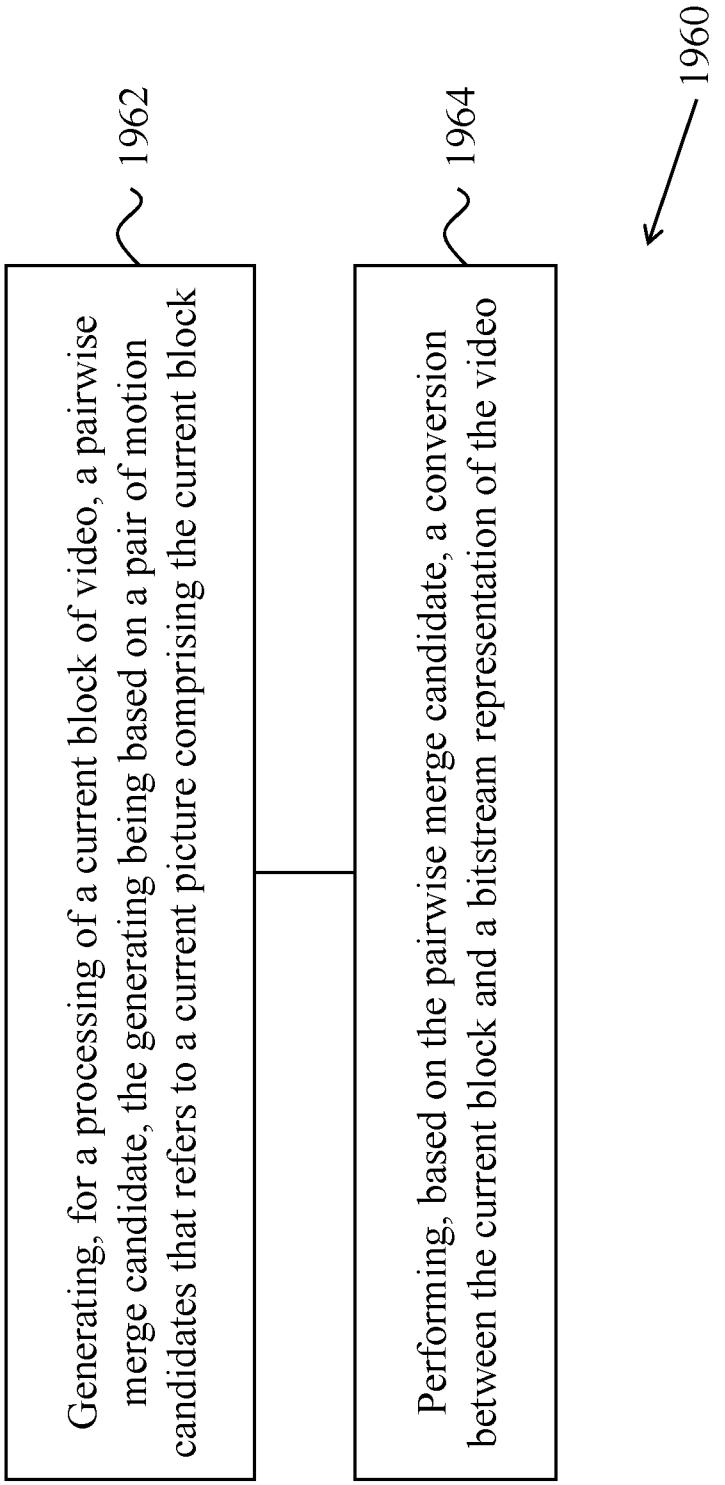
1960
1962
Generating, for a processing of a current block of video, a pairwise merge candidate, the generating being based on a pair of motion candidates that refers to a current picture comprising the current block
1964
Performing, based on the pairwise merge candidate, a conversion between the current block and a bitstream representation of the video

