



(51) International Patent Classification:

H04N 19/176 (2014.01) *H04N 19/82* (2014.01)
H04N 19/577 (2014.01)

(21) International Application Number:

PCT/CN2020/082038

(22) International Filing Date:

30 March 2020 (30.03.2020)

(25) Filing Language:

English

(26) Publication Language:

English

(30) Priority Data:

PCT/CN2019/080356
29 March 2019 (29.03.2019) CN

(71) Applicants: **BEIJING BYTEDANCE NETWORK TECHNOLOGY CO., LTD.** [CN/CN]; Room B-0035, 2/F, No.3 Building, No.30, Shixing Road, Shijingshan District, Beijing 100041 (CN). **BYTEDANCE INC.** [US/US];

12655 West Jefferson Boulevard, Sixth Floor, Suite No. 137, Los Angeles, California 90066 (US).

(72) Inventors: **ZHANG, Li**; 12655 West Jefferson Boulevard, Sixth Floor, Suite No. 137, Los Angeles, California 90066 (US). **ZHANG, Kai**; 12655 West Jefferson Boulevard, Sixth Floor, Suite No. 137, Los Angeles, California 90066 (US). **CHUANG, Hsiao Chiang**; 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,

(54) Title: INTERACTIONS BETWEEN ADAPTIVE LOOP FILTERING AND OTHER CODING TOOLS

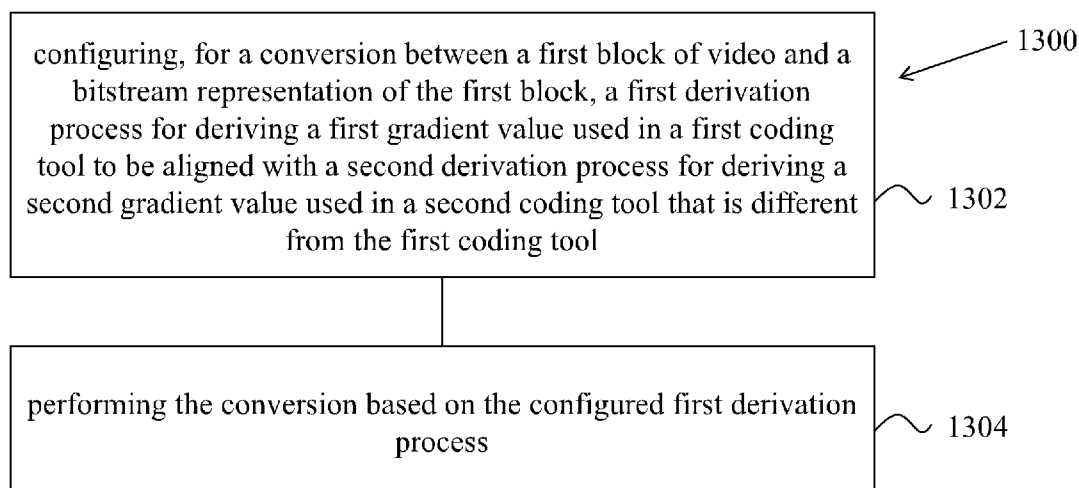


FIG. 13

(57) Abstract: Interactions between adaptive loop filtering and other coding tools are described. In an exemplary aspect, a method for video processing includes configuring, for a conversion between a first block of video and a bitstream representation of the first block, a first derivation process for deriving a first gradient value used in a first coding tool to be aligned with a second derivation process for deriving a second gradient value used in a second coding tool that is different from the first coding tool; and performing the conversion based on the configured first derivation process.



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

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

Declarations under Rule 4.17:

— *of inventorship (Rule 4.17(iv))*

Published:

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

INTERACTIONS BETWEEN ADAPTIVE LOOP FILTERING AND OTHER CODING TOOLS

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/CN2019/080356, filed on March 29, 2019. The entire disclosures of International Patent Application No. PCT/CN2019/080356 is incorporated by reference as part of the disclosure of this application.

TECHNICAL FIELD

[0002] This patent document relates to video coding techniques, devices and systems.

BACKGROUND

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

SUMMARY

[0004] Devices, systems and methods related to digital video coding, and specifically, to interactions between adaptive loop filtering and other coding tools are described. The described methods may be applied to both the existing video coding standards (e.g., High Efficiency Video Coding (HEVC)) and future video coding standards (e.g., Versatile Video Coding (VVC)) or codecs.

[0005] In one representative aspect, the disclosed technology may be used to provide a method for video processing. This method configuring, for a current video block, a first derivation process for deriving a first gradient value used in a first coding tool based on a second derivation process for deriving a second gradient value used in a second coding tool that is different from the first coding tool, and reconstructing, based on the first derivation process and the first coding tool, the current video block from a corresponding bitstream representation,

wherein at least one of the first coding tool and the second coding tool relates to a pixel filtering process, and wherein the first gradient value and the second gradient value are indicative of a directional change in an intensity or a color component over a subset of samples in the current video block.

[0006] In another representative aspect, the disclosed technology may be used to provide a method for video processing. This method configuring, for a current video block, a first padding process used in a first coding tool based on a second padding process used in a second coding tool that is different from the first coding tool, and reconstructing, based on the first padding process and the first coding tool, the current video block from a corresponding bitstream representation, wherein the first padding process and the second padding process comprise adding out of range samples in a calculation of a gradient value that is indicative of a directional change in an intensity or a color component over a subset of samples in the current video block.

[0007] In one representative aspect, the disclosed technology may be used to provide a method for video processing. This method comprises: configuring, for a conversion between a first block of video and a bitstream representation of the first block, a first derivation process for deriving a first gradient value used in a first coding tool to be aligned with a second derivation process for deriving a second gradient value used in a second coding tool that is different from the first coding tool; and performing the conversion based on the configured first derivation process.

[0008] In one representative aspect, the disclosed technology may be used to provide a method for video processing. This method comprises: deriving, for a conversion between a first block of video and a bitstream representation of the first block, gradient values used in one or more coding tools by applying a sub-block level gradient calculation process, wherein the gradient values are derived for partial samples within prediction blocks of the first block; and performing the conversion based on the derived gradient values.

[0009] In one representative aspect, the disclosed technology may be used to provide a method for video processing. This method comprises: configuring, for a conversion between a first block of video and a bitstream representation of the first block, a first padding process in a first coding tool to be aligned with a second padding process in a second coding tool that is different from the first coding tool, wherein the first padding process is for padding samples out

of range used to derive gradient values used in the first coding tool, and the second padding process is for padding samples out of range used to derive gradient values used in the second coding tool; and performing the conversion based on the configured first padding process.

[0010] 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.

[0011] 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.

[0012] In yet another representative aspect, a video decoder apparatus may implement a method as described herein.

[0013] 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

[0014] FIG. 1 shows an example of an encoder block diagram for video coding.

[0015] FIGS. 2A, 2B and 2C show examples of geometry transformation-based adaptive loop filter (GALF) filter shapes.

[0016] FIG. 3 shows an example of a flow graph for a GALF encoder decision.

[0017] FIGS. 4A–4D show example subsampled Laplacian calculations for adaptive loop filter (ALF) classification.

[0018] FIG. 5 shows an example of a luma filter shape.

[0019] FIG. 6 shows an example of region division of a Wide Video Graphic Array (WVGA) sequence.

[0020] FIG. 7 shows an example of an optical flow trajectory used by the bi-directional optical flow (BIO) algorithm.

[0021] FIGS. 8A and 8B show example snapshots of using of the bi-directional optical flow (BIO) algorithm without block extensions.

[0022] FIG. 9 shows an example of the interpolated samples used in BIO.

[0023] FIG. 10 shows an example of prediction refinement with optical flow (PROF).

[0024] FIGS. 11A and 11B show flowcharts of example methods for interactions between

adaptive loop filtering and other coding tools, in accordance with the disclosed technology.

[0025] FIG. 12 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.

[0026] FIG. 13 shows a flowchart of yet another example method for video processing.

[0027] FIG. 14 shows a flowchart of yet another example method for video processing.

[0028] FIG. 15 shows a flowchart of yet another example method for video processing.

DETAILED DESCRIPTION

[0029] 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 (VVC) standard to be finalized, or other current and/or future video coding standards.

[0030] 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).

[0031] 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 color space and chroma subsampling

[0032] Color space, also known as the color model (or color system), is an abstract mathematical model which simply describes the range of colors as tuples of numbers, typically as 3 or 4 values or color components (e.g. RGB). Basically speaking, color space is an elaboration of the coordinate system and sub-space.

[0033] For video compression, the most frequently used color spaces are YCbCr and RGB.

[0034] YCbCr, Y'CbCr, or Y Pb/Cb Pr/Cr, also written as YCBCR or Y'CBCR, is a family of color spaces used as a part of the color image pipeline in video and digital photography systems. Y' is the luma component and CB and CR are the blue-difference and red-difference chroma components. Y' (with prime) is distinguished from Y, which is luminance, meaning that light intensity is nonlinearly encoded based on gamma corrected RGB primaries.

[0035] Chroma subsampling is the practice of encoding images by implementing less resolution for chroma information than for luma information, taking advantage of the human visual system's lower acuity for color differences than for luminance.

1.1 The 4:4:4 color format

[0036] Each of the three Y'CbCr components have the same sample rate, thus there is no chroma subsampling. This scheme is sometimes used in high-end film scanners and cinematic post production.

1.2 The 4:2:2 color format

[0037] The two chroma components are sampled at half the sample rate of luma, e.g. the horizontal chroma resolution is halved. This reduces the bandwidth of an uncompressed video signal by one-third with little to no visual difference.

1.3 The 4:2:0 color format

[0038] In 4:2:0, the horizontal sampling is doubled compared to 4:1:1, but as the Cb and Cr channels are only sampled on each alternate line in this scheme, the vertical resolution is halved. The data rate is thus the same. Cb and Cr are each subsampled at a factor of 2 both horizontally and vertically. There are three variants of 4:2:0 schemes, having different horizontal and vertical siting.

[0039] ◦ In MPEG-2, Cb and Cr are cosited horizontally. Cb and Cr are sited between

pixels in the vertical direction (sited interstitially).

[0040] ○ In JPEG/JFIF, H.261, and MPEG-1, Cb and Cr are sited interstitially, halfway between alternate luma samples.

[0041] ○ In 4:2:0 DV, Cb and Cr are co-sited in the horizontal direction. In the vertical direction, they are co-sited on alternating lines.

2 **Examples of the coding flow of a typical video codec**

[0042] FIG. 1 shows an example of encoder block diagram of VVC, which contains three in-loop filtering blocks: deblocking filter (DF), sample adaptive offset (SAO) and ALF. Unlike DF, which uses predefined filters, SAO and ALF utilize the original samples of the current picture to reduce the mean square errors between the original samples and the reconstructed samples by adding an offset and by applying a finite impulse response (FIR) filter, respectively, with coded side information signaling the offsets and filter coefficients. ALF is located at the last processing stage of each picture and can be regarded as a tool trying to catch and fix artifacts created by the previous stages.

3 **Examples of a geometry transformation-based adaptive loop filter in JEM**

[0043] In the JEM, an geometry transformation-based adaptive loop filter (GALF) with block-based filter adaption is applied. For the luma component, one among 25 filters is selected for each 2×2 block, based on the direction and activity of local gradients.

3.1 **Examples of filter shape**

[0044] In the JEM, up to three diamond filter shapes (as shown in FIGS. 2A, 2B and 2C for the 5×5 diamond, 7×7 diamond and 9×9 diamond, respectively) can be selected for the luma component. An index is signalled at the picture level to indicate the filter shape used for the luma component. For chroma components in a picture, the 5×5 diamond shape is always used.

3.1.1 **Block classification**

Each 2×2 block is categorized into one out of 25 classes. The classification index C is derived based on its directionality D and a quantized value of activity $\hat{A}$, as follows:

$$C = 5D + \hat{A}. \quad (1)$$

To calculate D and $\hat{A}$, gradients of the horizontal, vertical and two diagonal direction are first calculated using 1-D Laplacian:

$$g_v = \sum_{k=i-2}^{i+3} \sum_{l=j-2}^{j+3} V_{k,l}, \quad V_{k,l} = |2R(k,l) - R(k,l-1) - R(k,l+1)|, \quad (2)$$

$$g_h = \sum_{k=i-2}^{i+3} \sum_{l=j-2}^{j+3} H_{k,l}, \quad H_{k,l} = |2R(k,l) - R(k-1,l) - R(k+1,l)|, \quad (3)$$

$$g_{d1} = \sum_{k=i-2}^{i+3} \sum_{l=j-3}^{j+3} D1_{k,l}, \quad D1_{k,l} = |2R(k,l) - R(k-1,l-1) - R(k+1,l+1)| \quad (4)$$

$$g_{d2} = \sum_{k=i-2}^{i+3} \sum_{l=j-2}^{j+3} D2_{k,l}, \quad D2_{k,l} = |2R(k,l) - R(k-1,l+1) - R(k+1,l-1)| \quad (5)$$

Indices i and j refer to the coordinates of the upper left sample in the 2×2 block and $R(i,j)$ indicates a reconstructed sample at coordinate (i,j) .

Then D maximum and minimum values of the gradients of horizontal and vertical directions are set as:

$$g_{h,v}^{max} = \max(g_h, g_v), \quad g_{h,v}^{min} = \min(g_h, g_v), \quad (6)$$

and the maximum and minimum values of the gradient of two diagonal directions are set as:

$$g_{d0,d1}^{max} = \max(g_{d0}, g_{d1}), \quad g_{d0,d1}^{min} = \min(g_{d0}, g_{d1}), \quad (7)$$

To derive the value of the directionality D , these values are compared against each other and with two thresholds t_1 and t_2 :

Step 1. If both $g_{h,v}^{max} \leq t_1 \cdot g_{h,v}^{min}$ and $g_{d0,d1}^{max} \leq t_1 \cdot g_{d0,d1}^{min}$ are true, D is set to 0.

Step 2. If $g_{h,v}^{max} / g_{h,v}^{min} > g_{d0,d1}^{max} / g_{d0,d1}^{min}$, continue from Step 3; otherwise continue from Step 4.

Step 3. If $g_{h,v}^{max} > t_2 \cdot g_{h,v}^{min}$, D is set to 2; otherwise D is set to 1.

Step 4. If $g_{d0,d1}^{max} > t_2 \cdot g_{d0,d1}^{min}$, D is set to 4; otherwise D is set to 3.

The activity value A is calculated as:

$$A = \sum_{k=i-2}^{i+3} \sum_{l=j-2}^{j+3} (V_{k,l} + H_{k,l}). \quad (8)$$

A is further quantized to the range of 0 to 4, inclusively, and the quantized value is denoted as $\hat{A}$. For both chroma components in a picture, no classification method is applied, i.e. a single set of ALF coefficients is applied for each chroma component.

3.1.2 Geometric transformations of filter coefficients

[0045] Before filtering each 2×2 block, geometric transformations such as rotation or diagonal and vertical flipping are applied to the filter coefficients $f(k, l)$ depending on gradient values calculated for that block. This is equivalent to applying these transformations to the samples in the filter support region. The idea is to make different blocks to which ALF is applied more similar by aligning their directionality.

[0046] Three geometric transformations, including diagonal, vertical flip and rotation are introduced:

$$\begin{aligned} \text{Diagonal: } f_D(k, l) &= f(l, k), \\ \text{Vertical flip: } f_V(k, l) &= f(k, K - l - 1), \\ \text{Rotation: } f_R(k, l) &= f(K - l - 1, k). \end{aligned} \quad (9)$$

[0047] Herein, K is the size of the filter and $0 \leq k, l \leq K - 1$ are coefficients coordinates, such that location $(0,0)$ is at the upper left corner and location $(K - 1, K - 1)$ is at the lower right corner. The transformations are applied to the filter coefficients $f(k, l)$ depending on gradient values calculated for that block. The relationship between the transformation and the four gradients of the four directions are summarized in Table 1.

Table 1: Mapping of the gradient calculated for one block and the transformations

Gradient values	Transformation
$g_{d2} < g_{d1}$ and $g_h < g_v$	No transformation
$g_{d2} < g_{d1}$ and $g_v < g_h$	Diagonal
$g_{d1} < g_{d2}$ and $g_h < g_v$	Vertical flip
$g_{d1} < g_{d2}$ and $g_v < g_h$	Rotation

3.1.3 Signaling of filter parameters

[0048] In the JEM, GALF filter parameters are signaled for the first CTU, i.e., after the slice header and before the SAO parameters of the first CTU. Up to 25 sets of luma filter coefficients could be signaled. To reduce bits overhead, filter coefficients of different classification can be merged. Also, the GALF coefficients of reference pictures are stored and allowed to be reused as

GALF coefficients of a current picture. The current picture may choose to use GALF coefficients stored for the reference pictures, and bypass the GALF coefficients signaling. In this case, only an index to one of the reference pictures is signaled, and the stored GALF coefficients of the indicated reference picture are inherited for the current picture.

[0049] To support GALF temporal prediction, a candidate list of GALF filter sets is maintained. At the beginning of decoding a new sequence, the candidate list is empty. After decoding one picture, the corresponding set of filters may be added to the candidate list. Once the size of the candidate list reaches the maximum allowed value (i.e., 6 in current JEM), a new set of filters overwrites the oldest set in decoding order, and that is, first-in-first-out (FIFO) rule is applied to update the candidate list. To avoid duplications, a set could only be added to the list when the corresponding picture doesn't use GALF temporal prediction. To support temporal scalability, there are multiple candidate lists of filter sets, and each candidate list is associated with a temporal layer. More specifically, each array assigned by temporal layer index (TempIdx) may compose filter sets of previously decoded pictures with equal to lower TempIdx. For example, the k-th array is assigned to be associated with TempIdx equal to k, and it only contains filter sets from pictures with TempIdx smaller than or equal to k. After coding a certain picture, the filter sets associated with the picture will be used to update those arrays associated with equal or higher TempIdx.

[0050] Temporal prediction of GALF coefficients is used for inter coded frames to minimize signaling overhead. For intra frames, temporal prediction is not available, and a set of 16 fixed filters is assigned to each class. To indicate the usage of the fixed filter, a flag for each class is signaled and if required, the index of the chosen fixed filter. Even when the fixed filter is selected for a given class, the coefficients of the adaptive filter $f(k,l)$ can still be sent for this class in which case the coefficients of the filter which will be applied to the reconstructed image are sum of both sets of coefficients.

