



(51) International Patent Classification:
H04N 19/149 (2014.01)

(21) International Application Number:
PCT/CN2024/097635

(22) International Filing Date:
05 June 2024 (05.06.2024)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
PCT/CN2023/098969
07 June 2023 (07.06.2023) CN

(71) Applicants: **DOUYIN VISION 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: **WANG, Yang**; Jinritoutiao Post Office, China
Satellite Communications Tower, No. 63, Zhichun Road,
Haidian District, Beijing 100080 (CN). **ZHANG, Kai**;
12655 West Jefferson Boulevard, Sixth Floor, Suite No.
137, Los Angeles, California 90066 (US). **ZHANG, Li**;

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

(74) Agent: **SHIHUI PARTNERS**; 42/F, Tower C, Beijing
Yintai Centre, No. 2 Jianguomenwai Avenue, Chaoyang
District, Beijing 100022 (CN).

(81) Designated States (*unless otherwise indicated, for every
kind of national protection available*): AE, AG, AL, AM,
AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ,
CA, CH, CL, CN, CO, CR, CU, CV, CZ, DE, DJ, DK, DM,
DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT,
HN, HR, HU, ID, IL, IN, IQ, IR, IS, IT, JM, JO, JP, KE, KG,
KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY,
MA, MD, MG, MK, MN, MU, 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, CV,
GH, GM, KE, LR, LS, MW, MZ, NA, RW, SC, 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, ME, MK, MT, NL, NO, PL, PT, RO, RS, SE,

(54) Title: METHOD, APPARATUS, AND MEDIUM FOR VIDEO PROCESSING

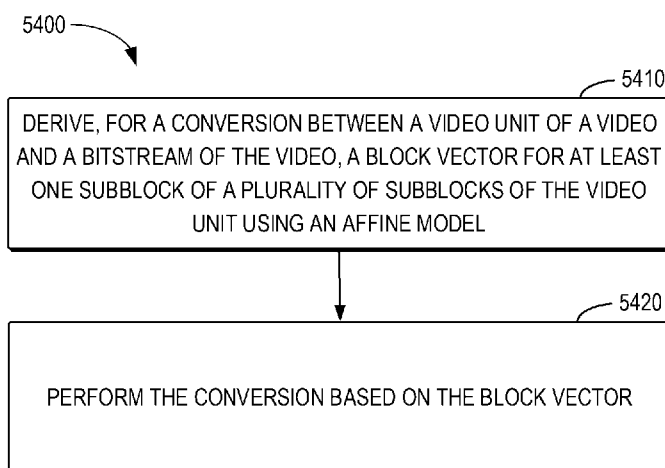


Fig. 54

(57) Abstract: Embodiments of the present disclosure provide a solution for video processing. A method for video processing is proposed. The method comprises: deriving, for a conversion between a video unit of a video and a bitstream of the video, a block vector (BV) for at least one subblock of a plurality of subblocks of the video unit using an affine model, wherein a same block size is used for all of the plurality of subblocks, and/or at least one of: a parameter of an affine mode or a position of the video unit is applied to a following video unit of the video unit; and performing the conversion based on the BV.



SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

- *with international search report (Art. 21(3))*
- *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

METHOD, APPARATUS, AND MEDIUM FOR VIDEO PROCESSING

FIELDS

[0001] Embodiments of the present disclosure relates generally to video processing techniques, and more particularly, to intra block copy (IBC) with affine model.

BACKGROUND

[0002] In nowadays, digital video capabilities are being applied in various aspects of peoples' lives. Multiple types of video compression technologies, such as MPEG-2, MPEG-4, ITU-TH.263, ITU-TH.264/MPEG-4 Part 10 Advanced Video Coding (AVC), ITU-TH.265 high efficiency video coding (HEVC) standard, versatile video coding (VVC) standard, have been proposed for video encoding/decoding. However, in current design of IBC in video coding, only translational motion is considered within the picture. Therefore, coding efficiency of video coding techniques is generally expected to be further improved.

SUMMARY

[0003] Embodiments of the present disclosure provide a solution for video processing.

[0004] In a first aspect, a method for video processing is proposed. The method comprises: deriving, for a conversion between a video unit of a video and a bitstream of the video, a block vector (BV) for at least one subblock of a plurality of subblocks of the video unit using an affine model, wherein a same block size is used for all of the plurality of subblocks, and/or at least one of: a parameter of an affine mode or a position of the video unit is applied to a following video unit of the video unit; and performing the conversion based on the BV. Compared with the conventional solution, the method in accordance with the first aspect of the present disclosure can improve the coding performance to handle the textures by the affine model.

[0005] In a second aspect, an apparatus for video processing is proposed. The apparatus comprises a processor and a non-transitory memory with instructions thereon. The instructions upon execution by the processor, cause the processor to perform a method in accordance with the first aspect of the present disclosure.

[0006] In a third aspect, a non-transitory computer-readable storage medium is proposed. The non-transitory computer-readable storage medium stores instructions that cause a processor to perform a method in accordance with the first aspect of the present disclosure.

[0007] In a fourth aspect, another non-transitory computer-readable recording medium is proposed. The non-transitory computer-readable recording medium stores a bitstream of a video which is generated by a method performed by an apparatus for video processing. The method comprises: deriving a block vector (BV) for at least one subblock of a plurality of subblocks of a video unit of the video using an affine model, wherein a same block size is used for all of the plurality of subblocks, and/or at least one of: a parameter of an affine mode or a position of the video unit is applied to a following video unit of the video unit; and generating the bitstream based on the BV.

[0008] In a fifth aspect, a method for storing a bitstream of a video is proposed. The method comprises: deriving a block vector (BV) for at least one subblock of a plurality of subblocks of a video unit of the video using an affine model, wherein a same block size is used for all of the plurality of subblocks, and/or at least one of: a parameter of an affine mode or a position of the video unit is applied to a following video unit of the video unit; generating the bitstream based on the BV; and storing the bitstream in a non-transitory computer-readable recording medium.

[0009] This Summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description. This Summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used to limit the scope of the claimed subject matter.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] Through the following detailed description with reference to the accompanying drawings, the above and other objectives, features, and advantages of example embodiments of the present disclosure will become more apparent. In the example embodiments of the present disclosure, the same reference numerals usually refer to the same components.

[0011] Fig. 1 illustrates a block diagram that illustrates an example video coding system, in accordance with some embodiments of the present disclosure;

[0012] Fig. 2 illustrates a block diagram that illustrates a first example video encoder, in accordance with some embodiments of the present disclosure;

[0013] Fig. 3 illustrates a block diagram that illustrates an example video decoder, in accordance with some embodiments of the present disclosure;

[0014] Fig. 4 shows an example of encoder block diagram;

[0015] Fig. 5 shows 67 intra prediction modes;

[0016] Fig. 6 shows reference samples for wide-angular intra prediction;

[0017] Fig. 7 shows problem of discontinuity in case of directions beyond 45° ;

[0018] Fig. 8 shows MMVD search point;

[0019] Fig. 9 is illustration for symmetrical MVD mode;

[0020] Fig. 10 shows extended CU regions used in BDOF;

[0021] Fig. 11a and Fig. 11b show control point based affine motion model;

[0022] Fig. 12 shows affine MVF per subblock;

[0023] Fig. 13 shows locations of inherited affine motion predictors;

[0024] Fig. 14 shows control point motion vector inheritance;

[0025] Fig. 15 shows locations of Candidates position for constructed affine merge mode;

- [0026] Fig. 16 is an illustration of motion vector usage for proposed combined method;
- [0027] Fig. 17 shows subblock MV VSB and pixel $\Delta v(i, j)$;
- [0028] Figs. 18a and 18b show the SbTMVP process in VVC;
- [0029] Fig. 19 shows local illumination compensation;
- [0030] Fig. 20 shows no subsampling for the short side;
- [0031] Fig. 21 shows decoding side motion vector refinement;
- [0032] Fig. 22 shows diamond regions in the search area;
- [0033] Fig. 23 shows positions of spatial merge candidate;
- [0034] Fig. 24 shows candidate pairs considered for redundancy check of spatial merge candidates;
- [0035] Fig. 25 is an illustration of motion vector scaling for temporal merge candidate;
- [0036] Fig. 26 shows candidate positions for temporal merge candidate, C0 and C1;
- [0037] Fig. 27 shows VVC spatial neighboring blocks of the current block;
- [0038] Fig. 28 is an illustration of virtual block in the i-th search round;
- [0039] Fig. 29 shows examples of the GPM splits grouped by identical angles;
- [0040] Fig. 30 shows uni-prediction MV selection for geometric partitioning mode;
- [0041] Fig. 31 shows exemplified generation of a bending weight w_0 using geometric partitioning mode;
- [0042] Fig. 32 shows spatial neighboring blocks used to derive the spatial merge candidates;
- [0043] Fig. 33 shows template matching performs on a search area around initial MV;
- [0044] Fig. 34 is an illustration of sub-blocks where OBMC applies;

- [0045] Fig. 35 shows SBT position, type and transform type;
- [0046] Fig. 36 shows neighbouring samples used for calculating SAD;
- [0047] Fig. 37 shows neighbouring samples used for calculating SAD for sub-CU level motion information;
- [0048] Fig. 38 shows the sorting process;
- [0049] Fig. 39 shows reorder process in encoder;
- [0050] Fig. 40 shows reorder process in decoder;
- [0051] Fig. 41 is an illustration of the extended reference area;
- [0052] Fig. 42 shows IBC reference region depending on current CU position;
- [0053] Fig. 43 shows examples of symmetry in screen content pictures;
- [0054] Fig. 44a is an illustration of BV adjustment for horizontal flip and Fig. 44b is an illustration of BV adjustment for vertical flip, respectively;
- [0055] Fig. 45 shows intra template matching search area used;
- [0056] Fig. 46 shows an illustrations of the template area;
- [0057] Fig. 47 shows spatial part of the convolutional filter;
- [0058] Fig. 48 shows reference area (with its paddings) used to derive the filter coefficients;
- [0059] Fig. 49 shows four Sobel based gradient patterns for GLM;
- [0060] Fig. 50 shows unreconstructed samples (shaded) in the reference block are padded by copying their prediction samples;
- [0061] Fig. 51 shows the five locations in reconstructed luma samples;
- [0062] Fig. 52 shows the prediction process of DBV mode;
- [0063] Fig. 53 shows a spatial part of the filter;

[0064] Fig. 54 illustrates a flowchart of a method for video processing in accordance with embodiments of the present disclosure;

[0065] Fig. 55 illustrates a block diagram of a computing device in which various embodiments of the present disclosure can be implemented.

[0066] Throughout the drawings, the same or similar reference numerals usually refer to the same or similar elements.

DETAILED DESCRIPTION

[0067] Principle of the present disclosure will now be described with reference to some embodiments. It is to be understood that these embodiments are described only for the purpose of illustration and help those skilled in the art to understand and implement the present disclosure, without suggesting any limitation as to the scope of the disclosure. The disclosure described herein can be implemented in various manners other than the ones described below.

[0068] In the following description and claims, unless defined otherwise, all technical and scientific terms used herein have the same meaning as commonly understood by one of ordinary skills in the art to which this disclosure belongs.

[0069] References in the present disclosure to “one embodiment,” “an embodiment,” “an example embodiment,” and the like indicate that the embodiment described may include a particular feature, structure, or characteristic, but it is not necessary that every embodiment includes the particular feature, structure, or characteristic. Moreover, such phrases are not necessarily referring to the same embodiment. Further, when a particular feature, structure, or characteristic is described in connection with an example embodiment, it is submitted that it is within the knowledge of one skilled in the art to affect such feature, structure, or characteristic in connection with other embodiments whether or not explicitly described.