FIG. 19B

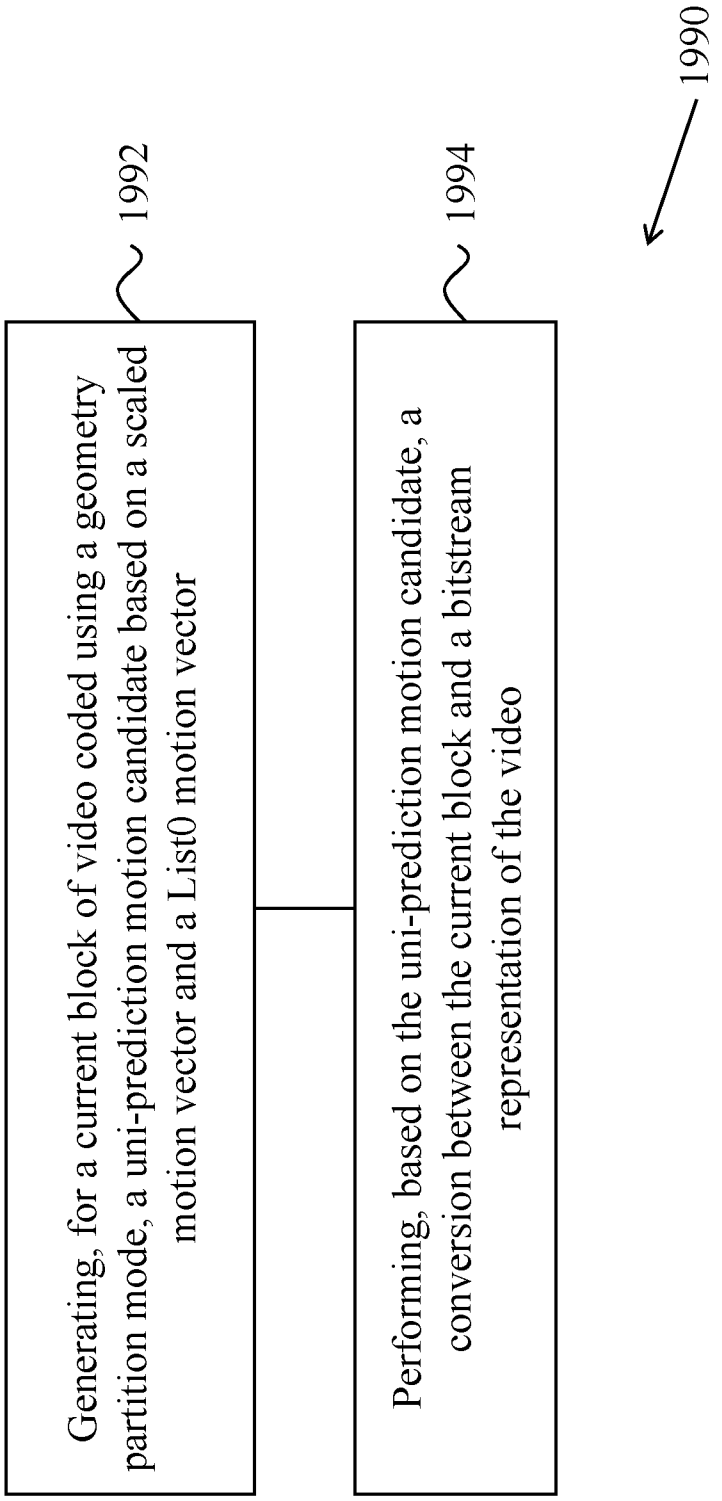
1990
1992
Generating, for a current block of video coded using a geometry partition mode, a uni-prediction motion candidate based on a scaled motion vector and a List0 motion vector
1994
Performing, based on the uni-prediction motion candidate, a conversion between the current block and a bitstream representation of the video