[0051] The filtering process of luma component can be controlled at CU level. A flag is signaled to indicate whether GALF is applied to the luma component of a CU. For chroma component, whether GALF is applied or not is indicated at picture level only.

3.1.4 Filtering process

[0052] At decoder side, when GALF is enabled for a block, each sample $R(i,j)$ within the

block is filtered, resulting in sample value $R'(i, j)$ as shown below, where L denotes filter length, $f_{m,n}$ represents filter coefficient, and $f(k, l)$ denotes the decoded filter coefficients.

$$R'(i, j) = \sum_{k=-L/2}^{L/2} \sum_{l=-L/2}^{L/2} f(k, l) \times R(i + k, j + l) \quad (10)$$

3.1.5 Determination process for encoder side filter parameters

[0053] Overall encoder decision process for GALF is illustrated in FIG. 3. For luma samples of each CU, the encoder makes a decision on whether or not the GALF is applied and the appropriate signalling flag is included in the slice header. For chroma samples, the decision to apply the filter is done based on the picture-level rather than CU-level. Furthermore, chroma GALF for a picture is checked only when luma GALF is enabled for the picture.

4 Examples of a geometry transformation-based adaptive loop filter in VVC

[0054] The current design of GALF in VVC has the following major changes compared to that in JEM:

- 1) The adaptive filter shape is removed. Only 7x7 filter shape is allowed for luma component and 5x5 filter shape is allowed for chroma component.
- 2) Temporal prediction of ALF parameters and prediction from fixed filters are both removed.
- 3) For each CTU, one bit flag is signaled whether ALF is enabled or disabled.
- 4) Calculation of class index is performed in 4x4 level instead of 2x2. In addition, as proposed, sub-sampled Laplacian calculation method for ALF classification is utilized. More specifically, there is no need to calculate the horizontal/vertical/45 diagonal /135 degree gradients for each sample within one block. Instead, 1:2 subsampling is utilized.

5 Examples of a region-based adaptive loop filter in AVS2

[0055] ALF is the last stage of in-loop filtering. There are two stages in this process. The first stage is filter coefficient derivation. To train the filter coefficients, the encoder classifies reconstructed pixels of the luminance component into 16 regions, and one set of filter coefficients is trained for each category using wiener-hopf equations to minimize the mean squared error between the original frame and the reconstructed frame. To reduce the redundancy between these 16 sets of filter coefficients, the encoder will adaptively merge them based on the rate-distortion performance. At its maximum, 16 different filter sets can be assigned for the luminance component and only one for the chrominance components. The second stage is a filter

decision, which includes both the frame level and LCU level. Firstly the encoder decides whether frame-level adaptive loop filtering is performed. If frame level ALF is on, then the encoder further decides whether the LCU level ALF is performed.

5.1 Filter shape

[0056] The filter shape adopted in AVS-2 is a 7×7 cross shape superposing a 3×3 square shape, just as illustrated in FIG. 5 for both luminance and chroma components. Each square in FIG. 5 corresponds to a sample. Therefore, a total of 17 samples are used to derive a filtered value for the sample of position C8. Considering overhead of transmitting the coefficients, a point-symmetrical filter is utilized with only nine coefficients left, $\{C_0, C_1, \dots, C_8\}$, which reduces the number of filter coefficients to half as well as the number of multiplications in filtering. The point-symmetrical filter can also reduce half of the computation for one filtered sample, e.g., only 9 multiplications and 14 add operations for one filtered sample.

5.2 Region-based adaptive merge

[0057] In order to adapt different coding errors, AVS-2 adopts region-based multiple adaptive loop filters for luminance component. The luminance component is divided into 16 roughly-equal-size basic regions where each basic region is aligned with largest coding unit (LCU) boundaries as shown in FIG. 6, and one Wiener filter is derived for each region. The more filters are used, the more distortions are reduced, but the bits used to encode these coefficients increase along with the number of filters. In order to achieve the best rate-distortion performance, these regions can be merged into fewer larger regions, which share the same filter coefficients. In order to simplify the merging process, each region is assigned with an index according to a modified Hilbert order based on the image prior correlations. Two regions with successive indices can be merged based on rate-distortion cost.

[0058] The mapping information between regions should be signaled to the decoder. In AVS-2, the number of basic regions is used to represent the merge results and the filter coefficients are compressed sequentially according to its region order. For example, when $\{0, 1\}$, $\{2, 3, 4\}$, $\{5, 6, 7, 8, 9\}$ and the left basic regions merged into one region respectively, only three integers are coded to represent this merge map, i.e., 2, 3, 5.

5.3 Signaling of side information

[0059] Multiple switch flags are also used. The sequence switch flag,

adaptive_loop_filter_enable, is used to control whether adaptive loop filter is applied for the whole sequence. The image switch flags, picture_alf_enble[i], control whether ALF is applied for the corresponding ith image component. Only if the picture_alf_enble[i] is enabled, the corresponding LCU-level flags and filter coefficients for that color component will be transmitted. The LCU level flags, lcu_alf_enable[k], control whether ALF is enabled for the corresponding kth LCU, and are interleaved into the slice data. The decision of different level regulated flags is all based on the rate-distortion cost. The high flexibility further makes the ALF improve the coding efficiency much more significantly.

[0060] In some embodiments, and for a luma component, there could be up to 16 sets of filter coefficients.

[0061] In some embodiments, and for each chroma component (Cb and Cr), one set of filter coefficients may be transmitted.

6 Bi-directional optical flow (BIO)

6.1 Overview and analysis of BIO

[0062] In BIO, motion compensation is first performed to generate the first predictions (in each prediction direction) of the current block. The first predictions are used to derive the spatial gradient, the temporal gradient and the optical flow of each sub-block or pixel within the block, which are then used to generate the second prediction, e.g., the final prediction of the sub-block or pixel. The details are described as follows.

[0063] 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.

[0064] Let $I^{(k)}$ be the luma value from reference k ($k=0, 1$) after block motion compensation, and denote $\partial I^{(k)}/\partial x$ and $\partial I^{(k)}/\partial y$ as the 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:

$$[0065] \quad \partial I^{(k)}/\partial t + v_x \partial I^{(k)}/\partial x + v_y \partial I^{(k)}/\partial y = 0. \quad \text{Eq. (11)}$$

[0066] Combining this optical flow equation with Hermite interpolation for the motion trajectory of each sample results in a unique third-order polynomial that matches both the function values $I^{(k)}$ and derivatives $\partial I^{(k)}/\partial x$ and $\partial I^{(k)}/\partial y$ at the ends. The value of this

polynomial at $t=0$ is the BIO prediction:

$$[0067] \quad pred_{BIO} = 1/2 \cdot (I^{(0)} + I^{(1)} + v_x / 2 \cdot (\tau_1 \partial I^{(1)} / \partial x - \tau_0 \partial I^{(0)} / \partial x) + v_y / 2 \cdot (\tau_1 \partial I^{(1)} / \partial y - \tau_0 \partial I^{(0)} / \partial y)) \quad \text{Eq. (12)}$$

[0068] FIG. 7 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 Ref₀ and Ref₁: $\tau_0 = \text{POC}(\text{current}) - \text{POC}(\text{Ref}_0)$, $\tau_1 = \text{POC}(\text{Ref}_1) - \text{POC}(\text{current})$. 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., $MVx_0 / MVx_1 = MVy_0 / MVy_1 = -\tau_0 / \tau_1$).

[0069] The motion vector field (v_x, v_y) is determined by minimizing the difference Δ between values in points A and B. FIG. 7 shows an example of intersection of motion trajectory and reference frame planes. Model uses only first linear term of a local Taylor expansion for Δ :

$$[0070] \quad \Delta = (I^{(0)} - I^{(1)}_0 + v_x (\tau_1 \partial I^{(1)} / \partial x + \tau_0 \partial I^{(0)} / \partial x) + v_y (\tau_1 \partial I^{(1)} / \partial y + \tau_0 \partial I^{(0)} / \partial y)) \quad \text{Eq. (13)}$$

[0071] 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 $(2M+1) \times (2M+1)$ square window Ω centered on the currently predicted point (i, j) , where M is equal to 2:

$$[0072] \quad (v_x, v_y) = \underset{v_x, v_y}{\operatorname{argmin}} \sum_{[i', j'] \in \Omega} \Delta^2[i', j'] \quad \text{Eq. (14)}$$

[0073] 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:

$$[0074] \quad v_x = (s_1 + r) > m? \operatorname{clip3} \left(-thBIO, thBIO, -\frac{s_3}{(s_1 + r)} \right) : 0 \quad \text{Eq. (15)}$$

$$[0075] \quad v_y = (s_5 + r) > m? \operatorname{clip3} \left(-thBIO, thBIO, -\frac{s_6 - v_x s_2 / 2}{(s_5 + r)} \right) : 0 \quad \text{Eq. (16)}$$

[0076] where,

$$\begin{aligned}
s_1 &= \sum_{[i',j'] \in \Omega} (\tau_1 \partial I^{(1)} / \partial x + \tau_0 \partial I^{(0)} / \partial x)^2; s_3 = \sum_{[i',j'] \in \Omega} (I^{(1)} - I^{(0)}) (\tau_1 \partial I^{(1)} / \partial x + \tau_0 \partial I^{(0)} / \partial x), \\
[0077] \quad s_2 &= \sum_{[i',j'] \in \Omega} (\tau_1 \partial I^{(1)} / \partial x + \tau_0 \partial I^{(0)} / \partial x) (\tau_1 \partial I^{(1)} / \partial y + \tau_0 \partial I^{(0)} / \partial y), \\
s_5 &= \sum_{[i',j'] \in \Omega} (\tau_1 \partial I^{(1)} / \partial y + \tau_0 \partial I^{(0)} / \partial y)^2; s_6 = \sum_{[i',j'] \in \Omega} (I^{(1)} - I^{(0)}) (\tau_1 \partial I^{(1)} / \partial y + \tau_0 \partial I^{(0)} / \partial y)
\end{aligned} \tag{Eq. (17)}$$

[0078] In order to avoid division by zero or a very small value, regularization parameters r and m can be introduced in Eq. (15) and Eq. (16), where:

$$[0079] \quad r = 500 \cdot 4^{d-8} \tag{Eq. (18)}$$

$$[0080] \quad m = 700 \cdot 4^{d-8} \tag{Eq. (19)}$$

[0081] Here, d is bit depth of the video samples.

[0082] 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. 8A shows an example of access positions outside of a block 800. As shown in FIG. 8A, in Eq. (17), $(2M+1) \times (2M+1)$ 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 801, as shown in FIG. 8B.

[0083] 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 4×4 block. In the block-based BIO, the values of s_n in Eq. (17) of all samples in a 4×4 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:

$$\begin{aligned}
s_{1,b_k} &= \sum_{(x,y) \in b_k} \sum_{[i',j'] \in \Omega(x,y)} (\tau_1 \partial I^{(1)} / \partial x + \tau_0 \partial I^{(0)} / \partial x)^2; s_{3,b_k} = \sum_{(x,y) \in b_k} \sum_{[i',j'] \in \Omega} (I^{(1)} - I^{(0)}) (\tau_1 \partial I^{(1)} / \partial x + \tau_0 \partial I^{(0)} / \partial x), \\
[0084] \quad s_{2,b_k} &= \sum_{(x,y) \in b_k} \sum_{[i',j'] \in \Omega} (\tau_1 \partial I^{(1)} / \partial x + \tau_0 \partial I^{(0)} / \partial x) (\tau_1 \partial I^{(1)} / \partial y + \tau_0 \partial I^{(0)} / \partial y), \\
s_{5,b_k} &= \sum_{(x,y) \in b_k} \sum_{[i',j'] \in \Omega} (\tau_1 \partial I^{(1)} / \partial y + \tau_0 \partial I^{(0)} / \partial y)^2; s_{6,b_k} = \sum_{(x,y) \in b_k} \sum_{[i',j'] \in \Omega} (I^{(1)} - I^{(0)}) (\tau_1 \partial I^{(1)} / \partial y + \tau_0 \partial I^{(0)} / \partial y)
\end{aligned} \tag{Eq. (20)}$$

[0085] Here, b_k denotes the set of samples belonging to the k -th 4×4 block of the predicted block. s_n in Eq (15) and Eq (16) are replaced by $((s_{n,b_k}) \gg 4)$ to derive the associated motion

vector offsets.

[0086] 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 $12 \times 2^{13-d}$.

[0087] 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: Exemplary filters for gradient calculations 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: Exemplary 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}

[0088] 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.

[0089] 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.

6.2 Examples of BIO in VTM-3.0

[0090] Step 1: Judge whether BIO is applicable (W/H are width/height of current block)

[0091] BIO is not applicable if

- [0092] ○ Current video block is affine coded or ATMVP coded
- [0093] ○ $(iPOC - iPOC_0) \times (iPOC - iPOC_1) \geq 0$
- [0094] ○ $H=4$ or ($W=4$ and $H=8$)
- [0095] ○ with Weighted Prediction
- [0096] ○ GBi weights are not (1,1)
- [0097] BIO is not used if total SAD between the two reference blocks (denoted as R_0 and R_1) is smaller than a threshold, wherein

$$SAD = \sum_{(x,y)} |R_0(x,y) - R_1(x,y)|$$

- [0098] Step 2: Data preparation
- [0099] For a $W \times H$ block, $(W+2) \times (H+2)$ samples are interpolated.
- [0100] The inner $W \times H$ samples are interpolated with the 8-tap interpolation filter as in normal motion compensation.
- [0101] The four side outer lines of samples (black circles in FIG. 9) are interpolated with the bi-linear filter.
- [0102] For each position, gradients are calculated on the two reference blocks (R_0 and R_1).
- [0103] $Gx0(x,y) = (R0(x+1,y) - R0(x-1,y)) \gg 4$
- [0104] $Gy0(x,y) = (R0(x,y+1) - R0(x,y-1)) \gg 4$
- [0105] $Gx1(x,y) = (R1(x+1,y) - R1(x-1,y)) \gg 4$
- [0106] $Gy1(x,y) = (R1(x,y+1) - R1(x,y-1)) \gg 4$
- [0107] For each position, internal values are calculated as:
- [0108] $T1 = (R0(x,y) \gg 6) - (R1(x,y) \gg 6)$, $T2 = (Gx0(x,y) + Gx1(x,y)) \gg 3$,
 $T3 = (Gy0(x,y) + Gy1(x,y)) \gg 3$; and
- [0109] $B1(x,y) = T2 * T2$, $B2(x,y) = T2 * T3$, $B3(x,y) = -T1 * T2$, $B5(x,y) = T3 * T3$, $B6(x,y) = -T1 * T3$
- [0110] Step 3: Calculate prediction for each block
- [0111] BIO is skipped for a 4×4 block if SAD between the two 4×4 reference blocks is smaller than a threshold.
- [0112] Calculate Vx and Vy .
- [0113] Calculate the final prediction for each position in the 4×4 block:

[00114] $b(x,y)=(Vx(Gx^0(x,y)-Gx^1(x,y))+Vy(Gy^0(x,y)-Gy^1(x,y))+1)>>1$

[00115] $P(x,y)=(R^0(x,y)+R^1(x,y)+b(x,y)+offset)>>shift$

[00116] Herein, $b(x,y)$ is known as a correction item.

6.3 Alternative examples of BIO in VTM-3.0

[00117] 8.3.4 Decoding process for inter blocks

[00118] -- If predFlagL0 and predFlagL1 are equal to 1, $\text{DiffPicOrderCnt}(\text{currPic}, \text{refPicList0}[\text{refIdx0}]) * \text{DiffPicOrderCnt}(\text{currPic}, \text{refPicList1}[\text{refIdx1}]) < 0$, $\text{MotionModelIdx}[\text{xCb}][\text{yCb}]$ is equal to 0 and $\text{MergeModeList}[\text{merge_idx}[\text{xCb}][\text{yCb}]]$ is not equal to SbCol , set the value of bioAvailableFlag to TRUE.

[00119] -- Otherwise, set the value of bioAvailableFlag to FALSE.

[00120] -- If bioAvailableFlag is equal to TRUE, the following is applied:

[00121] -- The variable shift is set equal to $\text{Max}(2, 14 - \text{bitDepth})$.

[00122] -- The variables $\text{cuLevelAbsDiffThres}$ and $\text{subCuLevelAbsDiffThres}$ are set equal to $(1 << (\text{bitDepth} - 8 + \text{shift})) * \text{cbWidth} * \text{cbHeight}$ and $1 << (\text{bitDepth} - 3 + \text{shift})$. The variable $\text{cuLevelSumAbsoluteDiff}$ is set to 0.

[00123] -- For $\text{xSbIdx}=0..(\text{cbWidth}>>2)-1$ and $\text{ySbIdx}=0..(\text{cbHeight}>>2)-1$, the variable $\text{subCuLevelSumAbsoluteDiff}[\text{xSbIdx}][\text{ySbIdx}]$ and the bidirectional optical flow utilization flag $\text{bioUtilizationFlag}[\text{xSbIdx}][\text{ySbIdx}]$ of the current subblock are derived as:

[00124] $\text{subCuLevelSumAbsoluteDiff}[\text{xSbIdx}][\text{ySbIdx}] = \sum_i \sum_j \text{Abs}(\text{predSamplesL0L}[(\text{xSbIdx}<<2)+1+i][(\text{ySbIdx}<<2)+1+j] - \text{predSamplesL1L}[(\text{xSbIdx}<<2)+1+i][(\text{ySbIdx}<<2)+1+j])$ with $i, j = 0..3$

[00125] $\text{bioUtilizationFlag}[\text{xSbIdx}][\text{ySbIdx}] = \text{subCuLevelSumAbsoluteDiff}[\text{xSbIdx}][\text{ySbIdx}] \geq \text{subCuLevelAbsDiffThres}$

[00126] $\text{cuLevelSumAbsoluteDiff} += \text{subCuLevelSumAbsoluteDiff}[\text{xSbIdx}][\text{ySbIdx}]$

[00127] -- If $\text{cuLevelSumAbsoluteDiff}$ is smaller than $\text{cuLevelAbsDiffThres}$, set bioAvailableFlag to FALSE.