[0070] It shall be understood that although the terms “first” and “second” etc. may be

used herein to describe various elements, these elements should not be limited by these terms. These terms are only used to distinguish one element from another. For example, a first element could be termed a second element, and similarly, a second element could be termed a first element, without departing from the scope of example embodiments. As used herein, the term “and/or” includes any and all combinations of one or more of the listed terms.

[0071] The terminology used herein is for the purpose of describing particular embodiments only and is not intended to be limiting of example embodiments. 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. It will be further understood that the terms “comprises”, “comprising”, “has”, “having”, “includes” and/or “including”, when used herein, specify the presence of stated features, elements, and/or components etc., but do not preclude the presence or addition of one or more other features, elements, components and/ or combinations thereof.

Example Environment

[0072] Fig. 1 is a block diagram that illustrates an example video coding system 100 that may utilize the techniques of this disclosure. As shown, the video coding system 100 may include a source device 110 and a destination device 120. The source device 110 can be also referred to as a video encoding device, and the destination device 120 can be also referred to as a video decoding device. In operation, the source device 110 can be configured to generate encoded video data and the destination device 120 can be configured to decode the encoded video data generated by the source device 110. The source device 110 may include a video source 112, a video encoder 114, and an input/output (I/O) interface 116.

[0073] The video source 112 may include a source such as a video capture device. Examples of the video capture device include, but are not limited to, an interface to receive video data from a video content provider, a computer graphics system for generating video

data, and/or a combination thereof.

[0074] The video data may comprise one or more pictures. The video encoder 114 encodes the video data from the video source 112 to generate a bitstream. The bitstream may include a sequence of bits that form a coded representation of the video data. The bitstream may include coded pictures and associated data. The coded picture is a coded representation of a picture. The associated data may include sequence parameter sets, picture parameter sets, and other syntax structures. The I/O interface 116 may include a modulator/demodulator and/or a transmitter. The encoded video data may be transmitted directly to destination device 120 via the I/O interface 116 through the network 130A. The encoded video data may also be stored onto a storage medium/server 130B for access by destination device 120.

[0075] The destination device 120 may include an I/O interface 126, a video decoder 124, and a display device 122. The I/O interface 126 may include a receiver and/or a modem. The I/O interface 126 may acquire encoded video data from the source device 110 or the storage medium/server 130B. The video decoder 124 may decode the encoded video data. The display device 122 may display the decoded video data to a user. The display device 122 may be integrated with the destination device 120, or may be external to the destination device 120 which is configured to interface with an external display device.

[0076] The video encoder 114 and the video decoder 124 may operate according to a video compression standard, such as the High Efficiency Video Coding (HEVC) standard, Versatile Video Coding (VVC) standard and other current and/or further standards.

[0077] Fig. 2 is a block diagram illustrating an example of a video encoder 200, which may be an example of the video encoder 114 in the system 100 illustrated in Fig. 1, in accordance with some embodiments of the present disclosure.

[0078] The video encoder 200 may be configured to implement any or all of the techniques of this disclosure. In the example of Fig. 2, the video encoder 200 includes a plurality of functional components. The techniques described in this disclosure may

be shared among the various components of the video encoder 200. In some examples, a processor may be configured to perform any or all of the techniques described in this disclosure.

[0079] In some embodiments, the video encoder 200 may include a partition unit 201, a predication unit 202 which may include a mode select unit 203, a motion estimation unit 204, a motion compensation unit 205 and an intra-prediction unit 206, a residual generation unit 207, a transform unit 208, a quantization unit 209, an inverse quantization unit 210, an inverse transform unit 211, a reconstruction unit 212, a buffer 213, and an entropy encoding unit 214.

[0080] In other examples, the video encoder 200 may include more, fewer, or different functional components. In an example, the predication unit 202 may include an intra block copy (IBC) unit. The IBC unit may perform predication in an IBC mode in which at least one reference picture is a picture where the current video block is located.

[0081] Furthermore, although some components, such as the motion estimation unit 204 and the motion compensation unit 205, may be integrated, but are represented in the example of Fig. 2 separately for purposes of explanation.

[0082] The partition unit 201 may partition a picture into one or more video blocks. The video encoder 200 and the video decoder 300 may support various video block sizes.

[0083] The mode select unit 203 may select one of the coding modes, intra or inter, e.g., based on error results, and provide the resulting intra-coded or inter-coded block to a residual generation unit 207 to generate residual block data and to a reconstruction unit 212 to reconstruct the encoded block for use as a reference picture. In some examples, the mode select unit 203 may select a combination of intra and inter predication (CIIP) mode in which the predication is based on an inter predication signal and an intra predication signal. The mode select unit 203 may also select a resolution for a motion vector (e.g., a sub-pixel or integer pixel precision) for the block in the case of inter-predication.

[0084] To perform inter prediction on a current video block, the motion estimation unit 204 may generate motion information for the current video block by comparing one or more reference frames from buffer 213 to the current video block. The motion compensation unit 205 may determine a predicted video block for the current video block based on the motion information and decoded samples of pictures from the buffer 213 other than the picture associated with the current video block.

[0085] The motion estimation unit 204 and the motion compensation unit 205 may perform different operations for a current video block, for example, depending on whether the current video block is in an I-slice, a P-slice, or a B-slice. As used herein, an “I-slice” may refer to a portion of a picture composed of macroblocks, all of which are based upon macroblocks within the same picture. Further, as used herein, in some aspects, “P-slices” and “B-slices” may refer to portions of a picture composed of macroblocks that are not dependent on macroblocks in the same picture.

[0086] In some examples, the motion estimation unit 204 may perform uni-directional prediction for the current video block, and the motion estimation unit 204 may search reference pictures of list 0 or list 1 for a reference video block for the current video block. The motion estimation unit 204 may then generate a reference index that indicates the reference picture in list 0 or list 1 that contains the reference video block and a motion vector that indicates a spatial displacement between the current video block and the reference video block. The motion estimation unit 204 may output the reference index, a prediction direction indicator, and the motion vector as the motion information of the current video block. The motion compensation unit 205 may generate the predicted video block of the current video block based on the reference video block indicated by the motion information of the current video block.

[0087] Alternatively, in other examples, the motion estimation unit 204 may perform bi-directional prediction for the current video block. The motion estimation unit 204 may search the reference pictures in list 0 for a reference video block for the current video

block and may also search the reference pictures in list 1 for another reference video block for the current video block. The motion estimation unit 204 may then generate reference indexes that indicate the reference pictures in list 0 and list 1 containing the reference video blocks and motion vectors that indicate spatial displacements between the reference video blocks and the current video block. The motion estimation unit 204 may output the reference indexes and the motion vectors of the current video block as the motion information of the current video block. The motion compensation unit 205 may generate the predicted video block of the current video block based on the reference video blocks indicated by the motion information of the current video block.

[0088] In some examples, the motion estimation unit 204 may output a full set of motion information for decoding processing of a decoder. Alternatively, in some embodiments, the motion estimation unit 204 may signal the motion information of the current video block with reference to the motion information of another video block. For example, the motion estimation unit 204 may determine that the motion information of the current video block is sufficiently similar to the motion information of a neighboring video block.

[0089] In one example, the motion estimation unit 204 may indicate, in a syntax structure associated with the current video block, a value that indicates to the video decoder 300 that the current video block has the same motion information as the another video block.

[0090] In another example, the motion estimation unit 204 may identify, in a syntax structure associated with the current video block, another video block and a motion vector difference (MVD). The motion vector difference indicates a difference between the motion vector of the current video block and the motion vector of the indicated video block. The video decoder 300 may use the motion vector of the indicated video block and the motion vector difference to determine the motion vector of the current video block.

[0091] As discussed above, video encoder 200 may predictively signal the motion vector. Two examples of predictive signaling techniques that may be implemented by video encoder 200 include advanced motion vector predication (AMVP) and merge mode

signaling.

[0092] The intra prediction unit 206 may perform intra prediction on the current video block. When the intra prediction unit 206 performs intra prediction on the current video block, the intra prediction unit 206 may generate prediction data for the current video block based on decoded samples of other video blocks in the same picture. The prediction data for the current video block may include a predicted video block and various syntax elements.

[0093] The residual generation unit 207 may generate residual data for the current video block by subtracting (e.g., indicated by the minus sign) the predicted video block (s) of the current video block from the current video block. The residual data of the current video block may include residual video blocks that correspond to different sample components of the samples in the current video block.

[0094] In other examples, there may be no residual data for the current video block for the current video block, for example in a skip mode, and the residual generation unit 207 may not perform the subtracting operation.

[0095] The transform processing unit 208 may generate one or more transform coefficient video blocks for the current video block by applying one or more transforms to a residual video block associated with the current video block.

[0096] After the transform processing unit 208 generates a transform coefficient video block associated with the current video block, the quantization unit 209 may quantize the transform coefficient video block associated with the current video block based on one or more quantization parameter (QP) values associated with the current video block.

[0097] The inverse quantization unit 210 and the inverse transform unit 211 may apply inverse quantization and inverse transforms to the transform coefficient video block, respectively, to reconstruct a residual video block from the transform coefficient video block. The reconstruction unit 212 may add the reconstructed residual video block to

corresponding samples from one or more predicted video blocks generated by the predication unit 202 to produce a reconstructed video block associated with the current video block for storage in the buffer 213.

[0098] After the reconstruction unit 212 reconstructs the video block, loop filtering operation may be performed to reduce video blocking artifacts in the video block.

[0099] The entropy encoding unit 214 may receive data from other functional components of the video encoder 200. When the entropy encoding unit 214 receives the data, the entropy encoding unit 214 may perform one or more entropy encoding operations to generate entropy encoded data and output a bitstream that includes the entropy encoded data.

[0100] Fig. 3 is a block diagram illustrating an example of a video decoder 300, which may be an example of the video decoder 124 in the system 100 illustrated in Fig. 1, in accordance with some embodiments of the present disclosure.

[0101] The video decoder 300 may be configured to perform any or all of the techniques of this disclosure. In the example of Fig. 3, the video decoder 300 includes a plurality of functional components. The techniques described in this disclosure may be shared among the various components of the video decoder 300. In some examples, a processor may be configured to perform any or all of the techniques described in this disclosure.

[0102] In the example of Fig. 3, the video decoder 300 includes an entropy decoding unit 301, a motion compensation unit 302, an intra prediction unit 303, an inverse quantization unit 304, an inverse transformation unit 305, and a reconstruction unit 306 and a buffer 307. The video decoder 300 may, in some examples, perform a decoding pass generally reciprocal to the encoding pass described with respect to video encoder 200.

[0103] The entropy decoding unit 301 may retrieve an encoded bitstream. The encoded bitstream may include entropy coded video data (e.g., encoded blocks of video data). The entropy decoding unit 301 may decode the entropy coded video data, and from the

entropy decoded video data, the motion compensation unit 302 may determine motion information including motion vectors, motion vector precision, reference picture list indexes, and other motion information. The motion compensation unit 302 may, for example, determine such information by performing the AMVP and merge mode. AMVP is used, including derivation of several most probable candidates based on data from adjacent PBs and the reference picture. Motion information typically includes the horizontal and vertical motion vector displacement values, one or two reference picture indices, and, in the case of prediction regions in B slices, an identification of which reference picture list is associated with each index. As used herein, in some aspects, a “merge mode” may refer to deriving the motion information from spatially or temporally neighboring blocks.

[0104] The motion compensation unit 302 may produce motion compensated blocks, possibly performing interpolation based on interpolation filters. Identifiers for interpolation filters to be used with sub-pixel precision may be included in the syntax elements.

[0105] The motion compensation unit 302 may use the interpolation filters as used by the video encoder 200 during encoding of the video block to calculate interpolated values for sub-integer pixels of a reference block. The motion compensation unit 302 may determine the interpolation filters used by the video encoder 200 according to the received syntax information and use the interpolation filters to produce predictive blocks.