FIG. 19C

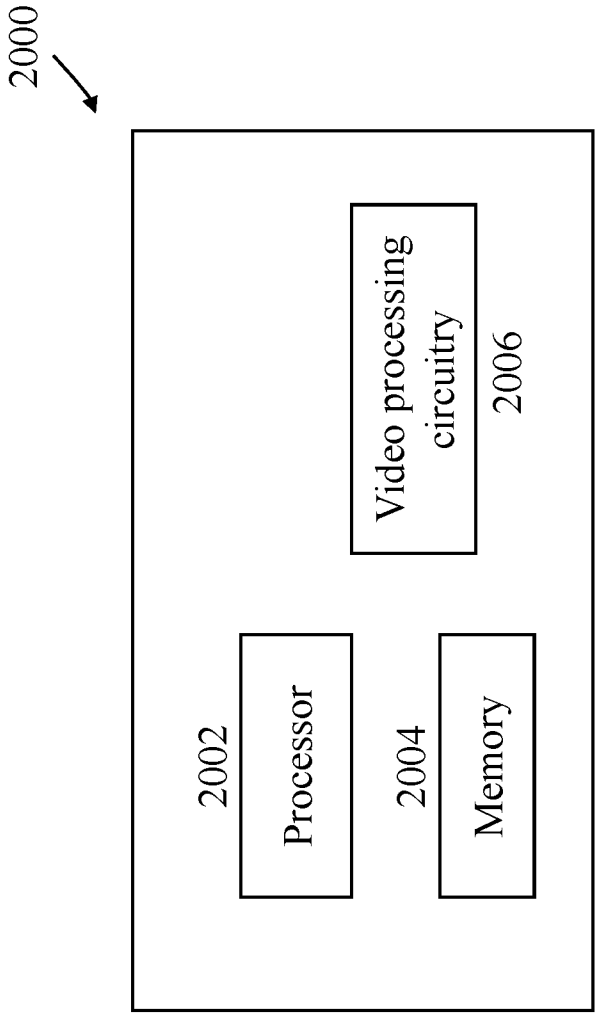
2000
2002
Processor
2004
Memory
Video processing circuitry
2006