[00128] -- If bioAvailableFlag is equal to TRUE, the prediction samples inside the current luma coding subblock, $\text{predSamplesL}[\text{xL} + \text{xSb}][\text{yL} + \text{ySb}]$ with $\text{xL} = 0.. \text{sbWidth} - 1$ and $\text{yL} = 0.. \text{sbHeight} - 1$, are derived by invoking the bi-directional optical flow sample prediction process specified in clause 8.3.4.5 with the luma coding subblock width sbWidth , the luma

coding subblock height $sbHeight$ and the sample arrays $predSamplesL0L$ and $predSamplesL1L$, and the variables $predFlagL0$, $predFlagL1$, $refIdxL0$, $refIdxL1$.

[00129] 8.3.4.3 Fractional sample interpolation process

[00130] 8.3.4.3.1 General

[00131] Inputs to this process are:

[00132] – a luma location (xSb , ySb) specifying the top-left sample of the current coding sub-block relative to the top left luma sample of the current picture,

[00133] – a variable $sbWidth$ specifying the width of the current coding sub-block in luma samples,

[00134] – a variable $sbHeight$ specifying the height of the current coding sub-block in luma samples,

[00135] – a luma motion vector $mvLX$ given in 1/16-luma-sample units,

[00136] – a chroma motion vector $mvCLX$ given in 1/32-chroma-sample units,

[00137] – the selected reference picture sample array $refPicLXL$ and the arrays $refPicLXCb$ and $refPicLXCr$.

[00138] – the bidirectional optical flow enabling flag $bioAvailableFlag$.

[00139] Outputs of this process are:

[00140] – an $(sbWidth) \times (sbHeight)$ array $predSamplesLXL$ of prediction luma sample values when $bioAvailableFlag$ is FALSE, or an $(sbWidth+2) \times (sbHeight+2)$ array $predSamplesLXL$ of prediction luma sample values when $bioAvailableFlag$ is TRUE.

[00141] – two $(sbWidth / 2) \times (sbHeight / 2)$ arrays $predSamplesLXCb$ and $predSamplesLXCr$ of prediction chroma sample values.

[00142] Let ($xIntL$, $yIntL$) be a luma location given in full-sample units and ($xFracL$, $yFracL$) be an offset given in 1/16-sample units. These variables are used only in this clause for specifying fractional-sample locations inside the reference sample arrays $refPicLXL$, $refPicLXCb$ and $refPicLXCr$.

[00143] When $bioAvailableFlag$ is equal to TRUE, for each luma sample location ($xL = -1..sbWidth$, $yL = -1..sbHeight$) inside the prediction luma sample array $predSamplesLXL$, the corresponding prediction luma sample value $predSamplesLXL[xL][yL]$ is derived as follows:

[00144] – The variables $xIntL$, $yIntL$, $xFracL$ and $yFracL$ are derived as follows:

- [00145] $x_{\text{IntL}} = x_{\text{Sb}} - 1 + (mvLX[0] \gg 4) + x_L$
- [00146] $y_{\text{IntL}} = y_{\text{Sb}} - 1 + (mvLX[1] \gg 4) + y_L$
- [00147] $x_{\text{FracL}} = mvLX[0] \& 15$
- [00148] $y_{\text{FracL}} = mvLX[1] \& 15$
- [00149] – The value of `bilinearFiltEnabledFlag` is derived as follows:
- [00150] – If x_L is equal to -1 or `sbWidth`, or y_L is equal to -1 or `sbHeight`, set the value of `bilinearFiltEnabledFlag` to `TRUE`.
- [00151] – Else, set the value of `bilinearFiltEnabledFlag` to `FALSE`
- [00152] – The prediction luma sample value `predSamplesLXL[xL][yL]` is derived by invoking the process specified in clause 8.3.4.3.2 with $(x_{\text{IntL}}, y_{\text{IntL}})$, $(x_{\text{FracL}}, y_{\text{FracL}})$, `refPicLXL` and `bilinearFiltEnabledFlag` as inputs.
- [00153] When `bioAvailableFlag` is equal to `FALSE`, for each luma sample location $(x_L = 0..sbWidth - 1, y_L = 0..sbHeight - 1)$ inside the prediction luma sample array `predSamplesLXL`, the corresponding prediction luma sample value `predSamplesLXL[xL][yL]` is derived as follows:
- [00154] – The variables x_{IntL} , y_{IntL} , x_{FracL} and y_{FracL} are derived as follows:
- [00155] $x_{\text{IntL}} = x_{\text{Sb}} + (mvLX[0] \gg 4) + x_L$
- [00156] $y_{\text{IntL}} = y_{\text{Sb}} + (mvLX[1] \gg 4) + y_L$
- [00157] $x_{\text{FracL}} = mvLX[0] \& 15$
- [00158] $y_{\text{FracL}} = mvLX[1] \& 15$
- [00159] – The variable `bilinearFiltEnabledFlag` is set to `FALSE`.
- [00160] – The prediction luma sample value `predSamplesLXL[xL][yL]` is derived by invoking the process specified in clause 8.3.4.3.2 with $(x_{\text{IntL}}, y_{\text{IntL}})$, $(x_{\text{FracL}}, y_{\text{FracL}})$, and `refPicLXL` and `bilinearFiltEnabledFlag` as inputs.
- [00161] 8.3.4.5 Bi-directional optical flow prediction process
- [00162] Inputs to this process are:
- [00163] – two variables `nCbW` and `nCbH` specifying the width and the height of the current coding block,
- [00164] – two $(nCbW+2) \times (nCbH+2)$ luma prediction sample arrays `predSamplesL0` and `predSamplesL1`,

- [00165] – the prediction list utilization flags, predFlagL0 and predFlagL1,
- [00166] – the reference indices refIdxL0 and refIdxL1,
- [00167] – the bidirectional optical flow utilization flags bioUtilizationFlag[xSbIdx][ySbIdx] with $xSbIdx = 0..(nCbw >> 2) - 1$, $ySbIdx = 0..(nCbh >> 2) - 1$
- [00168] Output of this process is the $(nCbw) \times (nCbh)$ array pbSamples of luma prediction sample values.
- [00169] The variable bitDepth is set equal to BitDepthY.
- [00170] The variable shift2 is set equal to $\text{Max}(3, 15 - \text{bitDepth})$ and the variable offset2 is set equal to $1 \ll (\text{shift2} - 1)$.
- [00171] The variable mvRefineThres is set equal to $1 \ll (13 - \text{bitDepth})$.
- [00172] For $xSbIdx = 0..(nCbw >> 2) - 1$ and $ySbIdx = 0..(nCbh >> 2) - 1$,
- [00173] – If bioUtilizationFlag[xSbIdx][ySbIdx] is FALSE, for $x = xSb..xSb+3, y = ySb..ySb+3$, the prediction sample values of the current prediction unit are derived as follows:
- [00174]
$$\text{pbSamples}[x][y] = \text{Clip3}(0, (1 \ll \text{bitDepth}) - 1,$$
- [00175]
$$(\text{predSamplesL0}[x][y] + \text{predSamplesL1}[x][y] + \text{offset2}) \gg \text{shift2})$$
- [00176] – Otherwise, the prediction sample values of the current prediction unit are derived as follows:
- [00177] – The location (xSb, ySb) specifying the top-left sample of the current subblock relative to the top left sample of prediction sample arrays predSamplesL0 and predSampleL1 is derived as follows:
- [00178]
$$xSb = (xSbIdx \ll 2) + 1$$
- [00179]
$$ySb = (ySbIdx \ll 2) + 1$$
- [00180] – For $x = xSb-1..xSb+4, y = ySb-1..ySb+4$, the followings are applied:
- [00181] – The locations (hx, vy) for each of the corresponding sample (x, y) inside the prediction sample arrays are derived as follows:
- [00182]
$$hx = \text{Clip3}(1, nCbw, x)$$
- [00183]
$$vy = \text{Clip3}(1, nCbh, y)$$
- [00184] – The variables gradientHL0[x][y], gradientVL0[x][y], gradientHL1[x][y] and gradientVL1[x][y] are derived as follows:

- [00185] $\text{gradientHL0}[x][y] = (\text{predSamplesL0}[hx+1][vy] - \text{predSampleL0}[hx-1][vy]) \gg 4$
- [00186] $\text{gradientVL0}[x][y] = (\text{predSampleL0}[hx][vy+1] - \text{predSampleL0}[hx][vy-1]) \gg 4$
- [00187] $\text{gradientHL1}[x][y] = (\text{predSamplesL1}[hx+1][vy] - \text{predSampleL1}[hx-1][vy]) \gg 4$
- [00188] $\text{gradientVL1}[x][y] = (\text{predSampleL1}[hx][vy+1] - \text{predSampleL1}[hx][vy-1]) \gg 4$
- [00189] – The variables temp, tempX and tempY are derived as follows:
- [00190] $\text{temp}[x][y] = (\text{predSamplesL0}[hx][vy] \gg 6) - (\text{predSamplesL1}[hx][vy] \gg 6)$
- [00191] $\text{tempX}[x][y] = (\text{gradientHL0}[x][y] + \text{gradientHL1}[x][y]) \gg 3$
- [00192] $\text{tempY}[x][y] = (\text{gradientVL0}[x][y] + \text{gradientVL1}[x][y]) \gg 3$
- [00193] – The variables sGx2, sGy2, sGxGy, sGxdI and sGydI are derived as follows:
- [00194] $sGx2 = \sum_x \sum_y (\text{tempX}[xSb+x][ySb+y] * \text{tempX}[xSb+x][ySb+y])$ with $x, y = -1..4$
- [00195] $sGy2 = \sum_x \sum_y (\text{tempY}[xSb+x][ySb+y] * \text{tempY}[xSb+x][ySb+y])$ with $x, y = -1..4$
- [00196] $sGxGy = \sum_x \sum_y (\text{tempX}[xSb+x][ySb+y] * \text{tempY}[xSb+x][ySb+y])$ with $x, y = -1..4$
- [00197] $sGxdI = \sum_x \sum_y (-\text{tempX}[xSb+x][ySb+y] * \text{temp}[xSb+x][ySb+y])$ with $x, y = -1..4$
- [00198] $sGydI = \sum_x \sum_y (-\text{tempY}[xSb+x][ySb+y] * \text{temp}[xSb+x][ySb+y])$ with $x, y = -1..4$
- [00199] – The horizontal and vertical motion refinements of the current sub-block are derived as:
- [00200] $vx = sGx2 > 0 ? \text{Clip3}(-mvRefineThres, mvRefineThres, -$

$(sGxdI \ll 3) \gg \text{Floor}(\text{Log2}(sGx2))) : 0$

[00201] $vy = sGy2 > 0 ? \text{Clip3}(-mvRefineThres, mvRefineThres, ((sGydI \ll 3) - ((vx * sGxGym) \ll 12 + vx * sGxGys) \gg 1) \gg \text{Floor}(\text{Log2}(sGy2))) : 0$

[00202] $sGxGym = sGxGy \gg 12;$

[00203] $sGxGys = sGxGy \& ((1 \ll 12) - 1)$

[00204] For $x=xSb-1..xSb+2, y=ySb-1..ySb+2$, the followings are applied:

[00205] $\text{sampleEnh} = \text{Round}((vx * (\text{gradientHL1}[x+1][y+1] - \text{gradientHL0}[x+1][y+1]) \gg 1) + \text{Round}((vy * (\text{gradientVL1}[x+1][y+1] - \text{gradientVL0}[x+1][y+1]) \gg 1))$

[00206] $\text{pbSamples}[x][y] = \text{Clip3}(0, (1 \ll \text{bitDepth}) - 1, (\text{predSamplesL0}[x+1][y+1] + \text{predSamplesL1}[x+1][y+1] + \text{sampleEnh} + \text{offset2}) \gg \text{shift2})$

7 Prediction refinement with optical flow (PROF)

[00207] This contribution proposes a method to refine the sub-block based affine motion compensated prediction with optical flow. After the sub-block based affine motion compensation is performed, prediction sample is refined by adding a difference derived by the optical flow equation, which is referred as prediction refinement with optical flow (PROF). The proposed method can achieve inter prediction in pixel level granularity without increasing the memory access bandwidth.

[00208] To achieve a finer granularity of motion compensation, this contribution proposes a method to refine the sub-block based affine motion compensated prediction with optical flow. After the sub-block based affine motion compensation is performed, luma prediction sample is refined by adding a difference derived by the optical flow equation. The proposed PROF is described as following four steps.

[00209] Step 1) The sub-block-based affine motion compensation is performed to generate sub-block prediction $I(i, j)$.

[00210] Step2) The spatial gradients $g_x(i, j)$ and $g_y(i, j)$ of the sub-block prediction are calculated at each sample location using a 3-tap filter $[-1, 0, 1]$.

[00211] $g_x(i, j) = I(i + 1, j) - I(i - 1, j)$

[00212] $g_y(i, j) = I(i, j + 1) - I(i, j - 1)$

[00213] The sub-block prediction is extended by one pixel on each side for the gradient calculation. To reduce the memory bandwidth and complexity, the pixels on the extended

borders are copied from the nearest integer pixel position in the reference picture. Therefore, additional interpolation for padding region is avoided.

[00214] Step 3) The luma prediction refinement is calculated by the optical flow equation.

$$[00215] \quad \Delta I(i, j) = g_x(i, j) * \Delta v_x(i, j) + g_y(i, j) * \Delta v_y(i, j)$$

[00216] Herein, $\Delta v(i, j)$ is the difference between pixel MV computed for sample location (i, j) , denoted by $v(i, j)$, and the sub-block MV of the sub-block to which pixel (i, j) belongs, as shown in FIG. 10.

[00217] Since the affine model parameters and the pixel location relative to the sub-block center are not changed from sub-block to sub-block, $\Delta v(i, j)$ can be calculated for the first sub-block, and reused for other sub-blocks in the same CU. Let x and y be the horizontal and vertical offset from the pixel location to the center of the sub-block, $\Delta v(x, y)$ can be derived by the following equation,

$$[00218] \quad \begin{cases} \Delta v_x(x, y) = c * x + d * y \\ \Delta v_y(x, y) = e * x + f * y \end{cases}$$

[00219] For 4-parameter affine model,

$$[00220] \quad \begin{cases} c = f = \frac{v_{1x} - v_{0x}}{w} \\ e = -d = \frac{v_{1y} - v_{0y}}{w} \end{cases}$$

[00221] For 6-parameter affine model,

$$[00222] \quad \begin{cases} c = \frac{v_{1x} - v_{0x}}{w} \\ d = \frac{v_{2x} - v_{0x}}{h} \\ e = \frac{v_{1y} - v_{0y}}{w} \\ f = \frac{v_{2y} - v_{0y}}{h} \end{cases}$$

[00223] Herein, (v_{0x}, v_{0y}) , (v_{1x}, v_{1y}) , (v_{2x}, v_{2y}) are the top-left, top-right and bottom-left control point motion vectors, w and h are the width and height of the CU.

[00224] Step 4) Finally, the luma prediction refinement is added to the sub-block prediction $I(i, j)$. The final prediction I' is generated as the following equation:

$$[00225] \quad I'(i, j) = I(i, j) + \Delta I(i, j)$$

8 Drawbacks of existing implementations

[00226] The design of VVC has the following problems:

[00227] (1) Different gradient calculation methods, such as [1, -1], [-1, 2, -1], are utilized in BIO (a.k.a., BDOF) /ALF/PROF.

[00228] (2) In the BIO process, the gradient calculation is done in sample level wherein for each sample, the gradient is calculated. While the refined motion vectors (such as V_x and V_y) are done in 4x4 sub-block level which depends on gradient values of a 6x6 block covering the sub-block. The per-sample calculation of gradient increases the computational complexity.

9 Exemplary methods for interactions of ALF with other coding tools

[00229] Embodiments of the presently disclosed technology overcome the drawbacks of existing implementations, thereby providing video coding with higher coding efficiencies. The interactions of adaptive loop filtering with other coding tools, based on the disclosed technology, may enhance both existing and future video coding standards, is elucidated in the following examples described for various implementations. The examples of the disclosed technology provided below explain general concepts, and are not meant to be interpreted as limiting. In an example, unless explicitly indicated to the contrary, the various features described in these examples may be combined.

[00230] In the following examples, $\text{Shift}(x, s)$ is defined as $\text{Shift}(x, s) = (x + \text{off}) \gg s$.

[00231] In the following examples, $\text{SatShift}(x, n)$ is defined as

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

[00232] In an example, offset0 and/or offset1 are set to $(1 \ll n) \gg 1$ or $(1 \ll (n-1))$. In another example, offset0 and/or offset1 are set to 0. In another example, $\text{offset0} = \text{offset1} = ((1 \ll n) \gg 1) - 1$ or $((1 \ll (n-1))) - 1$.

[00233] In the following examples, $\text{Clip3}(x, \text{Min}, \text{Max})$ is defined as

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

[00234] It is proposed to align the gradient calculation process in ALF/Non-local ALF to that used in the decoder derived methods, such as decoder motion vector refinement/BIO etc.