[0106] The motion compensation unit 302 may use at least part of the syntax information to determine sizes of blocks used to encode frame(s) and/or slice(s) of the encoded video sequence, partition information that describes how each macroblock of a picture of the encoded video sequence is partitioned, modes indicating how each partition is encoded, one or more reference frames (and reference frame lists) for each inter-encoded block, and other information to decode the encoded video sequence. As used herein, in some aspects, a “slice” may refer to a data structure that can be decoded independently from

other slices of the same picture, in terms of entropy coding, signal prediction, and residual signal reconstruction. A slice can either be an entire picture or a region of a picture.

[0107] The intra prediction unit 303 may use intra prediction modes for example received in the bitstream to form a prediction block from spatially adjacent blocks. The inverse quantization unit 304 inverse quantizes, i.e., de-quantizes, the quantized video block coefficients provided in the bitstream and decoded by entropy decoding unit 301. The inverse transform unit 305 applies an inverse transform.

[0108] The reconstruction unit 306 may obtain the decoded blocks, e.g., by summing the residual blocks with the corresponding prediction blocks generated by the motion compensation unit 302 or intra-prediction unit 303. If desired, a deblocking filter may also be applied to filter the decoded blocks in order to remove blockiness artifacts. The decoded video blocks are then stored in the buffer 307, which provides reference blocks for subsequent motion compensation/intra prediction and also produces decoded video for presentation on a display device.

[0109] Some exemplary embodiments of the present disclosure will be described in detailed hereinafter. It should be understood that section headings are used in the present document to facilitate ease of understanding and do not limit the embodiments disclosed in a section to only that section. Furthermore, while certain embodiments are described with reference to Versatile Video Coding or other specific video codecs, the disclosed techniques are applicable to other video coding technologies also. Furthermore, while some embodiments describe video coding steps in detail, it will be understood that corresponding steps decoding that undo the coding will be implemented by a decoder. Furthermore, the term video processing encompasses video coding or compression, video decoding or decompression and video transcoding in which video pixels are represented from one compressed format into another compressed format or at a different compressed bitrate.

1. Brief Summary

[0110] The present disclosure is related to video coding technologies. Specifically, it is related to intra block copy (IBC) with affine model and how to apply IBC with affine model to IBC coded blocks, and other coding tools in image/video coding. It may be applied to the existing video coding standard like HEVC, or Versatile Video Coding (VVC). It may be also applicable to future video coding standards or video codec.

2. Introduction

[0111] Video coding standards have evolved primarily through the development of the well-known ITU-T and ISO/IEC standards. The ITU-T produced H.261 and H.263, ISO/IEC produced MPEG-1 and MPEG-4 Visual, and the two organizations jointly produced the H.262/MPEG-2 Video and H.264/MPEG-4 Advanced Video Coding (AVC) and H.265/HEVC standards. Since H.262, the video coding standards are based on the hybrid video coding structure wherein temporal prediction plus transform coding are utilized. To explore the future video coding technologies beyond HEVC, Joint Video Exploration Team (JVET) was founded by VCEG and MPEG jointly in 2015. Since then, many new methods have been adopted by JVET and put into the reference software named Joint Exploration Model (JEM). In April 2018, the Joint Video Expert Team (JVET) between VCEG (Q6/16) and ISO/IEC JTC1 SC29/WG11 (MPEG) was created to work on the VVC standard targeting at 50% bitrate reduction compared to HEVC. ITU-T VCEG (Q6/16) and ISO/IEC MPEG (JTC 1/SC 29/WG 5) are studying the potential need for standardization of future video coding technology with a compression capability that significantly exceeds that of the current VVC standard. Such future standardization action could either take the form of additional extension(s) of VVC or an entirely new standard. The groups are working together on this exploration activity in a joint collaboration effort known as the Joint Video Exploration Team (JVET) to evaluate compression technology designs proposed by their experts in this area. New coding features and encoding methods implemented in Enhanced Compression Model (ECM) software that are under coordinated exploration study by the Joint Video Exploration Team (JVET) of ITU-T VCEG and

ISO/IEC MPEG as potential enhanced video coding technology beyond the capabilities of VVC.

2.1. Coding flow of a typical video codec

[0112] Fig. 4 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 signalling 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.

2.2. Intra mode coding with 67 intra prediction modes

[0113] Fig. 5 shows 67 intra prediction modes. To capture the arbitrary edge directions presented in natural video, the number of directional intra modes is extended from 33, as used in HEVC, to 65, as shown in Fig. 5, and the planar and DC modes remain the same. These denser directional intra prediction modes apply for all block sizes and for both luma and chroma intra predictions.

[0114] In the HEVC, every intra-coded block has a square shape and the length of each of its side is a power of 2. Thus, no division operations are required to generate an intra-predictor using DC mode. In VVC, blocks can have a rectangular shape that necessitates the use of a division operation per block in the general case. To avoid division operations for DC prediction, only the longer side is used to compute the average for non-square blocks.

2.2.1. Wide angle intra prediction

[0115] Although 67 modes are defined in the VVC, the exact prediction direction for a given intra prediction mode index is further dependent on the block shape. Conventional

angular intra prediction directions are defined from 45 degrees to -135 degrees in clockwise direction. In VVC, several conventional angular intra prediction modes are adaptively replaced with wide-angle intra prediction modes for non-square blocks. The replaced modes are signalled using the original mode indexes, which are remapped to the indexes of wide angular modes after parsing. The total number of intra prediction modes is unchanged, i.e., 67, and the intra mode coding method is unchanged.

[0116] To support these prediction directions, the top reference with length $2W+1$, and the left reference with length $2H+1$, are defined as shown in Fig. 6. Fig. 6 shows reference samples for wide-angular intra prediction.

[0117] The number of replaced modes in wide-angular direction mode depends on the aspect ratio of a block. The replaced intra prediction modes are illustrated in Table 2-1.

Table 2-1 – Intra prediction modes replaced by wide-angular modes

Aspect ratio	Replaced intra prediction modes
$W / H == 16$	Modes 2,3,4,5,6,7,8,9,10,11,12, 13,14,15
$W / H == 8$	Modes 2,3,4,5,6,7,8,9,10,11,12, 13
$W / H == 4$	Modes 2,3,4,5,6,7,8,9,10,11
$W / H == 2$	Modes 2,3,4,5,6,7,8,9
$W / H == 1$	None
$W / H == 1/2$	Modes 59,60,61,62,63,64,65,66
$W / H == 1/4$	Mode 57,58,59,60,61,62,63,64,65,66
$W / H == 1/8$	Modes 55, 56,57,58,59,60,61,62,63,64,65,66
$W / H == 1/16$	Modes 53, 54, 55, 56,57,58,59,60,61,62,63,64,65,66

[0118] Fig. 7 shows problem of discontinuity in case of directions beyond 45° . As shown in Fig. 7, two vertically adjacent predicted samples may use two non-adjacent reference samples in the case of wide-angle intra prediction. Hence, low-pass reference

samples filter and side smoothing are applied to the wide-angle prediction to reduce the negative effect of the increased gap Δp_a . If a wide-angle mode represents a non-fractional offset. There are 8 modes in the wide-angle modes satisfy this condition, which are $[-14, -12, -10, -6, 72, 76, 78, 80]$. When a block is predicted by these modes, the samples in the reference buffer are directly copied without applying any interpolation. With this modification, the number of samples needed to be smoothing is reduced. Besides, it aligns the design of non-fractional modes in the conventional prediction modes and wide-angle modes.

[0119] In VVC, 4:2:2 and 4:4:4 chroma formats are supported as well as 4:2:0. Chroma derived mode (DM) derivation table for 4:2:2 chroma format was initially ported from HEVC extending the number of entries from 35 to 67 to align with the extension of intra prediction modes. Since HEVC specification does not support prediction angle below -135 degree and above 45 degree, luma intra prediction modes ranging from 2 to 5 are mapped to 2. Therefore, chroma DM derivation table for 4:2:2: chroma format is updated by replacing some values of the entries of the mapping table to convert prediction angle more precisely for chroma blocks.

2.3. Inter prediction

[0120] For each inter-predicted CU, motion parameters consisting of motion vectors, reference picture indices and reference picture list usage index, and additional information needed for the new coding feature of VVC to be used for inter-predicted sample generation. The motion parameter can be signalled in an explicit or implicit manner. When a CU is coded with skip mode, the CU is associated with one PU and has no significant residual coefficients, no coded motion vector delta or reference picture index. A merge mode is specified whereby the motion parameters for the current CU are obtained from neighbouring CUs, including spatial and temporal candidates, and additional schedules introduced in VVC. The merge mode can be applied to any inter-predicted CU, not only for skip mode. The alternative to merge mode is the explicit transmission of motion parameters, where motion vector, corresponding reference picture index for each

reference picture list and reference picture list usage flag and other needed information are signalled explicitly per each CU.

2.4. Intra block copy (IBC)

[0121] Intra block copy (IBC) is a tool adopted in HEVC extensions on SCC. It is well known that it significantly improves the coding efficiency of screen content materials. Since IBC mode is implemented as a block level coding mode, block matching (BM) is performed at the encoder to find the optimal block vector (or motion vector) for each CU. Here, a block vector is used to indicate the displacement from the current block to a reference block, which is already reconstructed inside the current picture. The luma block vector of an IBC-coded CU is in integer precision. The chroma block vector rounds to integer precision as well. When combined with AMVR, the IBC mode can switch between 1-pel and 4-pel motion vector precisions. An IBC-coded CU is treated as the third prediction mode other than intra or inter prediction modes. The IBC mode is applicable to the CUs with both width and height smaller than or equal to 64 luma samples.

[0122] At the encoder side, hash-based motion estimation is performed for IBC. The encoder performs RD check for blocks with either width or height no larger than 16 luma samples. For non-merge mode, the block vector search is performed using hash-based search first. If hash search does not return valid candidate, block matching based local search will be performed.

[0123] In the hash-based search, hash key matching (32-bit CRC) between the current block and a reference block is extended to all allowed block sizes. The hash key calculation for every position in the current picture is based on 4×4 sub-blocks. For the current block of a larger size, a hash key is determined to match that of the reference block when all the hash keys of all 4×4 sub-blocks match the hash keys in the corresponding reference locations. If hash keys of multiple reference blocks are found to match that of the current block, the block vector costs of each matched reference are calculated and the one with the minimum cost is selected.

[0124] In block matching search, the search range is set to cover both the previous and current CTUs.

[0125] At CU level, IBC mode is signalled with a flag and it can be signalled as IBC AMVP mode or IBC skip/merge mode as follows:

- IBC skip/merge mode: a merge candidate index is used to indicate which of the block vectors in the list from neighbouring candidate IBC coded blocks is used to predict the current block. The merge list consists of spatial, HMVP, and pairwise candidates.
- IBC AMVP mode: block vector difference is coded in the same way as a motion vector difference. The block vector prediction method uses two candidates as predictors, one from left neighbour and one from above neighbour (if IBC coded). When either neighbour is not available, a default block vector will be used as a predictor. A flag is signalled to indicate the block vector predictor index.

2.5. IBC Motion Candidates

[0126] The term ‘block’ may represent a coding tree block (CTB), a coding tree unit (CTU), a coding block (CB), a CU, a PU, a TU, a PB, a TB or a video processing unit comprising multiple samples/pixels. A block may be rectangular or non-rectangular.

[0127] For an IBC coded block, a block vector (BV) is used to indicate the displacement from the current block to a reference block, which is already reconstructed inside the current picture.

[0128] W and H are the width and height of current block (e.g., luma block).