FIG. 20

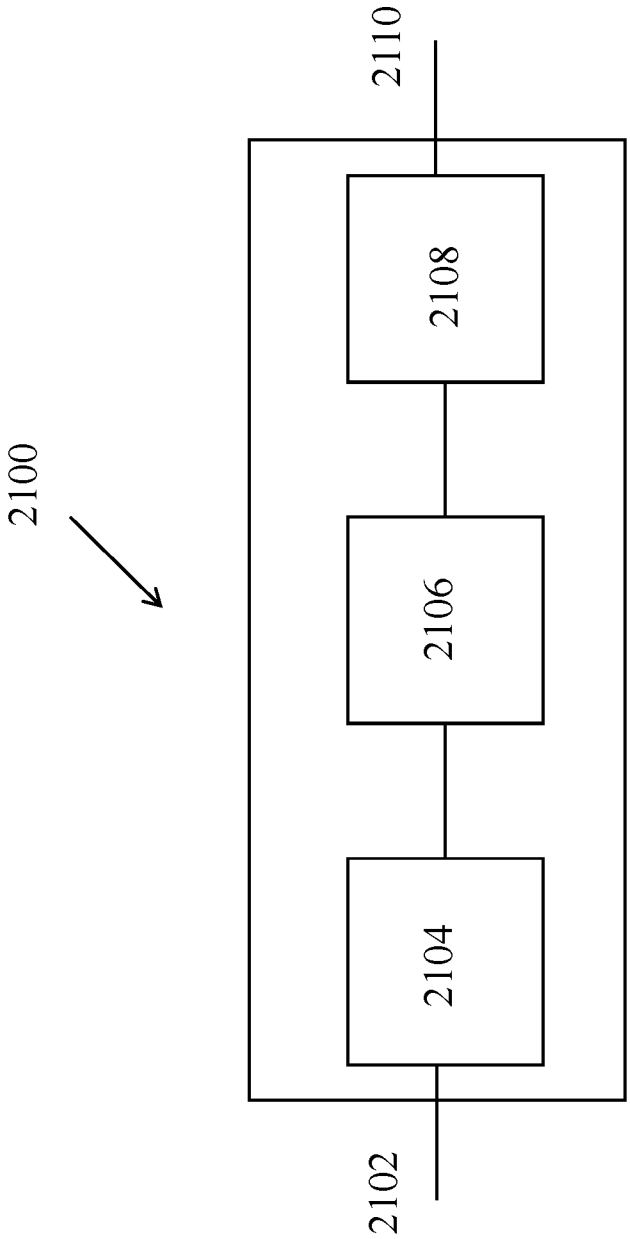

FIG. 21

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2019/117119

A. CLASSIFICATION OF SUBJECT MATTER

H04N 19/10(2014.01)i

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

H04N

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

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

CNPAT;CNKI;WPI;EPODOC;IEEE:BYTEDANCE, motion, pairwise, pair+, merge, video, image, picture, block, candidate, shift, offset, vector, represent+, second, reference, round+

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	HSIAO, Yu-Ling et.al. "CE4.4.12: Pairwise average candidates" *Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 1112th Meeting: Macao, CN, 3–12 Oct. 2018,* 12 October 2018 (2018-10-12), sections 1-2	1-17
A	ZHANG, Yan et.al. "Adaptive Motion Vector Resolution Rounding Align" *Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11 12th Meeting: Macao, CN, 3–12 Oct. 2018,* 12 October 2018 (2018-10-12), the whole document	1-17
A	US 2013101041 A1 (IMAGINATION TECHNOLOGIES, LTD.) 25 April 2013 (2013-04-25) the whole document	1-17
A	CN 108632629 A (KT CORP.) 09 October 2018 (2018-10-09) the whole document	1-17
A	CN 104053005 A (INTEL CORP.) 17 September 2014 (2014-09-17) the whole document	1-17
A	WO 2017157281 A1 (MEDIATEK INC.) 21 September 2017 (2017-09-21) the whole document	1-17

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

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

Date of the actual completion of the international search	Date of mailing of the international search report
02 January 2020	**23 January 2020**
Name and mailing address of the ISA/CN **National Intellectual Property Administration, PRC** **6, Xitucheng Rd., Jimen Bridge, Haidian District, Beijing 100088** **China** Facsimile No. **(86-10)62019451**	Authorized officer **WU,Qian** Telephone No. **86-（010）-53961822**

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2019/117119

Patent document cited in search report			Publication date (day/month/year)	Patent family member(s)			Publication date (day/month/year)
US	2013101041	A1	25 April 2013	EP	2555522	A2	06 February 2013
				GB	2495179	B	14 August 2013
				EP	2555522	A3	04 January 2017
				GB	201213785	D0	12 September 2012
				CN	102917197	A	06 February 2013
				GB	201113527	D0	21 September 2011
				CN	102917197	B	02 November 2016
				US	8929451	B2	06 January 2015
				GB	2495179	A	03 April 2013
CN	108632629	A	09 October 2018	CN	106105191	A	09 November 2016
				WO	2015142054	A1	24 September 2015
				CN	108989815	A	11 December 2018
				CN	108683922	A	19 October 2018
				CN	106105191	B	10 August 2018
				KR	20150109282	A	01 October 2015
				CN	108965888	A	07 December 2018
				US	2017006302	A1	05 January 2017
				US	10257531	B2	09 April 2019
CN	104053005	A	17 September 2014	CN	104053005	B	06 November 2018
				GB	2514441	B	28 September 2016
				GB	2514441	A	26 November 2014
				US	2014254678	A1	11 September 2014
				GB	201403951	D0	23 April 2014
WO	2017157281	A1	21 September 2017	US	2019082192	A1	14 March 2019
				CN	108781295	A	09 November 2018
				TW	201739256	A	01 November 2017