1. It is proposed that the whole or partial of the derivation process of the gradient values used in BIO is aligned with the corresponding derivation process of the gradient values used in ALF.
 - a. In one example, the vertical gradient values in BIO (denoted as g_v) are calculated in the same way to calculate the vertical gradient values in ALF.
 - i. In one example, the vertical gradient calculation is defined as $[-1, 2, -1]$ filter.
 - ii. In one example,

$$g_v = \text{Shift}(2R(k, l) - R(k, l - \text{off1}) - R(k, l + \text{off2}), \text{prec})$$
 wherein $R(i, j)$ indicates a reconstructed or prediction sample at coordinate (i, j) , variables off1 , off2 and prec are integers such as $\text{off1}=\text{off2}=1$ and $\text{prec} = 0$.
 - iii. Alternatively,

$$g_v = \text{SatShift}(2R(k, l) - R(k, l - \text{off1}) - R(k, l + \text{off2}), \text{prec})$$
 wherein $R(i, j)$ indicates a reconstructed or prediction sample at coordinate (i, j) , variables off1 , off2 and prec are integers such as $\text{off1}=\text{off2}=1$ and $\text{prec} = 0$.
 - b. In one example, the horizontal gradient values in BIO (denoted as g_h) are calculated in the same way as that utilized to calculate the horizontal gradient values in ALF.
 - i. In one example, the horizontal gradient calculation is defined as $[-1, 2, -1]$ filter.
 - ii. In one example,

$$g_h = \text{Shift}(2R(k, l) - R(k - \text{off1}, l) - R(k + \text{off2}, l), \text{prec})$$
 wherein $R(i, j)$ indicates a reconstructed or prediction sample at coordinate (i, j) , variables off1 , off2 and prec are integers such as $\text{off1}=\text{off2}=1$ and $\text{prec} = 0$;
 - iii. Alternatively,

$$g_h = \text{SatShift}(2R(k, l) - R(k - \text{off1}, l) - R(k + \text{off2}, l), \text{prec})$$

wherein $R(i, j)$ indicates a reconstructed or prediction sample at coordinate (i, j) , variables $off1$, $off2$ and $prec$ are integers such as $off1=off2=1$ and $prec = 0$.

2. It is proposed that the whole or partial of the derivation process of the gradient values used in PROF is the same as the corresponding derivation process of the gradient values used in ALF.
 - a. In one example, the vertical gradient values in PROF (denoted as g_v) are calculated in the same way to calculate the vertical gradient values in ALF.
 - i. In one example, the vertical gradient calculation is defined as $[-1, 2, -1]$ filter.
 - ii. In one example, $g_v = Shift(2R(k, l) - R(k, l - off1) - R(k, l + off2), prec)$ wherein $R(i, j)$ indicates a reconstructed or prediction sample at coordinate (i, j) . $off1$, $off2$ and $prec$ are integers such as $off1=off2=1$ and $prec = 0$;
 - iii. Alternatively,

$$g_v = SatShift(2R(k, l) - R(k, l - off1) - R(k, l + off2), prec)$$
 - b. In one example, the horizontal gradient values in PROF (denoted as g_h) are calculated in the same way to calculate the horizontal gradient values in ALF.
 - i. In one example, the horizontal gradient calculation is defined as $[-1, 2, -1]$ filter.
 - ii. In one example, $g_h = Shift(2R(k, l) - R(k - off1, l) - R(k + off2, l), prec)$ wherein $R(i, j)$ indicates a reconstructed or prediction sample at coordinate (i, j) . $off1$, $off2$ and $prec$ are integers such as $off1=off2=1$ and $prec = 0$;
 - iii. Alternatively,

$$g_h = SatShift(2R(k, l) - R(k - off1, l) - R(k + off2, l), prec)$$
3. It is proposed that a sub-block level gradient calculation method may be applied to BIO/PROF and other non-ALF coding tools wherein the gradient is not calculated for all samples within one block.

- a. In one example, only selected coordinates may be utilized to derive gradient values, such as depicted in FIG. 4.
 - b. In one example, when gradient values are not calculated for certain coordinates, the associated gradient values may be copied from that associated with its neighbors wherein gradient is calculated.
 - c. How to copy gradient values from selected samples to those remaining samples may depend on the gradient direction, such as horizontal gradient or vertical gradient.
4. It is proposed that the whole or partial of the derivation process of the gradient values used in ALF is the same as the corresponding derivation process of other coding tools, such as the gradient values used in BIO.
- a. In one example, the vertical gradient values in ALF (denoted as g_v) are calculated in the same way to calculate the vertical gradient values in BIO.
 - i. For example,

$$g_v(x,y) = \text{Shift}(R(x, y+off1) - R(x, y-off2), prec),$$
 where $R(x, y)$ indicates a reconstructed or prediction sample at coordinate (x,y) , variables $off1$, $off2$ and $prec$ are integers such as $off1=off2=1$ and $prec = 0$;
 - ii. Alternatively,

$$g_v(x,y) = \text{SatShift}(R(x, y+off1) - R(x, y-off2), prec).$$
 - iii. Alternatively,

$$g_v(x,y) = \text{Shift}(|R(x, y+off1) - R(x, y-off2)|, prec).$$
 - iv. Alternatively,

$$g_v(x,y) = \text{SatShift}(|R(x, y+off1) - R(x, y-off2)|, prec).$$
 - b. In one example, the horizontal gradient values in ALF (denoted as g_h) are calculated in the same way to calculate the horizontal gradient values in BIO.
 - i. For example,

$$g_h(x,y) = \text{Shift}(R(x+1,y) - R(x-1,y), prec),$$
 where $R(x, y)$ indicates a reconstructed or prediction sample at coordinate (x,y) . $off1$, $off2$ and $prec$ are integers such as $off1=off2=1$ and $prec = 0$;

ii. Alternatively,

$$g_h(x,y)=\text{SatShift}(R(x+1,y)-R(x-1,y), \text{prec}).$$

iii. Alternatively,

$$g_h(x,y)=\text{Shift}(|R(x+1,y)-R(x-1,y)|, \text{prec})..$$

iv. Alternatively,

$$g_h(x,y)=\text{SatShift}(|R(x+1,y)-R(x-1,y)|, \text{prec}).$$

- c. In one example, the vertical gradient values for all or some pixels in a block are calculated in the same way to calculate the vertical gradient values in BIO and averaged (or being processed in other ways) to get the vertical gradient value for a block used in ALF.
 - d. In one example, the horizontal gradient values for all or some pixels in a block are calculated in the same way to calculate the horizontal gradient values in BIO and averaged (or being processed in other ways) to get the horizontal gradient value for a block used in ALF.
5. It is proposed that the whole or partial of the padding method used to pad the samples out of range used to derive the gradients in ALF, is the same as the corresponding padding method used to pad the samples out of range used to derive the gradients in BIO.
- a. Alternatively, the whole or partial of the padding method used to pad the samples out of range used to derive the gradients in BIO, is the same as the corresponding padding method used to pad the samples out of range used to derive the gradients in ALF.
6. It is proposed that the whole or partial of the padding method used to pad the samples out of range used to derive the gradients in PROF, is the same as the corresponding padding method used to pad the samples out of range used to derive the gradients in BIO.
- a. Alternatively, the whole or partial of the padding method used to pad the samples out of range used to derive the gradients in BIO, is the same as the corresponding padding method used to pad the samples out of range used to derive the gradients in PROF.

7. It is proposed that the whole or partial of the padding method used to pad the samples out of range used to derive the gradients in PROF, is the same as the corresponding padding method used to pad the samples out of range used to derive the gradients in ALF.
 - a. Alternatively, the whole or partial of the padding method used to pad the samples out of range used to derive the gradients in ALF, is the same as the corresponding padding method used to pad the samples out of range used to derive the gradients in PROF.
8. The proposed methods may be also applicable to other coding tools that rely on the calculation of gradients.

[00235] The examples described above may be incorporated in the context of the method described below, e.g., methods 1100 and 1150, which may be implemented at a video decoder or a video encoder.

[00236] FIG. 11A shows a flowchart of an exemplary method for video processing. The method 1100 includes, at step 1102, configuring, for a current video block, a first derivation process for deriving a first gradient value used in a first coding tool based on a second derivation process for deriving a second gradient value used in a second coding tool that is different from the first coding tool.

[00237] The method 1100 includes, at step 1104, reconstructing, based on the first derivation process and the first coding tool, the current video block from a corresponding bitstream representation. In some embodiments, at least one of the first coding tool and the second coding tool relates to a pixel filtering process, and the first gradient value and the second gradient value are indicative of a directional change in an intensity or a color component over a subset of samples in the current video block.

[00238] In some embodiments, the pixel filtering process is an adaptive loop filtering process.

[00239] In some embodiments, the first coding tool is a bi-directional optical flow (BIO) refinement and the second coding tool is an adaptive loop filtering (ALF) process.

[00240] In some embodiments, the first coding tool is a prediction refinement with optical flow (PROF) process and the second coding tool is an adaptive loop filtering (ALF) process.

[00241] In some embodiments, the first coding tool is an adaptive loop filtering (ALF) process

and the second coding tool is a bi-directional optical flow (BIO) refinement.

[00242] In some embodiments, the first and second derivation processes comprise a vertical gradient value calculation or a horizontal gradient value calculation. In an example, the vertical gradient value calculation or the horizontal gradient value calculation is based on a $[-1, 2, -1]$ filter.

[00243] In some embodiments, the first derivation process comprises a sub-block level gradient value calculation. In an example, the first derivation process is not applied to each sample of the current video block.

[00244] FIG. 11B shows a flowchart of an exemplary method for video processing. The method 1150 includes, at step 1152, configuring, for a current video block, a first padding process used in a first coding tool based on a second padding process used in a second coding tool that is different from the first coding tool.

[00245] The method 1150 includes, at step 1154, reconstructing, based on the first padding process and the first coding tool, the current video block from a corresponding bitstream representation. In some embodiments, the first padding process and the second padding process comprise adding out of range samples in a calculation of a gradient value that is indicative of a directional change in an intensity or a color component over a subset of samples in the current video block.

[00246] In some embodiments, the first coding tool is a bi-directional optical flow (BIO) refinement and the second coding tool is an adaptive loop filtering (ALF) process.

[00247] In some embodiments, the first coding tool is an adaptive loop filtering (ALF) process and the second coding tool is a bi-directional optical flow (BIO) refinement.

[00248] In some embodiments, the first coding tool is a bi-directional optical flow (BIO) refinement and the second coding tool is a prediction refinement with optical flow (PROF) process.

[00249] In some embodiments, the first coding tool is a prediction refinement with optical flow (PROF) process and the second coding tool is a bi-directional optical flow (BIO) refinement.

[00250] In some embodiments, the first coding tool is a prediction refinement with optical flow (PROF) process and the second coding tool is an adaptive loop filtering (ALF) process.

[00251] In some embodiments, the first coding tool is an adaptive loop filtering (ALF) process and the second coding tool is a prediction refinement with optical flow (PROF) process.

10 Example implementations of the disclosed technology

[00252] FIG. 12 is a block diagram of a video processing apparatus 1200. The apparatus 1200 may be used to implement one or more of the methods described herein. The apparatus 1200 may be embodied in a smartphone, tablet, computer, Internet of Things (IoT) receiver, and so on. The apparatus 1200 may include one or more processors 1202, one or more memories 1204 and video processing hardware 1206. The processor(s) 1202 may be configured to implement one or more methods (including, but not limited to, methods 1100 and 1150) described in the present document. The memory (memories) 1204 may be used for storing data and code used for implementing the methods and techniques described herein. The video processing hardware 1206 may be used to implement, in hardware circuitry, some techniques described in the present document.

[00253] In some embodiments, the video coding methods may be implemented using an apparatus that is implemented on a hardware platform as described with respect to FIG. 12.

[00254] FIG. 13 is a flowchart for an example method 1300 of video processing. The method 1300 includes configuring (1302), for a conversion between a first block of video and a bitstream representation of the first block, a first derivation process for deriving a first gradient value used in a first coding tool to be aligned with a second derivation process for deriving a second gradient value used in a second coding tool that is different from the first coding tool; and performing (1304) the conversion based on the configured first derivation process.

[00255] In some embodiments, whole or partial of the first derivation process is aligned with the corresponding whole or partial of the second derivation process.

[00256] In some embodiments, the first derivation processes and the second derivation processes comprise the same vertical gradient value calculation for calculating vertical gradient values and/or the same horizontal gradient value calculation for calculating horizontal gradient values.

[00257] In some embodiments, the first coding tool is a bi-directional optical flow (BDOF) refinement, and the second coding tool is an adaptive loop filtering (ALF) process.

[00258] In some embodiments, the first coding tool is an ALF process refinement, and the

second coding tool is a BDOF refinement.

[00259] In some embodiments, the vertical gradient value calculation and/or the horizontal gradient value calculation are based on a $[-1, 2, -1]$ filter.

[00260] In some embodiments, the vertical gradient values in BDOF (g_v) and/or the horizontal gradient values in BDOF (g_h) are calculated by using a function of $Shift(x, n)$, the $Shift(x, n)$ is defined as

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

where, x is a variable, $offset0$ is set to $(1 \ll n) \gg 1$ or $(1 \ll (n-1))$, or $offset0$ is set to 0, or $offset0 = ((1 \ll n) \gg 1) - 1$ or $((1 \ll (n-1))) - 1$, n is an integer.

[00261] In some embodiments, the vertical gradient values in BDOF (g_v) are calculated as following:

$$g_v = Shift(2R(k, l) - R(k, l - off1) - R(k, l + off2), prec),$$

wherein $R(i, j)$ indicates a reconstructed or prediction sample at coordinate (i, j) , variables $off1$, $off2$ and $prec$ are integers.

[00262] In some embodiments, the horizontal gradient values in BDOF (g_h) are calculated as following:

$$g_h = Shift(2R(k, l) - R(k - off1, l) - R(k + off2, l), prec),$$

wherein $R(i, j)$ indicates a reconstructed or prediction sample at coordinate (i, j) , variables $off1$, $off2$ and $prec$ are integers.

[00263] In some embodiments, $off1 = off2 = 1$ and $prec = 0$.

[00264] In some embodiments, the vertical gradient values in BDOF (g_v) and/or the horizontal gradient values in BDOF (g_h) are calculated by using a function of $SatShift(x, n)$, the $SatShift(x, n)$ is defined as

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

where, x is a variable, $offset0$ and/or $offset1$ are set to $(1 \ll n) \gg 1$ or $(1 \ll (n-1))$, or $offset0$ and/or $offset1$ are set to 0, or $offset0 = offset1 = ((1 \ll n) \gg 1) - 1$ or $((1 \ll (n-1))) - 1$, n is an integer.

[00265] In some embodiments, the vertical gradient values in BDOF (g_v) are calculated as following:

$$g_v = SatShift(2R(k, l) - R(k, l - off1) - R(k, l + off2), prec),$$

wherein $R(i, j)$ indicates a reconstructed or prediction sample at coordinate (i, j) , variables $off1$, $off2$ and $prec$ are integers.

[00266] In some embodiments, the horizontal gradient values in BDOF (g_h) are calculated as following:

$$g_h = \text{SatShift}(2R(k, l) - R(k - off1, l) - R(k + off2, l), prec),$$

wherein $R(i, j)$ indicates a reconstructed or prediction sample at coordinate (i, j) , variables $off1$, $off2$ and $prec$ are integers.

[00267] In some embodiments, $off1=off2=1$ and $prec = 0$.

[00268] In some embodiments, the first coding tool is a prediction refinement with optical flow (PROF) process which is applied to affine coded blocks, and the second coding tool is an adaptive loop filtering (ALF) process.

[00269] In some embodiments, the vertical gradient value calculation and/or the horizontal gradient value calculation are based on a $[-1, 2, -1]$ filter.

[00270] In some embodiments, the vertical gradient values in PROF (g_v) and/or the horizontal gradient values in PROF (g_h) are calculated by using a function of $\text{Shift}(x, n)$, the $\text{Shift}(x, n)$ is defined as

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

where, x is a variable, offset0 is set to $(1 \ll n) \gg 1$ or $(1 \ll (n-1))$, or offset0 is set to 0, or $\text{offset0} = ((1 \ll n) \gg 1) - 1$ or $((1 \ll (n-1))) - 1$, n is an integer.

[00271] In some embodiments, the vertical gradient values in PROF (g_v) are calculated as following:

$$g_v = \text{Shift}(2R(k, l) - R(k, l - off1) - R(k, l + off2), prec),$$

wherein $R(i, j)$ indicates a reconstructed or prediction sample at coordinate (i, j) , variables $off1$, $off2$ and $prec$ are integers.

[00272] In some embodiments, the horizontal gradient values in PROF (g_h) are calculated as following:

$$g_h = \text{Shift}(2R(k, l) - R(k - off1, l) - R(k + off2, l), prec),$$

wherein $R(i, j)$ indicates a reconstructed or prediction sample at coordinate (i, j) , variables $off1$, $off2$ and $prec$ are integers.

[00273] In some embodiments, $off1=off2=1$ and $prec = 0$.

[00274] In some embodiments, the vertical gradient values in PROF (g_v) and/or the horizontal gradient values in PROF (g_h) are calculated by using a function of $SatShift(x, n)$, the $SatShift(x, n)$ is defined as

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

where, x is a variable, $offset0$ and/or $offset1$ are set to $(1 << n) >> 1$ or $(1 << (n-1))$, or $offset0$ and/or $offset1$ are set to 0, or $offset0=offset1 = ((1 << n) >> 1) - 1$ or $((1 << (n-1))) - 1$, n is an integer.

[00275] In some embodiments, the vertical gradient values in PROF (g_v) are calculated as following:

$$g_v = SatShift(2R(k, l) - R(k, l - off1) - R(k, l + off2), prec),$$

wherein $R(i, j)$ indicates a reconstructed or prediction sample at coordinate (i, j) , variables $off1$, $off2$ and $prec$ are integers.

[00276] In some embodiments, the horizontal gradient values in PROF (g_h) are calculated as following:

$$g_h = SatShift(2R(k, l) - R(k - off1, l) - R(k + off2, l), prec),$$

wherein $R(i, j)$ indicates a reconstructed or prediction sample at coordinate (i, j) , variables $off1$, $off2$ and $prec$ are integers.

[00277] In some embodiments, $off1=off2=1$ and $prec = 0$.

[00278] In some embodiments, the first coding tool is an adaptive loop filtering (ALF) process, and the second coding tool is a bi-directional optical flow (BDOF) refinement.

[00279] In some embodiments, the vertical gradient value calculation and/or the horizontal gradient value calculation are based on a $[1, -1]$ filter.

[00280] In some embodiments, the vertical gradient values in ALF (g_v) and/or the horizontal gradient values in ALF (g_h) are calculated by using a function of $Shift(x, n)$, the $Shift(x, n)$ is defined as

$$Shift(x, n) = (x + offset0) >> n,$$

where, x is a variable, $offset0$ is set to $(1 << n) >> 1$ or $(1 << (n-1))$, or $offset0$ is set to 0, or $offset0 = ((1 << n) >> 1) - 1$ or $((1 << (n-1))) - 1$, n is an integer.