[0129] The non-adjacent spatial candidates of current coding block are adjacent spatial candidates of a virtual block in the i th search round (as shown in Fig. 9). The width and height of the virtual block for the i th search round are calculated by: $\text{newWidth} = i \times 2 \times \text{gridX} + W$, $\text{newHeight} = i \times 2 \times \text{gridY} + H$. Obviously, the virtual block is the current block if the search round i is 0.

[0130] In the following, a BV predictor also is a BV candidate. The skip mode also is the

merge mode.

[0131] The BV candidates can be divided into several groups according to some criteria. Each group is called a subgroup. For example, we can take adjacent spatial and temporal BV candidates as a first subgroup and take the remaining BV candidates as a second subgroup; In another example, we can also take the first N ($N \geq 2$) BV candidates as a first subgroup, take the following M ($M \geq 2$) BV candidates as a second subgroup, and take the remaining BV candidates as a third subgroup.

2.6. Merge mode with MVD (MMVD)

[0132] In addition to merge mode, where the implicitly derived motion information is directly used for prediction samples generation of the current CU, the merge mode with motion vector differences (MMVD) is introduced in VVC. A MMVD flag is signalled right after sending a regular merge flag to specify whether MMVD mode is used for a CU.

[0133] In MMVD, after a merge candidate is selected, it is further refined by the signalled MVDs information. The further information includes a merge candidate flag, an index to specify motion magnitude, and an index for indication of motion direction. In MMVD mode, one for the first two candidates in the merge list is selected to be used as MV basis. The MMVD candidate flag is signalled to specify which one is used between the first and second merge candidates.

[0134] Distance index specifies motion magnitude information and indicate the pre-defined offset from the starting point. Fig. 8 shows MMVD search point. As shown in Fig. 8, an offset is added to either horizontal component or vertical component of starting MV. The relation of distance index and pre-defined offset is specified in Table 2-2.

Table 2-2 – The relation of distance index and pre-defined offset

Distance IDX	0	1	2	3	4	5	6	7
Offset (in unit of luma sample)	1/4	1/2	1	2	4	8	16	32

[0135] Direction index represents the direction of the MVD relative to the starting point.

The direction index can represent of the four directions as shown in Table 2-3. It's noted that the meaning of MVD sign could be variant according to the information of starting MVs. When the starting MVs is an un-prediction MV or bi-prediction MVs with both lists point to the same side of the current picture (i.e. POCs of two references are both larger than the POC of the current picture, or are both smaller than the POC of the current picture), the sign in Table 2-3 specifies the sign of MV offset added to the starting MV. When the starting MVs is bi-prediction MVs with the two MVs point to the different sides of the current picture (i.e. the POC of one reference is larger than the POC of the current picture, and the POC of the other reference is smaller than the POC of the current picture), and the difference of POC in list 0 is greater than the one in list 1, the sign in Table 2-3 specifies the sign of MV offset added to the list0 MV component of starting MV and the sign for the list1 MV has opposite value. Otherwise, if the difference of POC in list 1 is greater than list 0, the sign in Table 2-3 specifies the sign of MV offset added to the list1 MV component of starting MV and the sign for the list0 MV has opposite value.

[0136] The MVD is scaled according to the difference of POCs in each direction. If the differences of POCs in both lists are the same, no scaling is needed. Otherwise, if the difference of POC in list 0 is larger than the one of list 1, the MVD for list 1 is scaled, by defining the POC difference of L0 as td and POC difference of L1 as tb , described in Figure 23. If the POC difference of L1 is greater than L0, the MVD for list 0 is scaled in the same way. If the starting MV is uni-predicted, the MVD is added to the available MV.

Table 2-3 – Sign of MV offset specified by direction index

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

2.7. Symmetric MVD coding

[0137] In VVC, besides the normal unidirectional prediction and bi-directional prediction mode MVD signalling, symmetric MVD mode for bi-predictional MVD signalling is applied. In the symmetric MVD mode, motion information including

reference picture indices of both list-0 and list-1 and MVD of list-1 are not signaled but derived.

[0138] The decoding process of the symmetric MVD mode is as follows:

- 1) At slice level, variables BiDirPredFlag, RefIdxSymL0 and RefIdxSymL1 are derived as follows:
 - If mvd_l1_zero_flag is 1, BiDirPredFlag is set equal to 0.
 - Otherwise, if the nearest reference picture in list-0 and the nearest reference picture in list-1 form a forward and backward pair of reference pictures or a backward and forward pair of reference pictures, BiDirPredFlag is set to 1, and both list-0 and list-1 reference pictures are short-term reference pictures. Otherwise BiDirPredFlag is set to 0.
- 2) At CU level, a symmetrical mode flag indicating whether symmetrical mode is used or not is explicitly signaled if the CU is bi-prediction coded and BiDirPredFlag is equal to 1.

[0139] When the symmetrical mode flag is true, only mvp_l0_flag, mvp_l1_flag and MVD0 are explicitly signaled. The reference indices for list-0 and list-1 are set equal to the pair of reference pictures, respectively. MVD1 is set equal to (– MVD0). The final motion vectors are shown in below formula.

$$\begin{cases} (mvx_0, mvy_0) = (mvp_x_0 + mvd_x_0, mvp_y_0 + mvd_y_0) \\ (mvx_1, mvy_1) = (mvp_x_1 - mvd_x_0, mvp_y_1 - mvd_y_0) \end{cases} \quad (2-1)$$

[0140] In the encoder, symmetric MVD motion estimation starts with initial MV evaluation. A set of initial MV candidates comprising of the MV obtained from uni-prediction search, the MV obtained from bi-prediction search and the MVs from the AMVP list. The one with the lowest rate-distortion cost is chosen to be the initial MV for the symmetric MVD motion search.

2.8. Bi-directional optical flow (BDOF)

[0141] The bi-directional optical flow (BDOF) tool is included in VVC. BDOF, previously referred to as BIO, was included in the JEM. Compared to the JEM version, the BDOF in VVC is a simpler version that requires much less computation, especially in terms of number of multiplications and the size of the multiplier.

[0142] BDOF is used to refine the bi-prediction signal of a CU at the 4×4 subblock level.

BDOF is applied to a CU if it satisfies all the following conditions:

- The CU is coded using “true” bi-prediction mode, i.e., one of the two reference pictures is prior to the current picture in display order and the other is after the current picture in display order;
- The distances (i.e. POC difference) from two reference pictures to the current picture are same;
- Both reference pictures are short-term reference pictures;
- The CU is not coded using affine mode or the SbTMVP merge mode;
- CU has more than 64 luma samples;
- Both CU height and CU width are larger than or equal to 8 luma samples;
- BCW weight index indicates equal weight;
- WP is not enabled for the current CU;
- CIIP mode is not used for the current CU.

[0143] BDOF is only applied to the luma component. As its name indicates, the BDOF mode is based on the optical flow concept, which assumes that the motion of an object is smooth. For each 4×4 subblock, a motion refinement (v_x, v_y) is calculated by minimizing the difference between the L0 and L1 prediction samples. The motion refinement is then used to adjust the bi-predicted sample values in the 4x4 subblock. The following steps are applied in the BDOF process.

[0144] First, the horizontal and vertical gradients, $\frac{\partial I^{(k)}}{\partial x}(i, j)$ and $\frac{\partial I^{(k)}}{\partial y}(i, j)$, $k = 0, 1$, of the two prediction signals are computed by directly calculating the difference between two neighboring samples, i.e.,

$$\begin{aligned}\frac{\partial I^{(k)}}{\partial x}(i, j) &= ((I^{(k)}(i+1, j) \gg \text{shift1}) - (I^{(k)}(i-1, j) \gg \text{shift1})) \\ \frac{\partial I^{(k)}}{\partial y}(i, j) &= ((I^{(k)}(i, j+1) \gg \text{shift1}) - (I^{(k)}(i, j-1) \gg \text{shift1}))\end{aligned}\tag{2-2}$$

where $I^{(k)}(i, j)$ are the sample value at coordinate (i, j) of the prediction signal in list k , $k = 0, 1$, and shift1 is calculated based on the luma bit depth, bitDepth, as $\text{shift1} = \max(6, \text{bitDepth}-6)$.

[0145] Then, the auto- and cross-correlation of the gradients, S_1 , S_2 , S_3 , S_5 and S_6 ,

are calculated as

$$\begin{aligned} S_1 &= \sum_{(i,j) \in \Omega} Abs(\psi_x(i,j)), \quad S_3 = \sum_{(i,j) \in \Omega} \theta(i,j) \cdot Sign(\psi_x(i,j)) \\ S_2 &= \sum_{(i,j) \in \Omega} \psi_x(i,j) \cdot Sign(\psi_y(i,j)) \end{aligned} \quad (2-3)$$

$$S_5 = \sum_{(i,j) \in \Omega} Abs(\psi_y(i,j)), \quad S_6 = \sum_{(i,j) \in \Omega} \theta(i,j) \cdot Sign(\psi_y(i,j))$$

where

$$\begin{aligned} \psi_x(i,j) &= \left(\frac{\partial I^{(1)}}{\partial x}(i,j) + \frac{\partial I^{(0)}}{\partial x}(i,j) \right) \gg n_a \\ \psi_y(i,j) &= \left(\frac{\partial I^{(1)}}{\partial y}(i,j) + \frac{\partial I^{(0)}}{\partial y}(i,j) \right) \gg n_a \\ \theta(i,j) &= (I^{(1)}(i,j) \gg n_b) - (I^{(0)}(i,j) \gg n_b) \end{aligned} \quad (2-4)$$

where Ω is a 6×6 window around the 4×4 subblock, and the values of n_a and n_b are set equal to $\min(1, \text{bitDepth} - 11)$ and $\min(4, \text{bitDepth} - 8)$, respectively.

[0146] The motion refinement (v_x, v_y) is then derived using the cross- and auto-correlation terms using the following:

$$\begin{aligned} v_x &= S_1 > 0 ? \text{clip3} \left(-th'_{BIO}, th'_{BIO}, -((S_3 \cdot 2^{n_b - n_a}) \gg \lfloor \log_2 S_1 \rfloor) \right) : 0 \\ v_y &= S_5 > 0 ? \text{clip3} \left(-th'_{BIO}, th'_{BIO}, - \left((S_6 \cdot 2^{n_b - n_a} \right. \right. \\ &\quad \left. \left. - ((v_x S_{2,m}) \ll n_{S_2} + v_x S_{2,s}) / 2 \right) \gg \lfloor \log_2 S_5 \rfloor \right) : 0 \end{aligned} \quad (2-5)$$

where $S_{2,m} = S_2 \gg n_{S_2}$, $S_{2,s} = S_2 \& (2^{n_{S_2}} - 1)$, $th'_{BIO} = 2^{\max(5, BD-7)}$. $\lfloor \cdot \rfloor$ is the floor function, and $n_{S_2} = 12$.

[0147] Based on the motion refinement and the gradients, the following adjustment is calculated for each sample in the 4×4 subblock:

$$\begin{aligned} b(x,y) &= \text{rnd} \left(\left(v_x \left(\frac{\partial I^{(1)}(x,y)}{\partial x} - \frac{\partial I^{(0)}(x,y)}{\partial x} \right) + v_y \left(\frac{\partial I^{(1)}(x,y)}{\partial y} - \frac{\partial I^{(0)}(x,y)}{\partial y} \right) \right. \right. \\ &\quad \left. \left. + 1 \right) / 2 \right) \end{aligned} \quad (2-6)$$

[0148] Finally, the BDOF samples of the CU are calculated by adjusting the bi-prediction

samples as follows:

$$pred_{BDOF}(x, y) = (I^{(0)}(x, y) + I^{(1)}(x, y) + b(x, y) + o_{offset}) \gg shift \quad (2-7)$$

[0149] These values are selected such that the multipliers in the BDOF process do not exceed 15-bit, and the maximum bit-width of the intermediate parameters in the BDOF process is kept within 32-bit.