[00281] In some embodiments, the vertical gradient values in ALF (g_v) are calculated as following:

$$g_v(x,y)=\text{Shift}(R(x, y+off1)-R(x, y-off2), prec),$$

where, x and y are variables, $R(i, j)$ indicates a reconstructed or prediction sample at coordinate (i, j) , variables $off1$, $off2$ and $prec$ are integers.

[00282] In some embodiments, the vertical gradient values in ALF (g_v) are calculated as following:

$$g_v(x,y)=\text{Shift}(|R(x, y+off1)-R(x, y-off2)|, prec),$$

where, x and y are variables, $R(i, j)$ indicates a reconstructed or prediction sample at coordinate (i, j) , variables $off1$, $off2$ and $prec$ are integers.

[00283] In some embodiments, the horizontal gradient values in ALF (g_h) are calculated as following:

$$g_h(x,y)=\text{Shift}(R(x+1,y)-R(x-1,y), prec),$$

where, x and y are variables, $R(i, j)$ indicates a reconstructed or prediction sample at coordinate (i, j) , variable $prec$ is an integer.

[00284] In some embodiments, the horizontal gradient values in ALF (g_h) are calculated as following:

$$g_h(x,y)=\text{Shift}(|R(x+1,y)-R(x-1,y)|, prec),$$

where, x and y are variables, $R(i, j)$ indicates a reconstructed or prediction sample at coordinate (i, j) , variable $prec$ is an integer.

[00285] In some embodiments, $off1=off2=1$ and $prec = 0$.

[00286] In some embodiments, the vertical gradient values in ALF (g_v) and/or the horizontal gradient values in ALF (g_h) are calculated by using a function of $SatShift(x, n)$, the $SatShift(x, n)$ is defined as

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

where, x is a variable, $offset0$ and/or $offset1$ are set to $(1 << n) >> 1$ or $(1 << (n-1))$, or $offset0$ and/or $offset1$ are set to 0, or $offset0=offset1 = ((1 << n) >> 1) - 1$ or $((1 << (n-1))) - 1$, n is an integer.

[00287] In some embodiments, the vertical gradient values in ALF (g_v) are calculated as following:

$$g_v(x,y)=\text{SatShift}(R(x, y+off1)-R(x, y-off2), prec),$$

where, x and y are variables, $R(i, j)$ indicates a reconstructed or prediction sample at coordinate

(i, j) , variables $off1$, $off2$ and $prec$ are integers.

[00288] In some embodiments, the vertical gradient values in ALF (g_v) are calculated as following:

$$g_v(x, y) = \text{SatShift}(|R(x, y + off1) - R(x, y - off2)|, prec),$$

where, x and y are variables, $R(i, j)$ indicates a reconstructed or prediction sample at coordinate (i, j) , variables $off1$, $off2$ and $prec$ are integers.

[00289] In some embodiments, the horizontal gradient values in ALF (g_h) are calculated as following:

$$g_h(x, y) = \text{SatShift}(R(x + 1, y) - R(x - 1, y), prec),$$

where, x and y are variables, $R(i, j)$ indicates a reconstructed or prediction sample at coordinate (i, j) , variable $prec$ is an integer.

[00290] In some embodiments, the horizontal gradient values in ALF (g_h) are calculated as following:

$$g_h(x, y) = \text{SatShift}(|R(x + 1, y) - R(x - 1, y)|, prec),$$

where, x and y are variables, $R(i, j)$ indicates a reconstructed or prediction sample at coordinate (i, j) , variable $prec$ is an integer.

[00291] In some embodiments, $off1 = off2 = 1$ and $prec = 0$.

[00292] In some embodiments, the vertical gradient values for all or partial samples in the first block are calculated in the same way to calculate the vertical gradient values in BDOF and averaged to obtain the vertical gradient value for the first block used in ALF.

[00293] In some embodiments, the horizontal gradient values for all or partial samples in the first block are calculated in the same way to calculate the horizontal gradient values in BDOF and averaged to obtain the horizontal gradient value for the first block used in ALF.

[00294] FIG. 14 is a flowchart for an example method 1400 of video processing. The method 1400 includes deriving (1402), for a conversion between a first block of video and a bitstream representation of the first block, gradient values used in one or more coding tools by applying a sub-block level gradient calculation process, wherein the gradient values are derived for partial samples within prediction blocks of the first block; and performing (1404) the conversion based on the derived gradient values.

[00295] In some embodiments, the one or more coding tools include at least one of a bi-

directional optical flow (BDOF) refinement, a prediction refinement with optical flow (PROF) process and other non-ALF coding tools.

[00296] In some embodiments, only samples at selected coordinates are used to derive the gradient values.

[00297] In some embodiments, when gradient values are not calculated for samples at certain coordinates, the gradient values associated the samples at certain coordinates are copied from that associated with its neighbors wherein gradient vale is calculated.

[00298] In some embodiments, how to copy gradient values from selected samples to those remaining samples depends on gradient direction including at least one of horizontal gradient or vertical gradient.

[00299] FIG. 15 is a flowchart for an example method 1500 of video processing. The method 1500 includes configuring (1502), for a conversion between a first block of video and a bitstream representation of the first block, a first padding process in a first coding tool to be aligned with a second padding process in a second coding tool that is different from the first coding tool, wherein the first padding process is for padding samples out of range used to derive gradient values used in the first coding tool, and the second padding process is for padding samples out of range used to derive gradient values used in the second coding tool; and performing (1504) the conversion based on the configured first padding process.

[00300] In some embodiments, whole or partial of the first padding process is aligned with the corresponding whole or partial of the second padding process.

[00301] In some embodiments, the first coding tool is a bi-directional optical flow (BDOF) refinement, and the second coding tool is an adaptive loop filtering (ALF) process.

[00302] In some embodiments, the first coding tool is an adaptive loop filtering (ALF) process, and the second coding tool is a bi-directional optical flow (BDOF) refinement.

[00303] In some embodiments, the first coding tool is a bi-directional optical flow (BDOF) refinement, and the second coding tool is a prediction refinement with optical flow (PROF) process.

[00304] In some embodiments, the first coding tool is a prediction refinement with optical flow (PROF) process, and the second coding tool is a bi-directional optical flow (BDOF) refinement.

[00305] In some embodiments, the first coding tool is a prediction refinement with optical flow (PROF) process, and the second coding tool is an adaptive loop filtering (ALF) process.

[00306] In some embodiments, the first coding tool is an adaptive loop filtering (ALF) process, and the second coding tool is a prediction refinement with optical flow (PROF) process.

[00307] In some embodiments, the first coding tool and/or the second coding tool include coding tools that rely on calculation of gradient values.

[00308] In some embodiments, the conversion generates the first block of video from the bitstream representation.

[00309] In some embodiments, the conversion generates the bitstream representation from the first block of video.

[00310] 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.

[00311] 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.

[00312] 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.

[00313] 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).

[00314] 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.

[00315] 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.

[00316] 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.

[00317] 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.

[00318] Only a few implementations and examples are described and other implementations, enhancements and variations can be made based on what is described and illustrated in this patent document.

CLAIMS

What is claimed is:

1. A method for processing video, comprising:
configuring, for a conversion between a first block of video and a bitstream representation of the first block, a first derivation process for deriving a first gradient value used in a first coding tool to be aligned with a second derivation process for deriving a second gradient value used in a second coding tool that is different from the first coding tool; and
performing the conversion based on the configured first derivation process.
2. The method of claim 1, wherein whole or partial of the first derivation process is aligned with the corresponding whole or partial of the second derivation process.
3. The method of claim 1 or 2, wherein the first derivation processes and the second derivation processes comprise the same vertical gradient value calculation for calculating vertical gradient values and/or the same horizontal gradient value calculation for calculating horizontal gradient values.
4. The method of claim 3, wherein the first coding tool is a bi-directional optical flow (BDOF) refinement, and the second coding tool is an adaptive loop filtering (ALF) process.
5. The method of claim 3, wherein the first coding tool is an ALF process refinement, and the second coding tool is a BDOF refinement.
6. The method of claim 4 or 5, wherein the vertical gradient value calculation and/or the horizontal gradient value calculation are based on a $[-1, 2, -1]$ filter.
7. The method of claim 6, wherein the vertical gradient values in BDOF (g_v) and/or the horizontal gradient values in BDOF (g_h) are calculated by using a function of $Shift(x, n)$, the $Shift(x, n)$ is defined as

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

where, x is a variable, offset0 is set to $(1 \ll n) \gg 1$ or $(1 \ll (n-1))$, or offset0 is set to 0, or $\text{offset0} = ((1 \ll n) \gg 1) - 1$ or $((1 \ll (n-1))) - 1$, n is an integer.

8. The method of claim 7, wherein the vertical gradient values in BDOF (g_v) are calculated as following:

$$g_v = \text{Shift}(2R(k, l) - R(k, l - \text{off1}) - R(k, l + \text{off2}), \text{prec}),$$

wherein $R(i, j)$ indicates a reconstructed or prediction sample at coordinate (i, j) , variables off1 , off2 and prec are integers.

9. The method of claim 7, wherein the horizontal gradient values in BDOF (g_h) are calculated as following:

$$g_h = \text{Shift}(2R(k, l) - R(k - \text{off1}, l) - R(k + \text{off2}, l), \text{prec}),$$

wherein $R(i, j)$ indicates a reconstructed or prediction sample at coordinate (i, j) , variables off1 , off2 and prec are integers.

10. The method of claim 8 or 9, wherein $\text{off1} = \text{off2} = 1$ and $\text{prec} = 0$.

11. The method of claim 6, wherein the vertical gradient values in BDOF (g_v) and/or the horizontal gradient values in BDOF (g_h) are calculated by using a function of $\text{SatShift}(x, n)$, the $\text{SatShift}(x, n)$ is defined as

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

where, x is a variable, offset0 and/or offset1 are set to $(1 \ll n) \gg 1$ or $(1 \ll (n-1))$, or offset0 and/or offset1 are set to 0, or $\text{offset0} = \text{offset1} = ((1 \ll n) \gg 1) - 1$ or $((1 \ll (n-1))) - 1$, n is an integer.

12. The method of claim 11, wherein the vertical gradient values in BDOF (g_v) are calculated as following:

$$g_v = \text{SatShift}(2R(k, l) - R(k, l - \text{off1}) - R(k, l + \text{off2}), \text{prec}),$$

wherein $R(i, j)$ indicates a reconstructed or prediction sample at coordinate (i, j) , variables $off1$, $off2$ and $prec$ are integers.

13. The method of claim 11, wherein the horizontal gradient values in BDOF (g_h) are calculated as following:

$$g_h = \text{SatShift}(2R(k, l) - R(k - off1, l) - R(k + off2, l), prec),$$

wherein $R(i, j)$ indicates a reconstructed or prediction sample at coordinate (i, j) , variables $off1$, $off2$ and $prec$ are integers.

14. The method of claim 12 or 13, wherein $off1=off2=1$ and $prec = 0$.

15. The method of claim 14, wherein the first coding tool is a prediction refinement with optical flow (PROF) process which is applied to affine coded blocks, and the second coding tool is an adaptive loop filtering (ALF) process.

16. The method of claim 15, wherein the vertical gradient value calculation and/or the horizontal gradient value calculation are based on a $[-1, 2, -1]$ filter.

17. The method of claim 16, wherein the vertical gradient values in PROF (g_v) and/or the horizontal gradient values in PROF (g_h) are calculated by using a function of $Shift(x, n)$, the $Shift(x, n)$ is defined as

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

where, x is a variable, $offset0$ is set to $(1 \ll n) \gg 1$ or $(1 \ll (n-1))$, or $offset0$ is set to 0, or $offset0 = ((1 \ll n) \gg 1) - 1$ or $((1 \ll (n-1))) - 1$, n is an integer.

18. The method of claim 17, wherein the vertical gradient values in PROF (g_v) are calculated as following:

$$g_v = Shift(2R(k, l) - R(k, l - off1) - R(k, l + off2), prec),$$

wherein $R(i, j)$ indicates a reconstructed or prediction sample at coordinate (i, j) , variables $off1$, $off2$ and $prec$ are integers.

19. The method of claim 17, wherein the horizontal gradient values in PROF (g_h) are calculated as following:

$$g_h = \text{Shift}(2R(k, l) - R(k - \text{off1}, l) - R(k + \text{off2}, l), \text{prec}),$$

wherein $R(i, j)$ indicates a reconstructed or prediction sample at coordinate (i, j) , variables off1 , off2 and prec are integers.

20. The method of claim 18 or 19, wherein $\text{off1}=\text{off2}=1$ and $\text{prec} = 0$.

21. The method of claim 16, wherein the vertical gradient values in PROF (g_v) and/or the horizontal gradient values in PROF (g_h) are calculated by using a function of $\text{SatShift}(x, n)$, the $\text{SatShift}(x, n)$ is defined as

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

where, x is a variable, offset0 and/or offset1 are set to $(1 \ll n) \gg 1$ or $(1 \ll (n-1))$, or offset0 and/or offset1 are set to 0, or $\text{offset0}=\text{offset1} = ((1 \ll n) \gg 1) - 1$ or $((1 \ll (n-1))) - 1$, n is an integer.

22. The method of claim 21, wherein the vertical gradient values in PROF (g_v) are calculated as following:

$$g_v = \text{SatShift}(2R(k, l) - R(k, l - \text{off1}) - R(k, l + \text{off2}), \text{prec}),$$

wherein $R(i, j)$ indicates a reconstructed or prediction sample at coordinate (i, j) , variables off1 , off2 and prec are integers.

23. The method of claim 21, wherein the horizontal gradient values in PROF (g_h) are calculated as following:

$$g_h = \text{SatShift}(2R(k, l) - R(k - \text{off1}, l) - R(k + \text{off2}, l), \text{prec}),$$

wherein $R(i, j)$ indicates a reconstructed or prediction sample at coordinate (i, j) , variables off1 , off2 and prec are integers.

24. The method of claim 22 or 23, wherein $\text{off1}=\text{off2}=1$ and $\text{prec} = 0$.

25. The method of claim 3, wherein the first coding tool is an adaptive loop filtering (ALF) process, and the second coding tool is a bi-directional optical flow (BDOF) refinement.

26. The method of claim 25, wherein the vertical gradient value calculation and/or the horizontal gradient value calculation are based on a $[1, -1]$ filter.

27. The method of claim 26, wherein the vertical gradient values in ALF (g_v) and/or the horizontal gradient values in ALF (g_h) are calculated by using a function of $Shift(x, n)$, the $Shift(x, n)$ is defined as

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

where, x is a variable, $offset0$ is set to $(1 \ll n) \gg 1$ or $(1 \ll (n-1))$, or $offset0$ is set to 0, or $offset0 = ((1 \ll n) \gg 1) - 1$ or $((1 \ll (n-1))) - 1$, n is an integer.

28. The method of claim 27, wherein the vertical gradient values in ALF (g_v) are calculated as following:

$$g_v(x, y) = Shift(R(x, y + off1) - R(x, y - off2), prec),$$

where, x and y are variables, $R(i, j)$ indicates a reconstructed or prediction sample at coordinate (i, j) , variables $off1$, $off2$ and $prec$ are integers.

29. The method of claim 27, wherein the vertical gradient values in ALF (g_v) are calculated as following:

$$g_v(x, y) = Shift(|R(x, y + off1) - R(x, y - off2)|, prec),$$

where, x and y are variables, $R(i, j)$ indicates a reconstructed or prediction sample at coordinate (i, j) , variables $off1$, $off2$ and $prec$ are integers.

30. The method of claim 27, wherein the horizontal gradient values in ALF (g_h) are calculated as following:

$$g_h(x, y) = Shift(R(x+1, y) - R(x-1, y), prec),$$

where, x and y are variables, $R(i, j)$ indicates a reconstructed or prediction sample at coordinate (i, j) , variable $prec$ is an integer.

31. The method of claim 27, wherein the horizontal gradient values in ALF (g_h) are calculated as following:

$$g_h(x, y) = \text{Shift}(|R(x+1, y) - R(x-1, y)|, \text{prec}),$$

where, x and y are variables, $R(i, j)$ indicates a reconstructed or prediction sample at coordinate (i, j) , variable $prec$ is an integer.

32. The method of any of claims 28 - 31, wherein $off1 = off2 = 1$ and $prec = 0$.

33. The method of claim 26, wherein the vertical gradient values in ALF (g_v) and/or the horizontal gradient values in ALF (g_h) are calculated by using a function of $SatShift(x, n)$, the $SatShift(x, n)$ is defined as

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

where, x is a variable, $offset0$ and/or $offset1$ are set to $(1 \ll n) \gg 1$ or $(1 \ll (n-1))$, or $offset0$ and/or $offset1$ are set to 0, or $offset0 = offset1 = ((1 \ll n) \gg 1) - 1$ or $((1 \ll (n-1))) - 1$, n is an integer.

34. The method of claim 33, wherein the vertical gradient values in ALF (g_v) are calculated as following:

$$g_v(x, y) = \text{SatShift}(R(x, y+off1) - R(x, y-off2), \text{prec}),$$

where, x and y are variables, $R(i, j)$ indicates a reconstructed or prediction sample at coordinate (i, j) , variables $off1$, $off2$ and $prec$ are integers.

35. The method of claim 33, wherein the vertical gradient values in ALF (g_v) are calculated as following:

$$g_v(x, y) = \text{SatShift}(|R(x, y+off1) - R(x, y-off2)|, \text{prec}),$$

where, x and y are variables, $R(i, j)$ indicates a reconstructed or prediction sample at coordinate (i, j) , variables $off1$, $off2$ and $prec$ are integers.

36. The method of claim 33, wherein the horizontal gradient values in ALF (g_h) are calculated as following:

$$g_h(x,y)=\text{SatShift}(R(x+1,y)-R(x-1,y), \text{prec}),$$

where, x and y are variables, $R(i,j)$ indicates a reconstructed or prediction sample at coordinate (i,j) , variable prec is an integer.

37. The method of claim 33, wherein the horizontal gradient values in ALF (g_h) are calculated as following:

$$g_h(x,y)=\text{SatShift}(|R(x+1,y)-R(x-1,y)|, \text{prec}),$$

where, x and y are variables, $R(i,j)$ indicates a reconstructed or prediction sample at coordinate (i,j) , variable prec is an integer.