[0150] In order to derive the gradient values, some prediction samples $I^{(k)}(i, j)$ in list k ($k = 0, 1$) outside of the current CU boundaries need to be generated. Fig. 10 shows extended CU region used in BDOF. As depicted in Fig. 10, the BDOF in VVC uses one extended row/column around the CU's boundaries. In order to control the computational complexity of generating the out-of-boundary prediction samples, prediction samples in the extended area (white positions) are generated by taking the reference samples at the nearby integer positions (using floor() operation on the coordinates) directly without interpolation, and the normal 8-tap motion compensation interpolation filter is used to generate prediction samples within the CU (gray positions). These extended sample values are used in gradient calculation only. For the remaining steps in the BDOF process, if any sample and gradient values outside of the CU boundaries are needed, they are padded (i.e. repeated) from their nearest neighbors.

[0151] When the width and/or height of a CU are larger than 16 luma samples, it will be split into subblocks with width and/or height equal to 16 luma samples, and the subblock boundaries are treated as the CU boundaries in the BDOF process. The maximum unit size for BDOF process is limited to 16x16. For each subblock, the BDOF process could be skipped. When the SAD of between the initial L0 and L1 prediction samples is smaller than a threshold, the BDOF process is not applied to the subblock. The threshold is set equal to $(8 * W * (H \gg 1))$, where W indicates the subblock width, and H indicates subblock height. To avoid the additional complexity of SAD calculation, the SAD between the initial L0 and L1 prediction samples calculated in DVMR process is re-used here.

[0152] If BCW is enabled for the current block, i.e., the BCW weight index indicates

unequal weight, then bi-directional optical flow is disabled. Similarly, if WP is enabled for the current block, i.e., the luma_weight_lx_flag is 1 for either of the two reference pictures, then BDOF is also disabled. When a CU is coded with symmetric MVD mode or CIIP mode, BDOF is also disabled.

2.9. Combined inter and intra prediction (CIIP)

2.10. Affine motion compensated prediction

[0153] In HEVC, only translation motion model is applied for motion compensation prediction (MCP). While in the real world, there are many kinds of motion, e.g., zoom in/out, rotation, perspective motions and the other irregular motions. In VVC, a block-based affine transform motion compensation prediction is applied. Fig. 11a and Fig. 11b show control point based affine motion model, wherein Fig. 11a shows 4-parameter affine model and Fig. 11b shows 6-parameter affine model. As shown Fig. 11a and Fig. 11b, the affine motion field of the block is described by motion information of two control point (4-parameter) or three control point motion vectors (6-parameter).

[0154] For 4-parameter affine motion model, motion vector at sample location (x, y) in a block is derived as:

$$\begin{cases} mv_x = \frac{mv_{1x} - mv_{0x}}{W}x + \frac{mv_{1y} - mv_{0y}}{W}y + mv_{0x} \\ mv_y = \frac{mv_{1y} - mv_{0y}}{W}x + \frac{mv_{1x} - mv_{0x}}{W}y + mv_{0y} \end{cases} \quad (2-8)$$

[0155] For 6-parameter affine motion model, motion vector at sample location (x, y) in a block is derived as:

$$\begin{cases} mv_x = \frac{mv_{1x} - mv_{0x}}{W}x + \frac{mv_{2x} - mv_{0x}}{H}y + mv_{0x} \\ mv_y = \frac{mv_{1y} - mv_{0y}}{W}x + \frac{mv_{2y} - mv_{0y}}{H}y + mv_{0y} \end{cases} \quad (2-9)$$

where (mv_{0x}, mv_{0y}) is motion vector of the top-left corner control point, (mv_{1x}, mv_{1y}) is motion vector of the top-right corner control point, and (mv_{2x}, mv_{2y}) is motion vector of the bottom-left corner control point.

[0156] Fig. 12 shows Affine MVF per subblock. In order to simplify the motion

compensation prediction, block based affine transform prediction is applied. To derive motion vector of each 4×4 luma subblock, the motion vector of the center sample of each subblock, as shown in Fig. 12, is calculated according to above equations, and rounded to $1/16$ fraction accuracy. Then the motion compensation interpolation filters are applied to generate the prediction of each subblock with derived motion vector. The subblock size of chroma-components is also set to be 4×4 . The MV of a 4×4 chroma subblock is calculated as the average of the MVs of the four corresponding 4×4 luma subblocks.

[0157] As done for translational motion inter prediction, there are also two affine motion inter prediction modes: affine merge mode and affine AMVP mode.

2.10.1. Affine merge prediction

[0158] AF_MERGE mode can be applied for CUs with both width and height larger than or equal to 8. In this mode the CPMVs of the current CU is generated based on the motion information of the spatial neighbouring CUs. . There can be up to five CPMVP candidates and an index is signalled to indicate the one to be used for the current CU. The following three types of CPVM candidate are used to form the affine merge candidate list:

- Inherited affine merge candidates that extrapolated from the CPMVs of the neighbour CUs.
- Constructed affine merge candidates CPMVPs that are derived using the translational MVs of the neighbour CUs.
- Zero MVs.

[0159] In VVC, there are maximum two inherited affine candidates, which are derived from affine motion model of the neighbouring blocks, one from left neighbouring CUs and one from above neighbouring CUs. Fig. 13 shows locations of inherited affine motion predictors. The candidate blocks are shown in Fig. 13. For the left predictor, the scan order is $A0 \rightarrow A1$, and for the above predictor, the scan order is $B0 \rightarrow B1 \rightarrow B2$. Only the first inherited candidate from each side is selected. No pruning check is performed between two inherited candidates. When a neighbouring affine CU is identified, its control point motion vectors are used to derive the CPMVP candidate in the affine merge list of the current CU. As shown in , if the neighbour left bottom block A is coded in affine mode,

the motion vectors v_2 , v_3 and v_4 of the top left corner, above right corner and left bottom corner of the CU which contains the block A are attained. When block A is coded with 4-parameter affine model, the two CPMVs of the current CU are calculated according to v_2 , and v_3 . In case that block A is coded with 6-parameter affine model, the three CPMVs of the current CU are calculated according to v_2 , v_3 and v_4 . Fig. 14 shows control point motion vector inheritance.

[0160] Constructed affine candidate means the candidate is constructed by combining the neighbour translational motion information of each control point. Fig. 15 shows locations of Candidates position for constructed affine merge mode. The motion information for the control points is derived from the specified spatial neighbours and temporal neighbour shown in Fig. 15. $CPMV_k$ ($k=1, 2, 3, 4$) represents the k -th control point. For $CPMV_1$, the B2->B3->A2 blocks are checked and the MV of the first available block is used. For $CPMV_2$, the B1->B0 blocks are checked and for $CPMV_3$, the A1->A0 blocks are checked. For TMVP is used as $CPMV_4$ if it's available.

[0161] After MVs of four control points are attained, affine merge candidates are constructed based on that motion information. The following combinations of control point MVs are used to construct in order:

$\{CPMV_1, CPMV_2, CPMV_3\}$, $\{CPMV_1, CPMV_2, CPMV_4\}$, $\{CPMV_1, CPMV_3, CPMV_4\}$, $\{CPMV_2, CPMV_3, CPMV_4\}$, $\{CPMV_1, CPMV_2\}$, $\{CPMV_1, CPMV_3\}$.

[0162] The combination of 3 CPMVs constructs a 6-parameter affine merge candidate and the combination of 2 CPMVs constructs a 4-parameter affine merge candidate. To avoid motion scaling process, if the reference indices of control points are different, the related combination of control point MVs is discarded.

[0163] After inherited affine merge candidates and constructed affine merge candidate are checked, if the list is still not full, zero MVs are inserted to the end of the list.

2.10.2. Affine AMVP prediction

[0164] Affine AMVP mode can be applied for CUs with both width and height larger than or equal to 16. An affine flag in CU level is signalled in the bitstream to indicate whether affine AMVP mode is used and then another flag is signalled to indicate whether 4-parameter affine or 6-parameter affine. In this mode, the difference of the CPMVs of current CU and their predictors CPMVPs is signalled in the bitstream. The affine AMVP candidate list size is 2 and it is generated by using the following four types of CPVM candidate in order:

- Inherited affine AMVP candidates that extrapolated from the CPMVs of the neighbour CUs.
- Constructed affine AMVP candidates CPMVPs that are derived using the translational MVs of the neighbour CUs.
- Translational MVs from neighbouring CUs.
- Zero MVs.

[0165] The checking order of inherited affine AMVP candidates is same to the checking order of inherited affine merge candidates. The only difference is that, for AMVP candidate, only the affine CU that has the same reference picture as in current block is considered. No pruning process is applied when inserting an inherited affine motion predictor into the candidate list.

[0166] Constructed AMVP candidate is derived from the specified spatial neighbours shown in Fig. 15. The same checking order is used as done in affine merge candidate construction. In addition, reference picture index of the neighbouring block is also checked. The first block in the checking order that is inter coded and has the same reference picture as in current CUs is used. There is only one When the current CU is coded with 4-parameter affine mode, and mv_0 and mv_1 are both available, they are added as one candidate in the affine AMVP list. When the current CU is coded with 6-parameter affine mode, and all three CPMVs are available, they are added as one candidate in the affine AMVP list. Otherwise, constructed AMVP candidate is set as unavailable.

[0167] If affine AMVP list candidates is still less than 2 after inherited affine AMVP

candidates and Constructed AMVP candidate are checked, mv_0 , mv_1 , and mv_2 will be added, in order, as the translational MVs to predict all control point MVs of the current CU, when available. Finally, zero MVs are used to fill the affine AMVP list if it is still not full.

2.10.3. Affine motion information storage

[0168] In VVC, the CPMVs of affine CUs are stored in a separate buffer. The stored CPMVs are only used to generate the inherited CPMVPs in affine merge mode and affine AMVP mode for the lately coded CUs. The subblock MVs derived from CPMVs are used for motion compensation, MV derivation of merge/AMVP list of translational MVs and de-blocking.

[0169] To avoid the picture line buffer for the additional CPMVs, affine motion data inheritance from the CUs from above CTU is treated differently to the inheritance from the normal neighbouring CUs. If the candidate CU for affine motion data inheritance is in the above CTU line, the bottom-left and bottom-right subblock MVs in the line buffer instead of the CPMVs are used for the affine MVP derivation. In this way, the CPMVs are only stored in local buffer. If the candidate CU is 6-parameter affine coded, the affine model is degraded to 4-parameter model. Fig. 16 shows motion vector usage for proposed combined method. As shown in Fig. 16, along the top CTU boundary, the bottom-left and bottom right subblock motion vectors of a CU are used for affine inheritance of the CUs in bottom CTUs.

2.10.4. Prediction refinement with optical flow for affine mode

[0170] Subblock based affine motion compensation can save memory access bandwidth and reduce computation complexity compared to pixel-based motion compensation, at the cost of prediction accuracy penalty. To achieve a finer granularity of motion compensation, prediction refinement with optical flow (PROF) is used to refine the subblock based affine motion compensated prediction without increasing the memory access bandwidth for motion compensation. In VVC, after the subblock based affine motion compensation is

performed, luma prediction sample is refined by adding a difference derived by the optical flow equation. The PROF is described as following four steps:

Step 1) The subblock-based affine motion compensation is performed to generate subblock prediction $I(i, j)$.

Step2) The spatial gradients $g_x(i, j)$ and $g_y(i, j)$ of the subblock prediction are calculated at each sample location using a 3-tap filter $[-1, 0, 1]$. The gradient calculation is exactly the same as gradient calculation in BDOF.

$$g_x(i, j) = (I(i + 1, j) \gg \text{shift1}) - (I(i - 1, j) \gg \text{shift1}) \quad (2-10)$$

$$g_y(i, j) = (I(i, j + 1) \gg \text{shift1}) - (I(i, j - 1) \gg \text{shift1}) \quad (2-11)$$

shift1 is used to control the gradient's precision. The subblock (i.e. 4x4) prediction is extended by one sample on each side for the gradient calculation. To avoid additional memory bandwidth and additional interpolation computation, those extended samples on the extended borders are copied from the nearest integer pixel position in the reference picture.

Step 3) The luma prediction refinement is calculated by the following optical flow equation.

$$\Delta I(i, j) = g_x(i, j) * \Delta v_x(i, j) + g_y(i, j) * \Delta v_y(i, j) \quad (2-12)$$

where the $\Delta v(i, j)$ is the difference between sample MV computed for sample location (i, j) , denoted by $v(i, j)$, and the subblock MV of the subblock to which sample (i, j) belongs, as shown in Fig. 17. The $\Delta v(i, j)$ is quantized in the unit of 1/32 luam sample precision. Fig. 17 shows subblock MV VSB and pixel $\Delta v(i, j)$.

[0171] Since the affine model parameters and the sample location relative to the subblock center are not changed from subblock to subblock, $\Delta v(i, j)$ can be calculated for the first subblock, and reused for other subblocks in the same CU. Let $dx(i, j)$ and $dy(i, j)$ be the horizontal and vertical offset from the sample location (i, j) to the center of the

subblock (x_{SB}, y_{SB}) , $\Delta v(x, y)$ can be derived by the following equation,

$$\begin{cases} dx(i, j) = i - x_{SB} \\ dy(i, j) = j - y_{SB} \end{cases} \quad (2-13)$$

$$\begin{cases} \Delta v_x(i, j) = C * dx(i, j) + D * dy(i, j) \\ \Delta v_y(i, j) = E * dx(i, j) + F * dy(i, j) \end{cases} \quad (2-14)$$

[0172] In order to keep accuracy, the enter of the subblock (x_{SB}, y_{SB}) is calculated as $((W_{SB} - 1) / 2, (H_{SB} - 1) / 2)$, where W_{SB} and H_{SB} are the subblock width and height, respectively.

[0173] For 4-parameter affine model,

$$\begin{cases} C = F = \frac{v_{1x} - v_{0x}}{w} \\ E = -D = \frac{v_{1y} - v_{0y}}{w} \end{cases} \quad (2-15)$$

[0174] For 6-parameter affine model,

$$\begin{cases} C = \frac{v_{1x} - v_{0x}}{w} \\ D = \frac{v_{2x} - v_{0x}}{h} \\ E = \frac{v_{1y} - v_{0y}}{w} \\ F = \frac{v_{2y} - v_{0y}}{h} \end{cases} \quad (2-16)$$

where (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.

Step 4) Finally, the luma prediction refinement $\Delta I(i, j)$ is added to the subblock prediction $I(i, j)$. The final prediction I' is generated as the following equation.

$$I' (i, j) = I(i, j) + \Delta I(i, j) \quad (2-17)$$

[0175] PROF is not be applied in two cases for an affine coded CU: 1) all control point MVs are the same, which indicates the CU only has translational motion; 2) the affine motion parameters are greater than a specified limit because the subblock based affine MC is degraded to CU based MC to avoid large memory access bandwidth requirement.

[0176] A fast encoding method is applied to reduce the encoding complexity of affine motion estimation with PROF. PROF is not applied at affine motion estimation stage in following two situations: a) if this CU is not the root block and its parent block does not select the affine mode as its best mode, PROF is not applied since the possibility for

current CU to select the affine mode as best mode is low; b) if the magnitude of four affine parameters (C, D, E, F) are all smaller than a predefined threshold and the current picture is not a low delay picture, PROF is not applied because the improvement introduced by PROF is small for this case. In this way, the affine motion estimation with PROF can be accelerated.

2.11. Subblock-based temporal motion vector prediction (SbTMVP)

[0177] VVC supports the subblock-based temporal motion vector prediction (SbTMVP) method. Similar to the temporal motion vector prediction (TMVP) in HEVC, SbTMVP uses the motion field in the collocated picture to improve motion vector prediction and merge mode for CUs in the current picture. The same collocated picture used by TMVP is used for SbTMVP. SbTMVP differs from TMVP in the following two main aspects:

- TMVP predicts motion at CU level, but SbTMVP predicts motion at sub-CU level;
- Whereas TMVP fetches the temporal motion vectors from the collocated block in the collocated picture (the collocated block is the bottom-right or center block relative to the current CU), SbTMVP applies a motion shift before fetching the temporal motion information from the collocated picture, where the motion shift is obtained from the motion vector from one of the spatial neighboring blocks of the current CU.

[0178] The SbTMVP process is illustrated in Fig. 18. SbTMVP predicts the motion vectors of the sub-CUs within the current CU in two steps. In the first step, the spatial neighbor A1 in (a) is examined. If A1 has a motion vector that uses the collocated picture as its reference picture, this motion vector is selected to be the motion shift to be applied. If no such motion is identified, then the motion shift is set to (0, 0).

[0179] In the second step, the motion shift identified in Step 1 is applied (i.e., added to the current block's coordinates) to obtain sub-CU-level motion information (motion vectors and reference indices) from the collocated picture as shown in (b). The example in (b) assumes the motion shift is set to block A1's motion. Then, for each sub-CU, the motion information of its corresponding block (the smallest motion grid that covers the center sample) in the collocated picture is used to derive the motion information for the sub-CU. After the motion information of the collocated sub-CU is identified, it is

converted to the motion vectors and reference indices of the current sub-CU in a similar way as the TMVP process of HEVC, where temporal motion scaling is applied to align the reference pictures of the temporal motion vectors to those of the current CU.

[0180] In VVC, a combined subblock based merge list which contains both SbTMVP candidate and affine merge candidates is used for the signalling of subblock based merge mode. The SbTMVP mode is enabled/disabled by a sequence parameter set (SPS) flag. If the SbTMVP mode is enabled, the SbTMVP predictor is added as the first entry of the list of subblock based merge candidates, and followed by the affine merge candidates. The size of subblock based merge list is signalled in SPS and the maximum allowed size of the subblock based merge list is 5 in VVC.

[0181] The sub-CU size used in SbTMVP is fixed to be 8x8, and as done for affine merge mode, SbTMVP mode is only applicable to the CU with both width and height are larger than or equal to 8.

[0182] The encoding logic of the additional SbTMVP merge candidate is the same as for the other merge candidates, that is, for each CU in P or B slice, an additional RD check is performed to decide whether to use the SbTMVP candidate.

2.12. Adaptive motion vector resolution (AMVR)

[0183] In HEVC, motion vector differences (MVDs) (between the motion vector and predicted motion vector of a CU) are signalled in units of quarter-luma-sample when `use_integer_mv_flag` is equal to 0 in the slice header. In VVC, a CU-level adaptive motion vector resolution (AMVR) scheme is introduced. AMVR allows MVD of the CU to be coded in different precision. Dependent on the mode (normal AMVP mode or affine AMVP mode) for the current CU, the MVDs of the current CU can be adaptively selected as follows:

- Normal AMVP mode: quarter-luma-sample, half-luma-sample, integer-luma-sample or four-luma-sample.
- Affine AMVP mode: quarter-luma-sample, integer-luma-sample or 1/16 luma-sample.

[0184] The CU-level MVD resolution indication is conditionally signalled if the current CU has at least one non-zero MVD component. If all MVD components (that is, both horizontal and vertical MVDs for reference list L0 and reference list L1) are zero, quarter-luma-sample MVD resolution is inferred.

[0185] For a CU that has at least one non-zero MVD component, a first flag is signalled to indicate whether quarter-luma-sample MVD precision is used for the CU. If the first flag is 0, no further signaling is needed and quarter-luma-sample MVD precision is used for the current CU. Otherwise, a second flag is signalled to indicate half-luma-sample or other MVD precisions (integer or four-luma sample) is used for normal AMVP CU. In the case of half-luma-sample, a 6-tap interpolation filter instead of the default 8-tap interpolation filter is used for the half-luma sample position. Otherwise, a third flag is signalled to indicate whether integer-luma-sample or four-luma-sample MVD precision is used for normal AMVP CU. In the case of affine AMVP CU, the second flag is used to indicate whether integer-luma-sample or 1/16 luma-sample MVD precision is used. In order to ensure the reconstructed MV has the intended precision (quarter-luma-sample, half-luma-sample, integer-luma-sample or four-luma-sample), the motion vector predictors for the CU will be rounded to the same precision as that of the MVD before being added together with the MVD. The motion vector predictors are rounded toward zero (that is, a negative motion vector predictor is rounded toward positive infinity and a positive motion vector predictor is rounded toward negative infinity).

[0186] The encoder determines the motion vector resolution for the current CU using RD check. To avoid always performing CU-level RD check four times for each MVD resolution, in VTM11, the RD check of MVD precisions other than quarter-luma-sample is only invoked conditionally. For normal AVMP mode, the RD cost of quarter-luma-sample MVD precision and integer-luma sample MV precision is computed first. Then, the RD cost of integer-luma-sample MVD precision is compared to that of quarter-luma-sample MVD precision to decide whether it is necessary to further check the RD cost of four-luma-sample MVD precision. When the RD cost for quarter-luma-sample MVD

precision is much smaller than that of the integer-luma-sample MVD precision, the RD check of four-luma-sample MVD precision is skipped. Then, the check of half-luma-sample MVD precision is skipped if the RD cost of integer-luma-sample MVD precision is significantly larger than the best RD cost of previously tested MVD precisions. For affine AMVP mode, if affine inter mode is not selected after checking rate-distortion costs of affine merge/skip mode, merge/skip mode, quarter-luma-sample MVD precision normal AMVP mode and quarter-luma-sample MVD precision affine AMVP mode, then 1/16 luma-sample MV precision and 1-pel MV precision affine inter modes are not checked. Furthermore, affine parameters obtained in quarter-luma-sample MV precision affine inter mode is used as starting search point in 1/16 luma-sample and quarter-luma-sample MV precision affine inter modes.

2.13. Bi-prediction with CU-level weight (BCW)

[0187] In HEVC, the bi-prediction signal is generated by averaging two prediction signals obtained from two different reference pictures and/or using two different motion vectors. In VVC, the bi-prediction mode is extended beyond simple averaging to allow weighted averaging of the two prediction signals.

$$P_{\text{bi-pred}} = ((8 - w) * P_0 + w * P_1 + 4) \gg 3 \quad (2-18)$$

[0188] Five weights are allowed in the weighted averaging bi-prediction, $w \in \{-2, 3, 4, 5, 10\}$. For each bi-predicted CU, the weight w is determined in one of two ways:

- 1) for a non-merge CU, the weight index is signalled after the motion vector difference;
- 2) for a merge CU, the weight index is inferred from neighbouring blocks based on the merge candidate index. BCW is only applied to CUs with 256 or more luma samples (i.e., CU width times CU height is greater than or equal to 256). For low-delay pictures, all 5 weights are used. For non-low-delay pictures, only 3 weights ($w \in \{3, 4, 5\}$) are used.

- At the encoder, fast search algorithms are applied to find the weight index without significantly increasing the encoder complexity. These algorithms are summarized as follows. For further details readers are referred to the VTM software. When combined with

AMVR, unequal weights are only conditionally checked for 1-pel and 4-pel motion vector precisions if the current picture is a low-delay picture.

- When combined with affine, affine ME will be performed for unequal weights if and only if the affine mode is selected as the current best mode.
- When the two reference pictures in bi-prediction are the same, unequal weights are only conditionally checked.
- Unequal weights are not searched when certain conditions are met, depending on the POC distance between current picture and its reference pictures, the coding QP, and the temporal level.