38. The method of any of claims 34-37, wherein $\text{off1}=\text{off2}=1$ and $\text{prec} = 0$.

39. The method of claim 25, wherein the vertical gradient values for all or partial samples in the first block are calculated in the same way to calculate the vertical gradient values in BDOF and averaged to obtain the vertical gradient value for the first block used in ALF.

40. The method of claim 25, wherein the horizontal gradient values for all or partial samples in the first block are calculated in the same way to calculate the horizontal gradient values in BDOF and averaged to obtain the horizontal gradient value for the first block used in ALF.

41. A method for processing video, comprising:

deriving, for a conversion between a first block of video and a bitstream representation of the first block, gradient values used in one or more coding tools by applying a sub-block level gradient calculation process, wherein the gradient values are derived for partial samples within prediction blocks of the first block; and

performing the conversion based on the derived gradient values.

42. The method of claim 41, wherein the one or more coding tools include at least one of a bi-directional optical flow (BDOF) refinement, a prediction refinement with optical flow (PROF) process and other non-ALF coding tools.
43. The method of claim 41 or 42, wherein only samples at selected coordinates are used to derive the gradient values.
44. The method of any of claims 41 - 43, wherein when gradient values are not calculated for samples at certain coordinates, the gradient values associated the samples at certain coordinates are copied from that associated with its neighbors wherein gradient vale is calculated.
45. The method of claim 44, wherein how to copy gradient values from selected samples to those remaining samples depends on gradient direction including at least one of horizontal gradient or vertical gradient.
46. A method for processing video, comprising:
 configuring, for a conversion between a first block of video and a bitstream representation of the first block, a first padding process in a first coding tool to be aligned with a second padding process in a second coding tool that is different from the first coding tool, wherein the first padding process is for padding samples out of range used to derive gradient values used in the first coding tool, and the second padding process is for padding samples out of range used to derive gradient values used in the second coding tool; and
 performing the conversion based on the configured first padding process.
47. The method of claim 46, wherein whole or partial of the first padding process is aligned with the corresponding whole or partial of the second padding process.
48. The method of claim 47, wherein the first coding tool is a bi-directional optical flow (BDOF) refinement, and the second coding tool is an adaptive loop filtering (ALF) process.

49. The method of claim 47, wherein the first coding tool is an adaptive loop filtering (ALF) process, and the second coding tool is a bi-directional optical flow (BDOF) refinement.
50. The method of claim 47, wherein the first coding tool is a bi-directional optical flow (BDOF) refinement, and the second coding tool is a prediction refinement with optical flow (PROF) process.
51. The method of claim 47, wherein the first coding tool is a prediction refinement with optical flow (PROF) process, and the second coding tool is a bi-directional optical flow (BDOF) refinement.
52. The method of claim 47, wherein the first coding tool is a prediction refinement with optical flow (PROF) process, and the second coding tool is an adaptive loop filtering (ALF) process.
53. The method of claim 47, wherein the first coding tool is an adaptive loop filtering (ALF) process, and the second coding tool is a prediction refinement with optical flow (PROF) process.
54. The method of any of claims 1-53, wherein the first coding tool and/or the second coding tool include coding tools that rely on calculation of gradient values.
55. The method of any one of claims 1 - 54, wherein the conversion generates the first block of video from the bitstream representation.
56. The method of any one of claims 1 - 54, wherein the conversion generates the bitstream representation from the first block of video.
57. An apparatus in a video system comprising a processor and a non-transitory memory with instructions thereon, wherein the instructions upon execution by the processor, cause the processor to implement the method in any one of claims 1 to 56.

58. A computer program product stored on a non-transitory computer readable media, the computer program product including program code for carrying out the method in any one of claims 1 to 56.

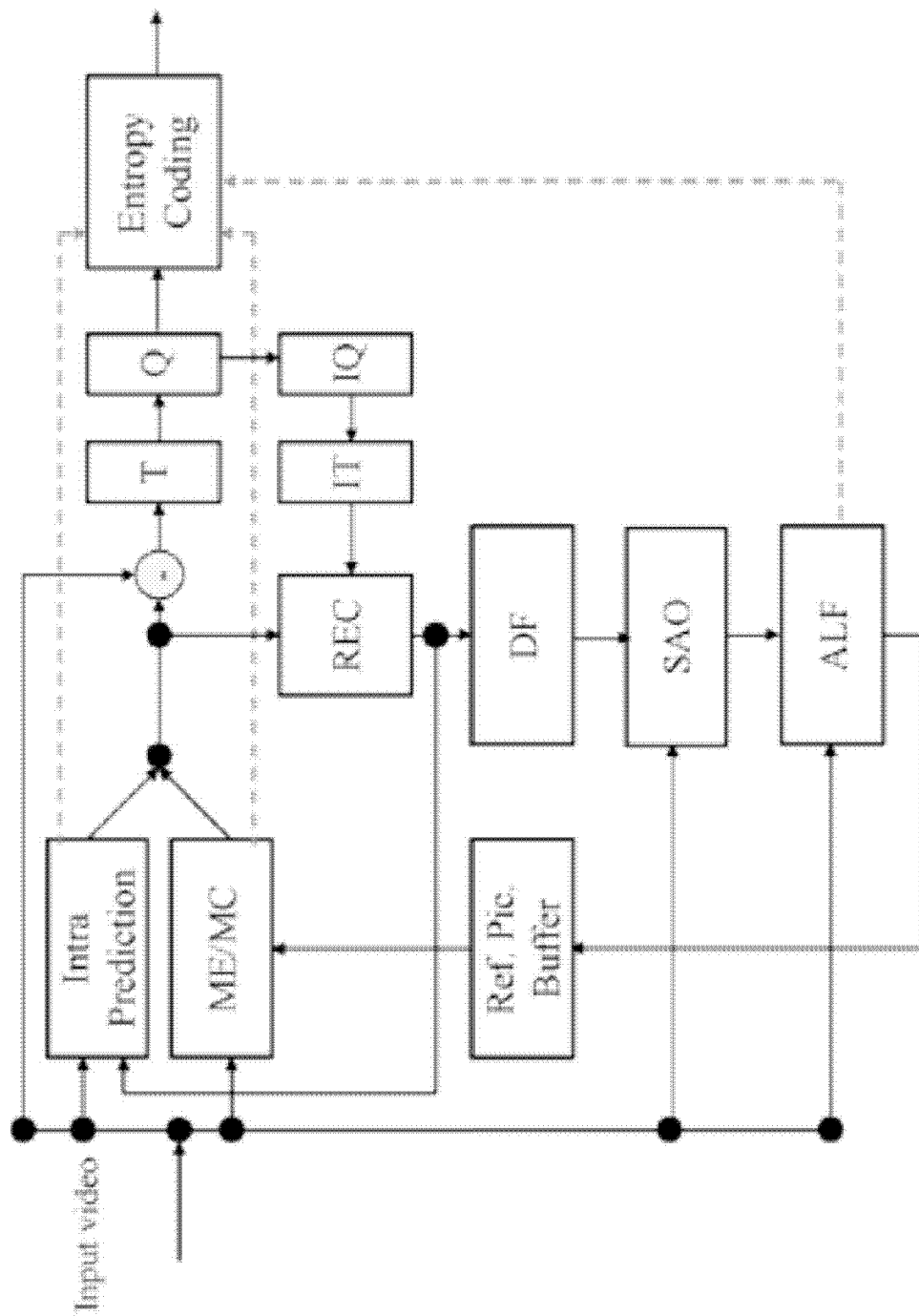
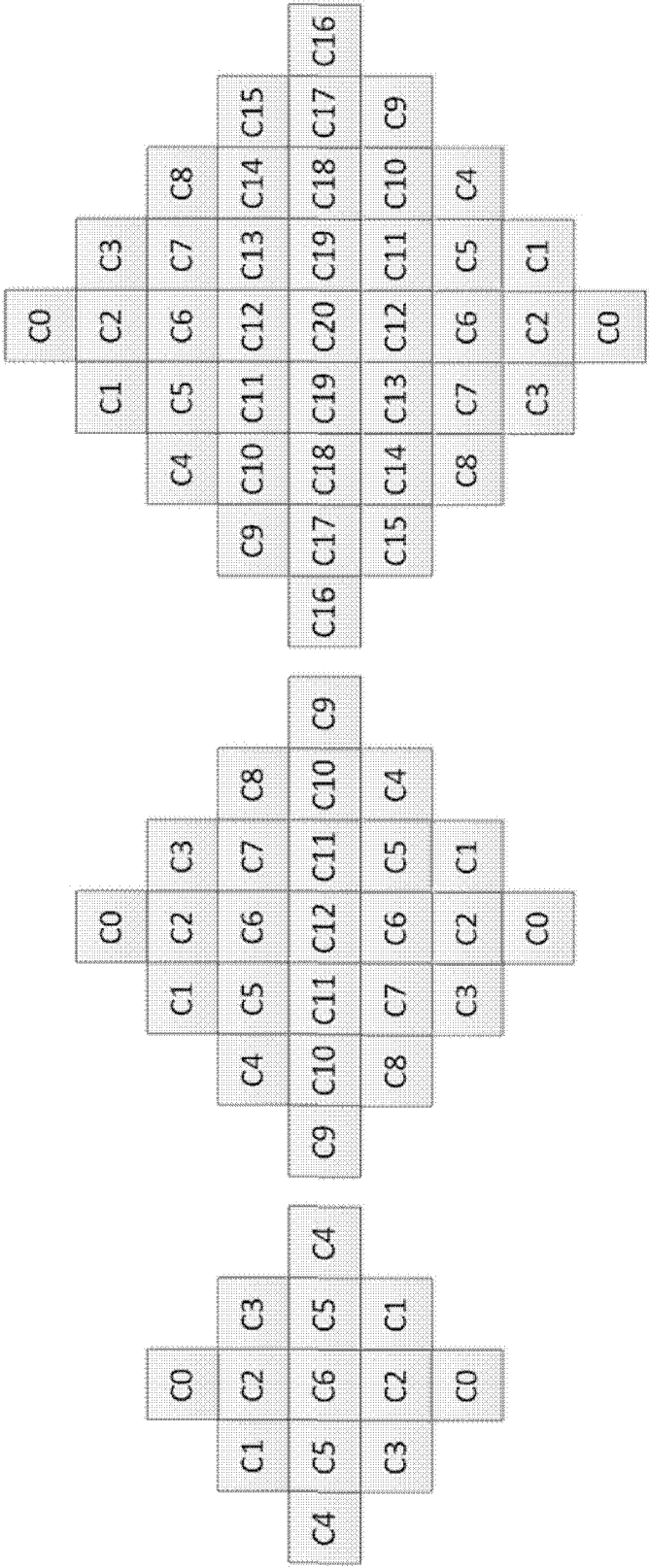


FIG. 1



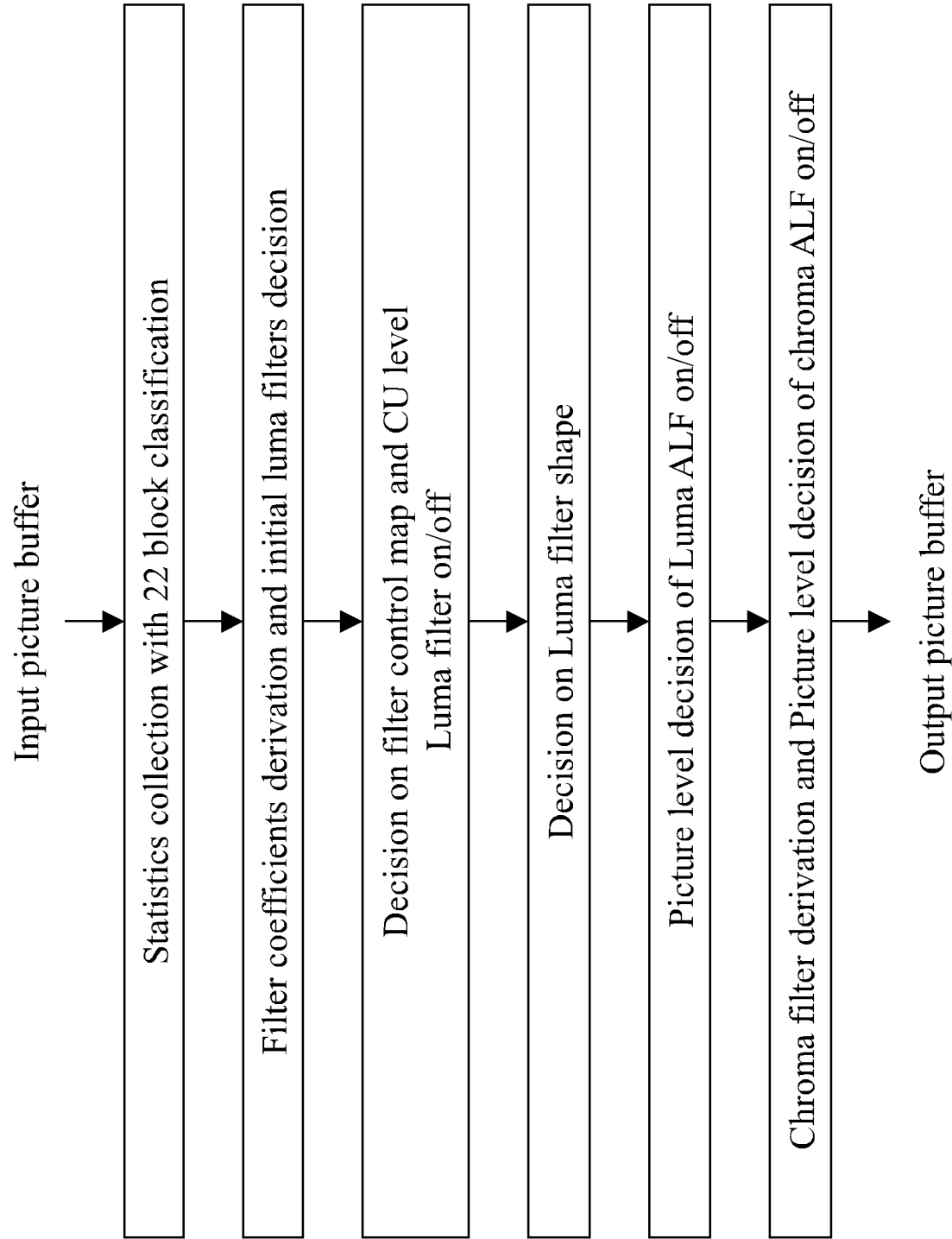


FIG. 3

	H		H		H		H
H		H		H		H	
	H		H		H		H
H		H		H		H	
	H		H		H		H
H		H		H		H	
	H		H		H		H
H		H		H		H	

FIG. 4B

	V		V		V		V
V		V		V		V	
	V		V		V		V
V		V		V		V	
	V		V		V		V
V		V		V		V	
	V		V		V		V
V		V		V		V	

FIG. 4A

D2		D2		D2		D2		D2	
	D2		D2		D2		D2		D2
D2		D2		D2		D2		D2	
	D2		D2		D2		D2		D2
D2		D2		D2		D2		D2	
	D2		D2		D2		D2		D2
D2		D2		D2		D2		D2	
	D2		D2		D2		D2		D2
D2		D2		D2		D2		D2	
	D2		D2		D2		D2		D2

FIG. 4D

$D1$		$D1$		$D1$		$D1$	
	$D1$		$D1$		$D1$		$D1$
$D1$		$D1$		$D1$		$D1$	
	$D1$		$D1$		$D1$		$D1$
$D1$		$D1$		$D1$		$D1$	
	$D1$		$D1$		$D1$		$D1$
$D1$		$D1$		$D1$		$D1$	
	$D1$		$D1$		$D1$		$D1$