[0189] The BCW weight index is coded using one context coded bin followed by bypass coded bins. The first context coded bin indicates if equal weight is used; and if unequal weight is used, additional bins are signalled using bypass coding to indicate which unequal weight is used.

[0190] Weighted prediction (WP) is a coding tool supported by the H.264/AVC and HEVC standards to efficiently code video content with fading. Support for WP was also added into the VVC standard. WP allows weighting parameters (weight and offset) to be signalled for each reference picture in each of the reference picture lists L0 and L1. Then, during motion compensation, the weight(s) and offset(s) of the corresponding reference picture(s) are applied. WP and BCW are designed for different types of video content. In order to avoid interactions between WP and BCW, which will complicate VVC decoder design, if a CU uses WP, then the BCW weight index is not signalled, and w is inferred to be 4 (i.e. equal weight is applied). For a merge CU, the weight index is inferred from neighbouring blocks based on the merge candidate index. This can be applied to both normal merge mode and inherited affine merge mode. For constructed affine merge mode, the affine motion information is constructed based on the motion information of up to 3 blocks. The BCW index for a CU using the constructed affine merge mode is simply set equal to the BCW index of the first control point MV.

[0191] In VVC, CIIP and BCW cannot be jointly applied for a CU. When a CU is coded with CIIP mode, the BCW index of the current CU is set to 2, e.g., equal weight.

2.14. Local illumination compensation (LIC)

[0192] Local illumination compensation (LIC) is a coding tool to address the issue of local illumination changes between current picture and its temporal reference pictures. The LIC is based on a linear model where a scaling factor and an offset are applied to the reference samples to obtain the prediction samples of a current block. Specifically, the LIC can be mathematically modeled by the following equation:

$$P(x, y) = \alpha \cdot P_r(x + v_x, y + v_y) + \beta$$

where $P(x, y)$ is the prediction signal of the current block at the coordinate (x, y) ; $P_r(x + v_x, y + v_y)$ is the reference block pointed by the motion vector (v_x, v_y) ; α and β are the corresponding scaling factor and offset that are applied to the reference block. Fig. 19 illustrates the LIC process. In Fig. 19, when the LIC is applied for a block, a least mean square error (LMSE) method is employed to derive the values of the LIC parameters (i.e., α and β) by minimizing the difference between the neighboring samples of the current block (i.e., the template T in Fig. 19) and their corresponding reference samples in the temporal reference pictures (i.e., either $T0$ or $T1$ in Fig. 19). Additionally, to reduce the computational complexity, both the template samples and the reference template samples are subsampled (adaptive subsampling) to derive the LIC parameters, i.e., only the shaded samples in Fig. 19 are used to derive α and β .

[0193] To improve the coding performance, no subsampling for the short side is performed as shown in Fig. 20.

2.15. Decoder side motion vector refinement (DMVR)

[0194] In order to increase the accuracy of the MVs of the merge mode, a bilateral-matching (BM) based decoder side motion vector refinement is applied in VVC. In bi-prediction operation, a refined MV is searched around the initial MVs in the reference picture list L0 and reference picture list L1. The BM method calculates the distortion between the two candidate blocks in the reference picture list L0 and list L1. As illustrated

in Fig. 21, the SAD between the two blocks based on each MV candidate (e.g., MV0' and MV1') around the initial MV is calculated. The MV candidate with the lowest SAD becomes the refined MV and used to generate the bi-predicted signal.

[0195] In VVC, the application of DMVR is restricted and is only applied for the CUs which are coded with following modes and features:

- CU level merge mode with bi-prediction MV.
- One reference picture is in the past and another reference picture is in the future with respect to the current picture.
- The distances (i.e. POC difference) from two reference pictures to the current picture are same.
- Both reference pictures are short-term reference pictures.
- CU has more than 64 luma samples.
- Both CU height and CU width are larger than or equal to 8 luma samples.
- BCW weight index indicates equal weight.
- WP is not enabled for the current block.
- CIIP mode is not used for the current block.

[0196] The refined MV derived by DMVR process is used to generate the inter prediction samples and also used in temporal motion vector prediction for future pictures coding. While the original MV is used in deblocking process and also used in spatial motion vector prediction for future CU coding.

[0197] The additional features of DMVR are mentioned in the following sub-clauses.

2.15.1. Searching scheme

[0198] In DVMR, the search points are surrounding the initial MV and the MV offset obey the MV difference mirroring rule. In other words, any points that are checked by DMVR, denoted by candidate MV pair (MV0, MV1) obey the following two equations:

$$MV0' = MV0 + MV_offset \quad (2-19)$$

$$MV1' = MV1 - MV_offset \quad (2-20)$$

where MV_offset represents the refinement offset between the initial MV and the refined MV in one of the reference pictures. The refinement search range is two integer luma samples from the initial MV. The searching includes the integer sample offset search stage and fractional sample refinement stage.

[0199] 25 points full search is applied for integer sample offset searching. The SAD of the initial MV pair is first calculated. If the SAD of the initial MV pair is smaller than a threshold, the integer sample stage of DMVR is terminated. Otherwise SADs of the remaining 24 points are calculated and checked in raster scanning order. The point with the smallest SAD is selected as the output of integer sample offset searching stage. To reduce the penalty of the uncertainty of DMVR refinement, it is proposed to favor the original MV during the DMVR process. The SAD between the reference blocks referred by the initial MV candidates is decreased by 1/4 of the SAD value.

[0200] The integer sample search is followed by fractional sample refinement. To save the calculational complexity, the fractional sample refinement is derived by using parametric error surface equation, instead of additional search with SAD comparison. The fractional sample refinement is conditionally invoked based on the output of the integer sample search stage. When the integer sample search stage is terminated with center having the smallest SAD in either the first iteration or the second iteration search, the fractional sample refinement is further applied.

[0201] In parametric error surface based sub-pixel offsets estimation, the center position cost and the costs at four neighboring positions from the center are used to fit a 2-D parabolic error surface equation of the following form

$$E(x, y) = A(x - x_{min})^2 + B(y - y_{min})^2 + C \quad (2-21)$$

where (x_{min}, y_{min}) corresponds to the fractional position with the least cost and C corresponds to the minimum cost value. By solving the above equations by using the cost value of the five search points, the (x_{min}, y_{min}) is computed as:

$$x_{min} = (E(-1,0) - E(1,0)) / (2(E(-1,0) + E(1,0) - 2E(0,0))) \quad (2-22)$$

$$y_{min} = (E(0,-1) - E(0,1)) / (2(E(0,-1) + E(0,1) - 2E(0,0))) \quad (2-23)$$

[0202] The value of x_{min} and y_{min} are automatically constrained to be between -8 and 8 since all cost values are positive and the smallest value is $E(0,0)$. This corresponds to half pel offset with 1/16th-pel MV accuracy in VVC. The computed fractional (x_{min}, y_{min}) are added to the integer distance refinement MV to get the sub-pixel accurate refinement delta MV.

2.15.2. Bilinear-interpolation and sample padding

[0203] In VVC, the resolution of the MVs is 1/16 luma samples. The samples at the fractional position are interpolated using an 8-tap interpolation filter. In DMVR, the search points are surrounding the initial fractional-pel MV with integer sample offset, therefore the samples of those fractional position need to be interpolated for DMVR search process. To reduce the calculation complexity, the bi-linear interpolation filter is used to generate the fractional samples for the searching process in DMVR. Another important effect is that by using bi-linear filter is that with 2-sample search range, the DMVR does not access more reference samples compared to the normal motion compensation process. After the refined MV is attained with DMVR search process, the normal 8-tap interpolation filter is applied to generate the final prediction. In order to not access more reference samples to normal MC process, the samples, which is not needed for the interpolation process based on the original MV but is needed for the interpolation process based on the refined MV, will be padded from those available samples.

2.15.3. Maximum DMVR processing unit

[0204] When the width and/or height of a CU are larger than 16 luma samples, it will be further split into subblocks with width and/or height equal to 16 luma samples. The maximum unit size for DMVR searching process is limit to 16x16.

2.16. Multi-pass decoder-side motion vector refinement

[0205] In this contribution, a multi-pass decoder-side motion vector refinement is applied instead of DMVR. In the first pass, bilateral matching (BM) is applied to a coding block. In the second pass, BM is applied to each 16x16 subblock within the coding block. In the third pass, MV in each 8x8 subblock is refined by applying bi-directional optical flow (BDOF). The refined MVs are stored for both spatial and temporal motion vector prediction.

2.16.1. First pass – Block based bilateral matching MV refinement

[0206] In the first pass, a refined MV is derived by applying BM to a coding block. Similar to decoder-side motion vector refinement (DMVR), the refined MV is searched around the two initial MVs (MV0 and MV1) in the reference picture lists L0 and L1. The refined MVs (MV0_pass1 and MV1_pass1) are derived around the initial MVs based on the minimum bilateral matching cost between the two reference blocks in L0 and L1.

[0207] BM performs local search to derive integer sample precision intDeltaMV and half-pel sample precision halfDeltaMV . The local search applies a 3x3 square search pattern to loop through the search range $[-sHor, sHor]$ in a horizontal direction and $[-sVer, sVer]$ in a vertical direction, wherein, the values of $sHor$ and $sVer$ are determined by the block dimension, and the maximum value of $sHor$ and $sVer$ is 8.

[0208] The bilateral matching cost is calculated as: $\text{bilCost} = \text{mvDistanceCost} + \text{sadCost}$. When the block size $\text{cbW} * \text{cbH}$ is greater than 64, MRSAD cost function is applied to remove the DC effect of the distortion between the reference blocks. When the bilCost at the center point of the 3x3 search pattern has the minimum cost, the intDeltaMV or halfDeltaMV local search is terminated. Otherwise, the current minimum cost search point becomes the new center point of the 3x3 search pattern and the search for the minimum cost continues, until it reaches the end of the search range.

[0209] The existing fractional sample refinement is further applied to derive the final deltaMV . The refined MVs after the first pass are then derived as:

- $MV0_pass1 = MV0 + \Delta MV$
- $MV1_pass1 = MV1 - \Delta MV$

2.16.2. Second pass – Subblock based bilateral matching MV refinement

[0210] In the second pass, a refined MV is derived by applying BM to a 16×16 grid subblock. For each subblock, the refined MV is searched around the two MVs ($MV0_pass1$ and $MV1_pass1$), obtained on the first pass for the reference picture list L0 and L1. The refined MVs ($MV0_pass2(sbIdx2)$ and $MV1_pass2(sbIdx2)$) are derived based on the minimum bilateral matching cost between the two reference subblocks in L0 and L1.

[0211] For each subblock, BM performs full search to derive integer sample precision $intDeltaMV$. The full search has a search range $[-sHor, sHor]$ in a horizontal direction and $[-sVer, sVer]$ in a vertical direction, wherein, the values of $sHor$ and $sVer$ are determined by the block dimension, and the maximum value of $sHor$ and $sVer$ is 8.

[0212] The bilateral matching cost is calculated by applying a cost factor to the SATD cost between the two reference subblocks, as: $bilCost = satdCost * costFactor$. The search area $(2*sHor + 1) * (2*sVer + 1)$ is divided up to 5 diamond shape search regions shown on Fig. 22. Each search region is assigned a costFactor, which is determined by the distance ($intDeltaMV$) between each search point and the starting MV, and each diamond region is processed in the order starting from the center of the search area. In each region, the search points are processed in the raster scan order starting from the top left going to the bottom right corner of the region. When the minimum $bilCost$ within the current search region is less than a threshold equal to $sbW * sbH$, the int-pel full search is terminated, otherwise, the int-pel full search continues to the next search region until all search points are examined. Fig. 22 shows diamond regions in the search area.

[0213] BM performs local search to derive half sample precision $halfDeltaMv$. The search pattern and cost function are the same as defined in 2.9.1.