FIG. 4C

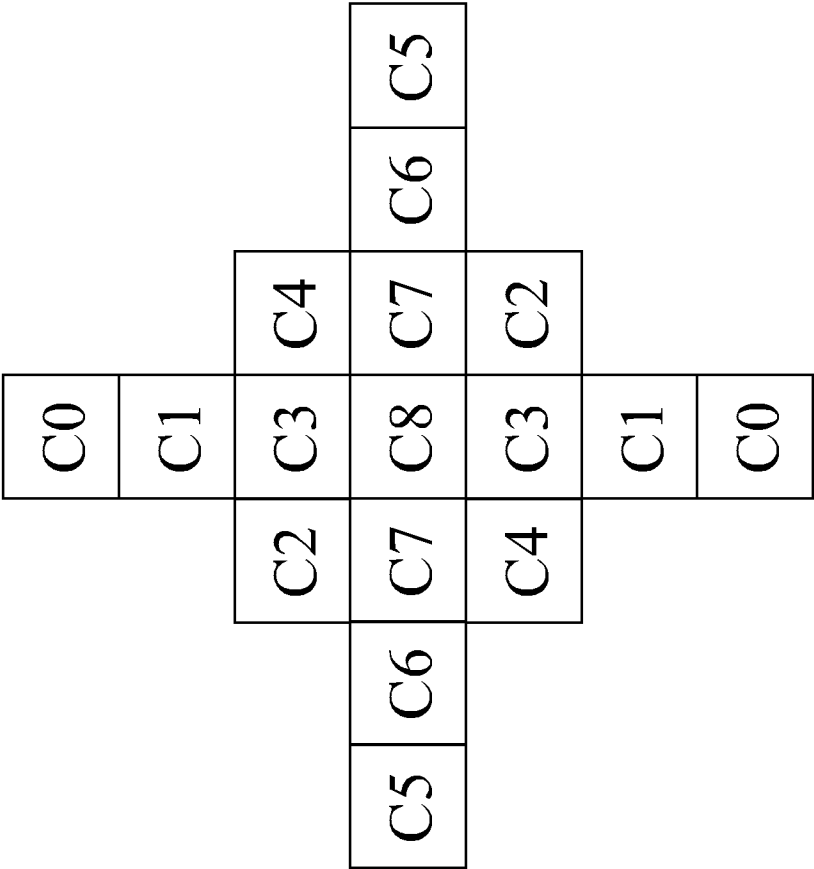


FIG. 5

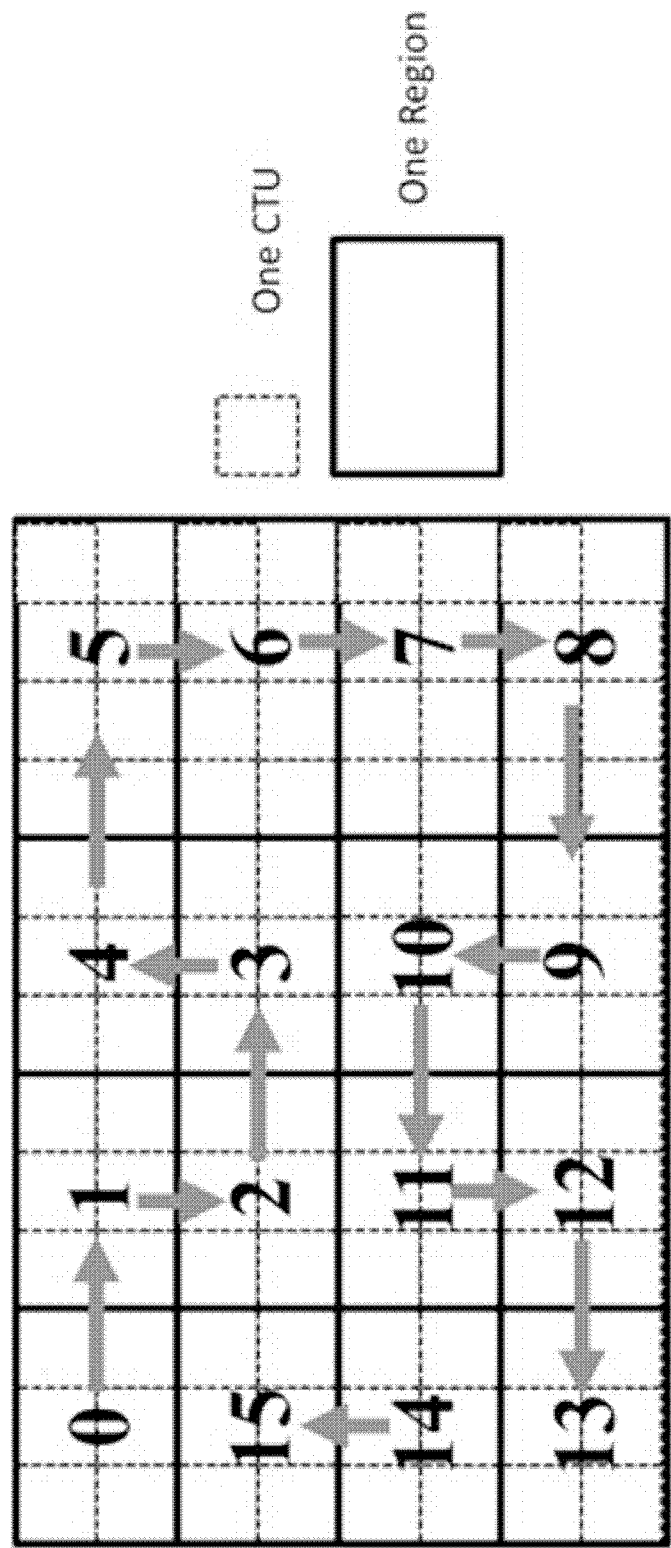


FIG. 6

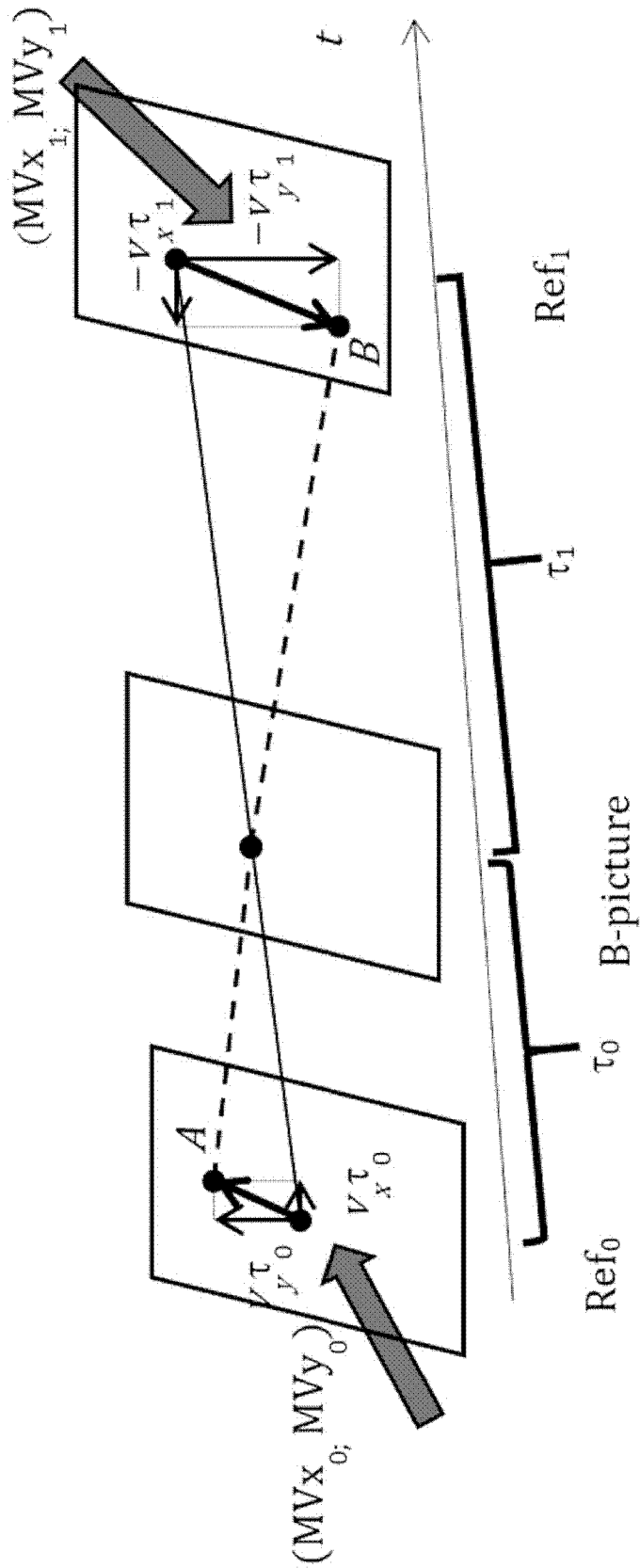


FIG. 7

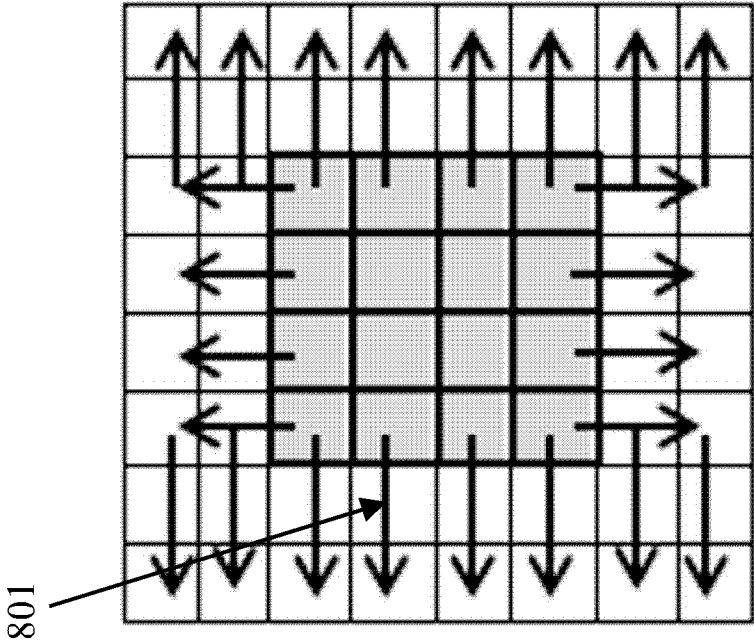


FIG. 8A

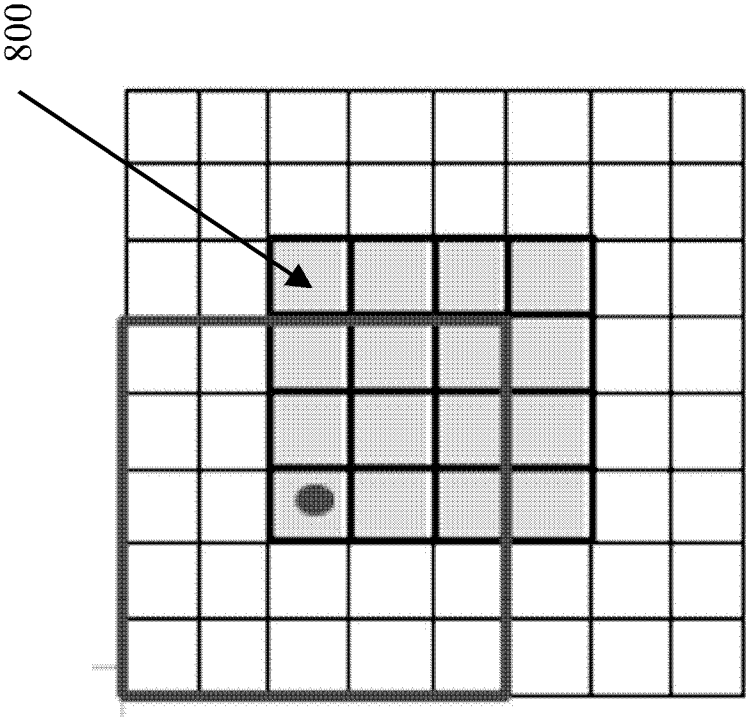


FIG. 8B

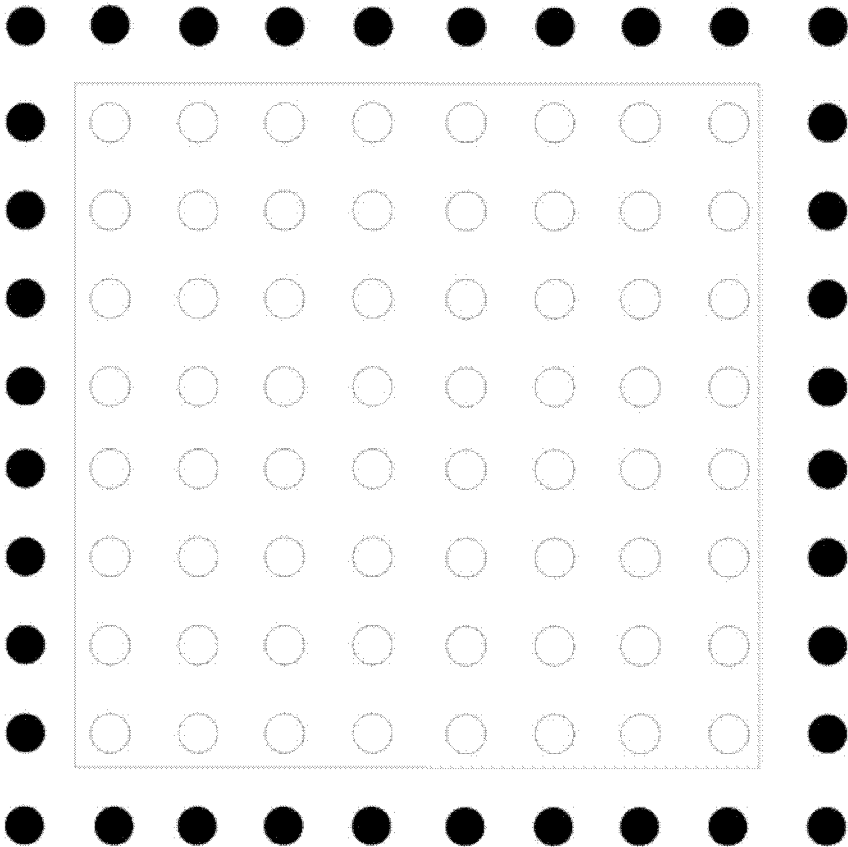


FIG. 9

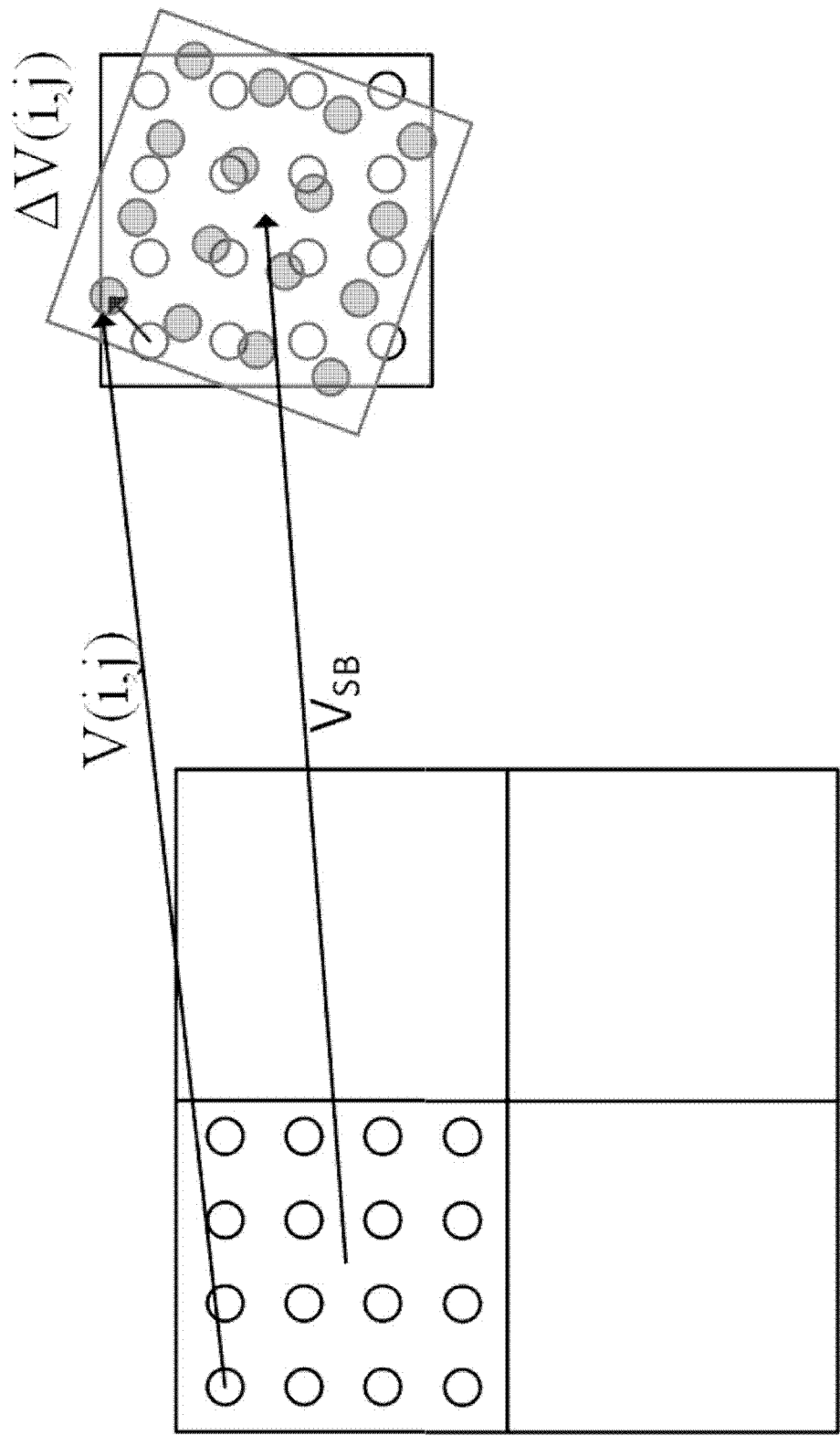


FIG. 10

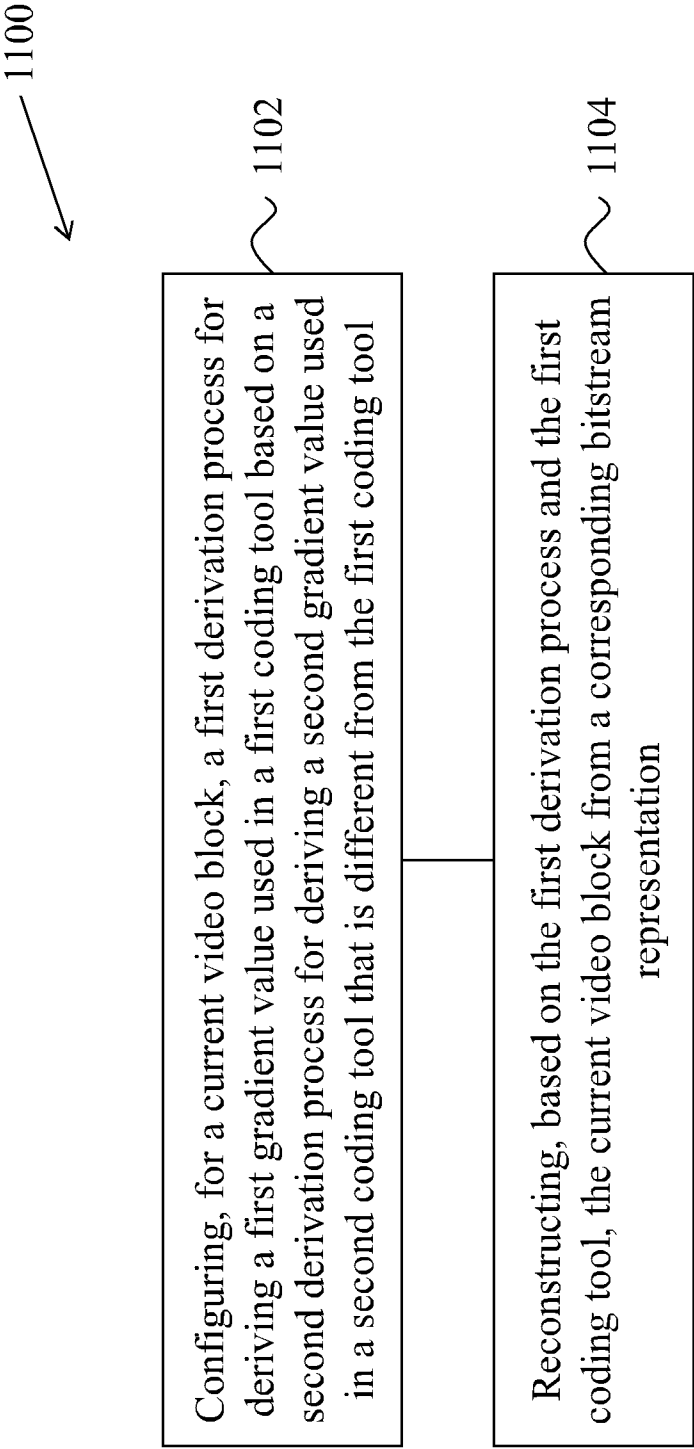


FIG. 11A

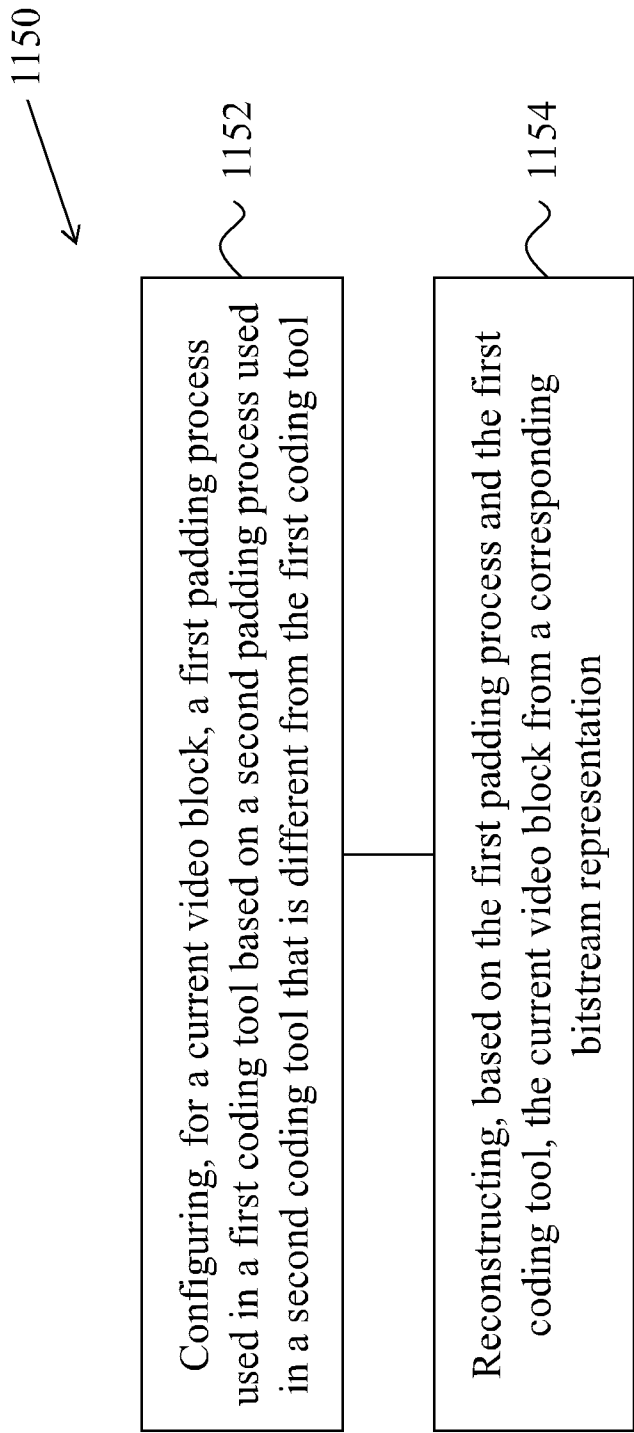


FIG. 11B

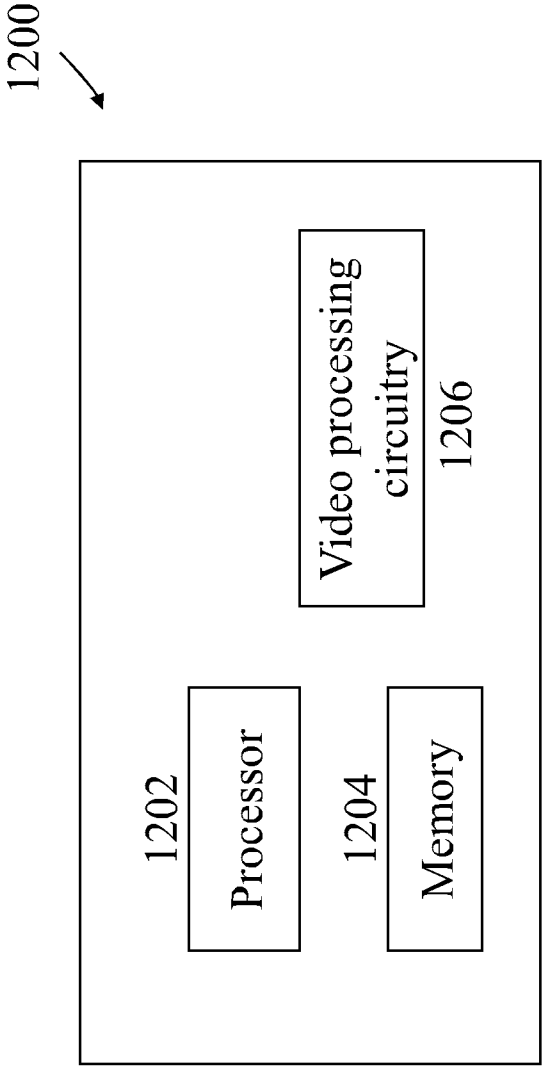


FIG. 12

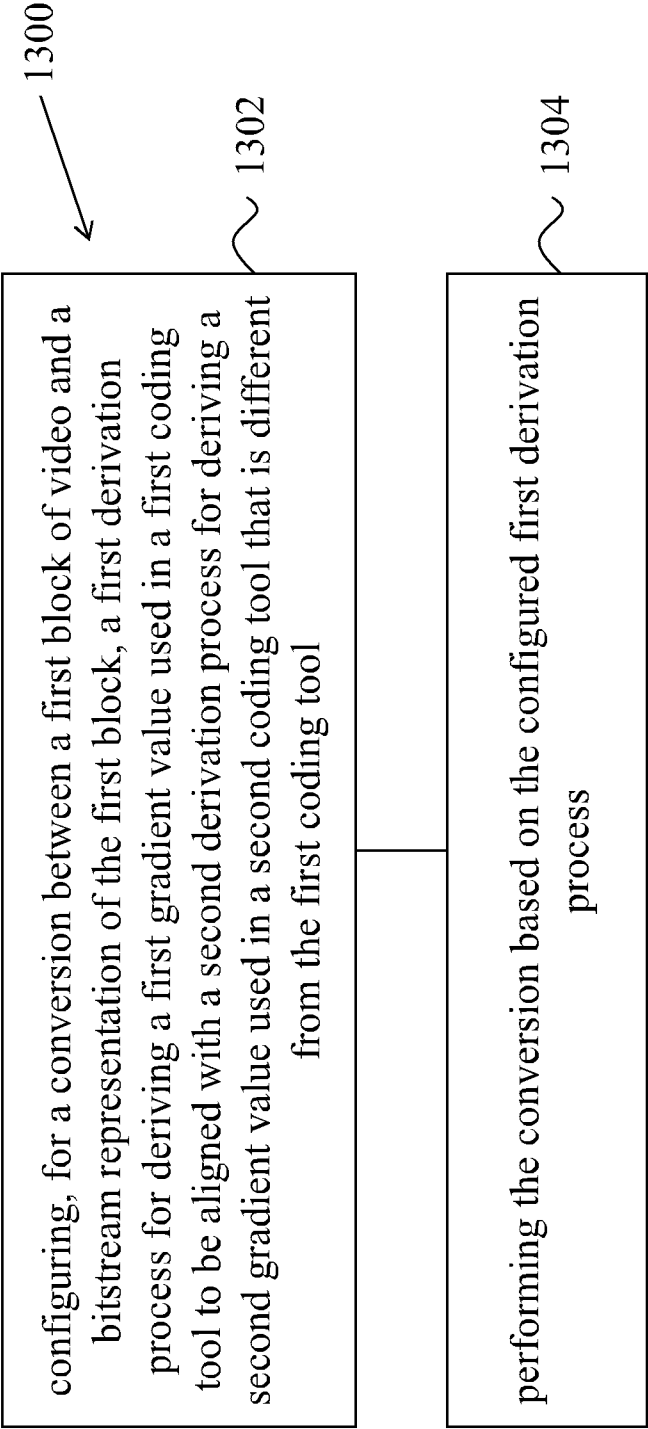


FIG. 13

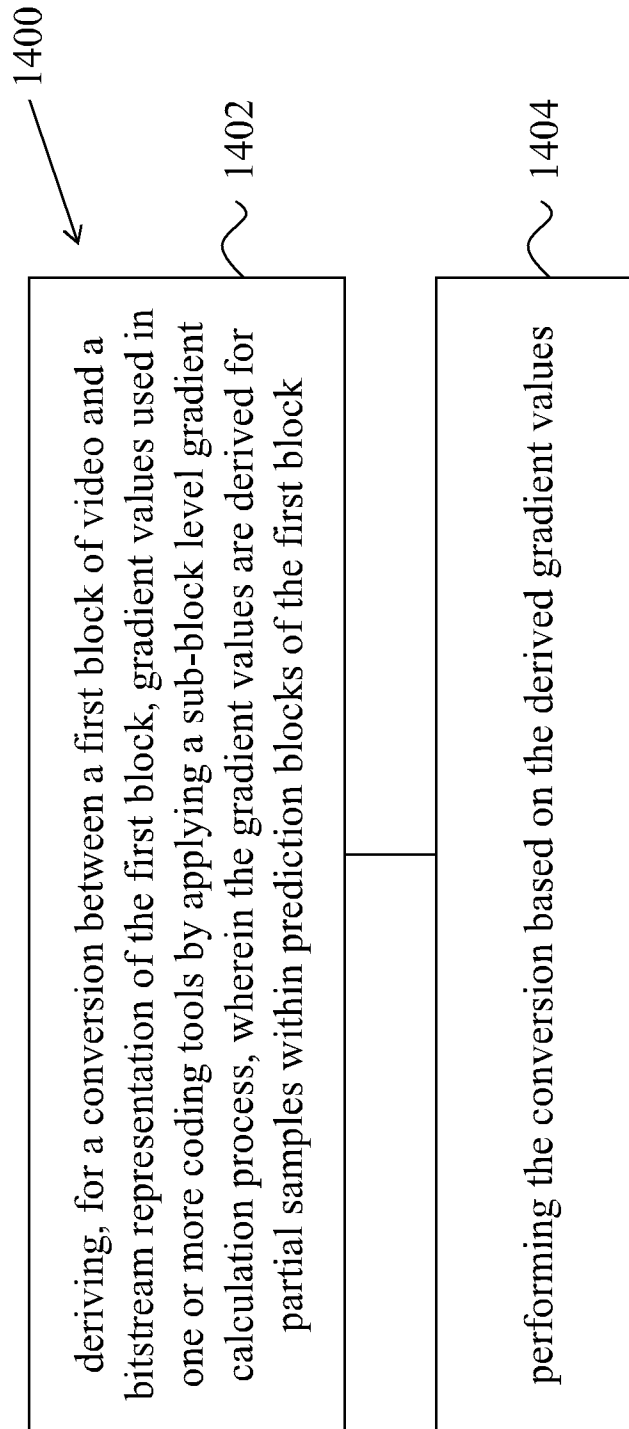


FIG. 14

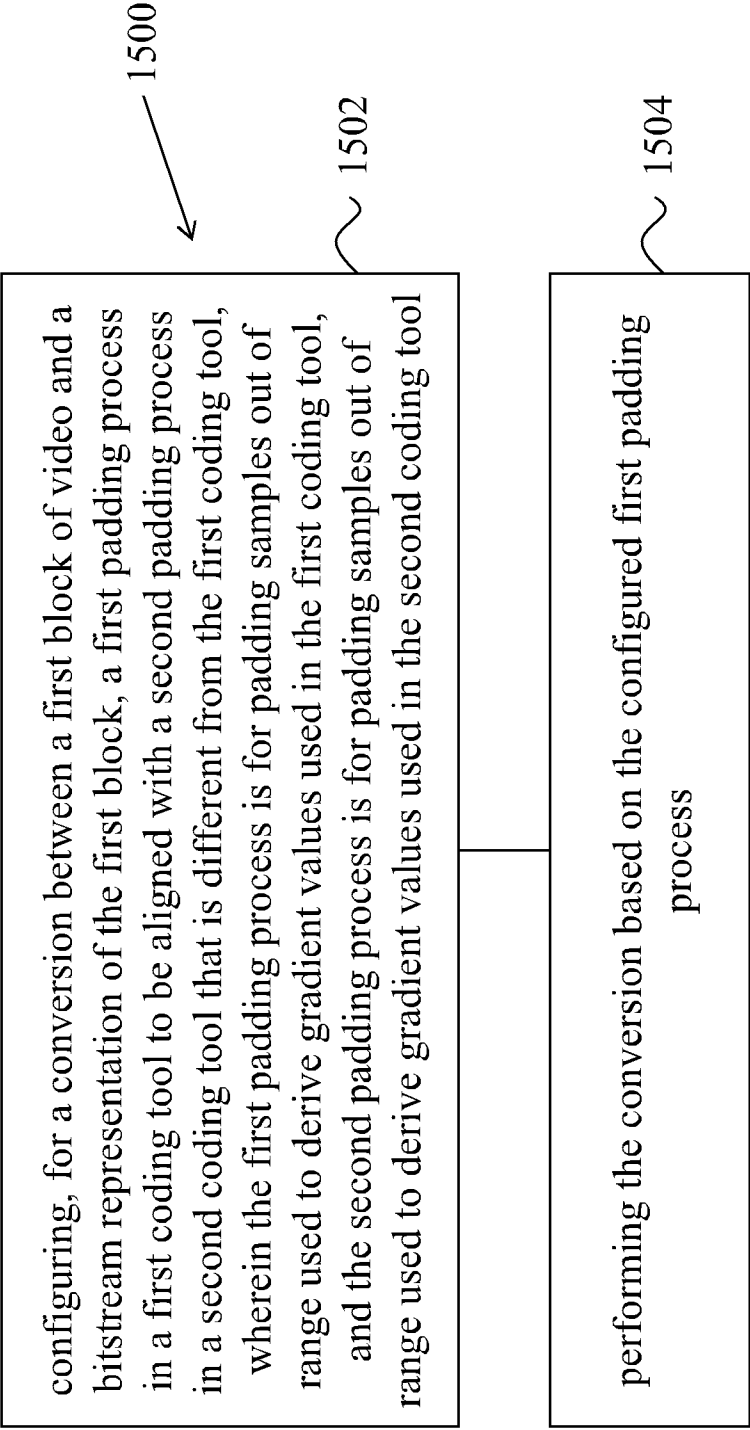


FIG. 15

INTERNATIONAL SEARCH REPORT

International application No.

PCT/CN2020/082038

A. CLASSIFICATION OF SUBJECT MATTER

H04N 19/176(2014.01)i; H04N 19/577(2014.01)i; H04N 19/82(2014.01)i

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

H04N

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

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

CNKI, CNPAT, WPI, EPODOC, JNET: ALF, BDOF, BIO, PROF, adaptive loop filtering, bi-directional optical flow, prediction refinement with optical flow, align, interact, combine, gradient, derive, padding

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	WO 2018166357 A1 (MEDIATEK INC.) 20 September 2018 (2018-09-20) page 5 line 37- page 7 line 18	1-58
A	CN 109417620 A (PANASONIC INTELLECTUAL PROPERTY MANAGEME CO., LTD.) 01 March 2019 (2019-03-01) the whole document	1-58
A	WO 2018221631 A1 (PANASONIC INTELLECTUAL PROPERTY CORPORATION OF AMERICA) 06 December 2018 (2018-12-06) the whole document	1-58
A	WO 2019010156 A1 (VID SCALE, INC.) 10 January 2019 (2019-01-10) the whole document	1-58
A	Vadim Seregin et al. "CE5: Summary Report on Adaptive Loop Filter" <i>Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11 14th Meeting: Geneva, CH, 19-27 March 2019, 27 March 2019 (2019-03-27),</i> the whole document	1-58



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

19 June 2020

Date of mailing of the international search report

29 June 2020

Name and mailing address of the ISA/CN

National Intellectual Property Administration, PRC
6, Xitucheng Rd., Jimen Bridge, Haidian District, Beijing
100088
China

Authorized officer

JIN,Xiaocong

Facsimile No. (86-10)62019451

Telephone No. 86-10-53961714

INTERNATIONAL SEARCH REPORT

International application No.

PCT/CN2020/082038

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	Ching-Yeh Chen et al. "Description of Core Experiment 5 (CE5): Adaptive Loop Filter" <i>Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11 13th Meeting: Marrakech, MA, 9–18 Jan. 2019</i> , 18 January 2019 (2019-01-18), the whole document	1-58

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.

PCT/CN2020/082038

Patent document cited in search report			Publication date (day/month/year)	Patent family member(s)			Publication date (day/month/year)
WO	2018166357	A1	20 September 2018	EP	3586513	A1	01 January 2020
				TW	201840190	A	01 November 2018
				CN	110476424	A	19 November 2019
				BR	112019018922	A2	14 April 2020
CN	109417620	A	01 March 2019	US	2019075293	A1	07 March 2019
				JP	WO2017164297	A1	31 January 2019
				WO	2017164297	A1	28 September 2017
				DE	112017001540	T5	06 December 2018
				TW	201803350	A	16 January 2018
WO	2018221631	A1	06 December 2018	US	2020099952	A1	26 March 2020
WO	2019010156	A1	10 January 2019	CN	110832858	A	21 February 2020
				KR	20200033228	A	27 March 2020
				EP	3649780	A1	13 May 2020