[0214] The existing VVC DMVR fractional sample refinement is further applied to

derive the final $\text{deltaMV}(\text{sbIdx2})$. The refined MVs at second pass is then derived as:

- $\text{MV0_pass2}(\text{sbIdx2}) = \text{MV0_pass1} + \text{deltaMV}(\text{sbIdx2})$
- $\text{MV1_pass2}(\text{sbIdx2}) = \text{MV1_pass1} - \text{deltaMV}(\text{sbIdx2})$

2.16.3. Third pass – Subblock based bi-directional optical flow MV refinement

[0215] In the third pass, a refined MV is derived by applying BDOF to an 8×8 grid subblock. For each 8×8 subblock, BDOF refinement is applied to derive scaled V_x and V_y without clipping starting from the refined MV of the parent subblock of the second pass. The derived $\text{bioMv}(V_x, V_y)$ is rounded to 1/16 sample precision and clipped between -32 and 32.

[0216] The refined MVs ($\text{MV0_pass3}(\text{sbIdx3})$ and $\text{MV1_pass3}(\text{sbIdx3})$) at third pass are derived as:

- $\text{MV0_pass3}(\text{sbIdx3}) = \text{MV0_pass2}(\text{sbIdx2}) + \text{bioMv}$
- $\text{MV1_pass3}(\text{sbIdx3}) = \text{MV0_pass2}(\text{sbIdx2}) - \text{bioMv}$

2.17. Sample-based BDOF

[0217] In the sample-based BDOF, instead of deriving motion refinement (V_x, V_y) on a block basis, it is performed per sample.

[0218] The coding block is divided into 8×8 subblocks. For each subblock, whether to apply BDOF or not is determined by checking the SAD between the two reference subblocks against a threshold. If decided to apply BDOF to a subblock, for every sample in the subblock, a sliding 5×5 window is used and the existing BDOF process is applied for every sliding window to derive V_x and V_y . The derived motion refinement (V_x, V_y) is applied to adjust the bi-predicted sample value for the center sample of the window.

2.18. Extended merge prediction

[0219] In VVC, the merge candidate list is constructed by including the following five types of candidates in order:

- (1) Spatial MVP from spatial neighbour CUs.
- (2) Temporal MVP from collocated CUs.
- (3) History-based MVP from a FIFO table.

(4) Pairwise average MVP.

(5) Zero MVs.

[0220] The size of merge list is signalled in sequence parameter set header and the maximum allowed size of merge list is 6. For each CU code in merge mode, an index of best merge candidate is encoded using truncated unary binarization (TU). The first bin of the merge index is coded with context and bypass coding is used for other bins.

[0221] The derivation process of each category of merge candidates is provided in this session. As done in HEVC, VVC also supports parallel derivation of the merging candidate lists for all CUs within a certain size of area.

2.18.1. Spatial candidates derivation

[0222] The derivation of spatial merge candidates in VVC is same to that in HEVC except the positions of first two merge candidates are swapped. A maximum of four merge candidates are selected among candidates located in the positions depicted in Fig. 23. The order of derivation is B0, A0, B1, A1 and B2. Position B2 is considered only when one or more than one CUs of position B0, A0, B1, A1 are not available (e.g. because it belongs to another slice or tile) or is intra coded. After candidate at position A1 is added, the addition of the remaining candidates is subject to a redundancy check which ensures that candidates with same motion information are excluded from the list so that coding efficiency is improved. To reduce computational complexity, not all possible candidate pairs are considered in the mentioned redundancy check. Instead only the pairs linked with an arrow in Fig. 24 are considered and a candidate is only added to the list if the corresponding candidate used for redundancy check has not the same motion information.

2.18.2. Temporal candidates derivation

[0223] In this step, only one candidate is added to the list. Particularly, in the derivation of this temporal merge candidate, a scaled motion vector is derived based on co-located CU belonging to the collocated reference picture. The reference picture list to be used for derivation of the co-located CU is explicitly signalled in the slice header. The scaled

motion vector for temporal merge candidate is obtained as illustrated by the dotted line in Fig. 25, which is scaled from the motion vector of the co-located CU using the POC distances, t_b and t_d , where t_b is defined to be the POC difference between the reference picture of the current picture and the current picture and t_d is defined to be the POC difference between the reference picture of the co-located picture and the co-located picture. The reference picture index of temporal merge candidate is set equal to zero.

[0224] The position for the temporal candidate is selected between candidates C0 and C1, as depicted in Fig. 26. If CU at position C0 is not available, is intra coded, or is outside of the current row of CTUs, position C1 is used. Otherwise, position C0 is used in the derivation of the temporal merge candidate.

2.18.3. History-based merge candidates derivation

[0225] The history-based MVP (HMVP) merge candidates are added to merge list after the spatial MVP and TMVP. In this method, the motion information of a previously coded block is stored in a table and used as MVP for the current CU. The table with multiple HMVP candidates is maintained during the encoding/decoding process. The table is reset (emptied) when a new CTU row is encountered. Whenever there is a non-subblock inter-coded CU, the associated motion information is added to the last entry of the table as a new HMVP candidate.

[0226] The HMVP table size S is set to be 6, which indicates up to 6 History-based MVP (HMVP) candidates may be added to the table. When inserting a new motion candidate to the table, a constrained first-in-first-out (FIFO) rule is utilized wherein redundancy check is firstly applied to find whether there is an identical HMVP in the table. If found, the identical HMVP is removed from the table and all the HMVP candidates afterwards are moved forward, and the identical HMVP is inserted to the last entry of the table.

[0227] HMVP candidates could be used in the merge candidate list construction process. The latest several HMVP candidates in the table are checked in order and inserted to the

candidate list after the TMVP candidate. Redundancy check is applied on the HMVP candidates to the spatial or temporal merge candidate.

[0228] To reduce the number of redundancy check operations, the following simplifications are introduced:

[0229] Number of HMPV candidates is used for merge list generation is set as $(N \leq 4)$?
 $M: (8 - N)$, wherein N indicates number of existing candidates in the merge list and M indicates number of available HMVP candidates in the table.

[0230] Once the total number of available merge candidates reaches the maximally allowed merge candidates minus 1, the merge candidate list construction process from HMVP is terminated.

2.18.4. Pair-wise average merge candidates derivation

[0231] Pairwise average candidates are generated by averaging predefined pairs of candidates in the existing merge candidate list, and the predefined pairs are defined as $\{(0, 1), (0, 2), (1, 2), (0, 3), (1, 3), (2, 3)\}$, where the numbers denote the merge indices to the merge candidate list. The averaged motion vectors are calculated separately for each reference list. If both motion vectors are available in one list, these two motion vectors are averaged even when they point to different reference pictures; if only one motion vector is available, use the one directly; if no motion vector is available, keep this list invalid.

[0232] When the merge list is not full after pair-wise average merge candidates are added, the zero MVPs are inserted in the end until the maximum merge candidate number is encountered.

2.18.5. Merge estimation region

[0233] Merge estimation region (MER) allows independent derivation of merge candidate list for the CUs in the same merge estimation region (MER). A candidate block that is within the same MER to the current CU is not included for the generation of the

merge candidate list of the current CU. In addition, the updating process for the history-based motion vector predictor candidate list is updated only if $(x_{Cb} + cbWidth) \gg \text{Log2ParMrgLevel}$ is greater than $x_{Cb} \gg \text{Log2ParMrgLevel}$ and $(y_{Cb} + cbHeight) \gg \text{Log2ParMrgLevel}$ is greater than $(y_{Cb} \gg \text{Log2ParMrgLevel})$ and where (x_{Cb}, y_{Cb}) is the top-left luma sample position of the current CU in the picture and $(cbWidth, cbHeight)$ is the CU size. The MER size is selected at encoder side and signalled as `log2_parallel_merge_level_minus2` in the sequence parameter set.

2.19. New merge candidates

2.19.1. Non-adjacent merge candidates derivation

[0234] In VVC, five spatially neighboring blocks shown in Fig. 27 as well as one temporal neighbor are used to derive merge candidates.

[0235] It is proposed to derive the additional merge candidates from the positions non-adjacent to the current block using the same pattern as that in VVC. To achieve this, for each search round i , a virtual block is generated based on the current block as follows:

First, the relative position of the virtual block to the current block is calculated by:

$$\text{Offsetx} = -i \times \text{gridX}, \text{Offsety} = -i \times \text{gridY}$$

where the `Offsetx` and `Offsety` denote the offset of the top-left corner of the virtual block relative to the top-left corner of the current block, `gridX` and `gridY` are the width and height of the search grid.

Second, the width and height of the virtual block are calculated by:

$$\text{newWidth} = i \times 2 \times \text{gridX} + \text{currWidth} \quad \text{newHeight} = i \times 2 \times \text{gridY} + \text{currHeight}.$$

where the `currWidth` and `currHeight` are the width and height of current block. The `newWidth` and `newHeight` are the width and height of new virtual block.

`gridX` and `gridY` are currently set to `currWidth` and `currHeight`, respectively.

[0236] Fig. 28 is an illustration of virtual block in the i -th search round. After generating the virtual block, the blocks A_i , B_i , C_i , D_i and E_i can be regarded as the VVC spatial neighboring blocks of the virtual block and their positions are obtained with the

same pattern as that in VVC. Obviously, the virtual block is the current block if the search round i is 0. In this case, the blocks A_i , B_i , C_i , D_i and E_i are the spatially neighboring blocks that are used in VVC merge mode.

[0237] When constructing the merge candidate list, the pruning is performed to guarantee each element in merge candidate list to be unique. The maximum search round is set to 1, which means that five non-adjacent spatial neighbor blocks are utilized.

[0238] Non-adjacent spatial merge candidates are inserted into the merge list after the temporal merge candidate in the order of $B_1 \rightarrow A_1 \rightarrow C_1 \rightarrow D_1 \rightarrow E_1$.

2.19.2. STMVP

[0239] It is proposed to derive an averaging candidate as STMVP candidate using three spatial merge candidates and one temporal merge candidate.

[0240] STMVP is inserted before the above-left spatial merge candidate.

[0241] The STMVP candidate is pruned with all the previous merge candidates in the merge list.

[0242] For the spatial candidates, the first three candidates in the current merge candidate list are used.

[0243] For the temporal candidate, the same position as VTM / HEVC collocated position is used.

[0244] For the spatial candidates, the first, second, and third candidates inserted in the current merge candidate list before STMVP are denoted as F, S, and T.

[0245] The temporal candidate with the same position as VTM / HEVC collocated position used in TMVP is denoted as Col.

[0246] The motion vector of the STMVP candidate in prediction direction X (denoted as mv_{LX}) is derived as follows:

- 1) If the reference indices of the four merge candidates are all valid and are all equal to zero in prediction direction X (X = 0 or 1),

$$mvLX = (mvLX_F + mvLX_S + mvLX_T + mvLX_Col) >> 2$$

- 2) If reference indices of three of the four merge candidates are valid and are equal to zero in prediction direction X (X = 0 or 1),

$$mvLX = (mvLX_F \times 3 + mvLX_S \times 3 + mvLX_Col \times 2) >> 3 \quad \text{or}$$

$$mvLX = (mvLX_F \times 3 + mvLX_T \times 3 + mvLX_Col \times 2) >> 3 \quad \text{or}$$

$$mvLX = (mvLX_S \times 3 + mvLX_T \times 3 + mvLX_Col \times 2) >> 3$$

- 3) If reference indices of two of the four merge candidates are valid and are equal to zero in prediction direction X (X = 0 or 1),

$$mvLX = (mvLX_F + mvLX_Col) >> 1 \quad \text{or}$$

$$mvLX = (mvLX_S + mvLX_Col) >> 1 \quad \text{or}$$

$$mvLX = (mvLX_T + mvLX_Col) >> 1$$

Note: If the temporal candidate is unavailable, the STMVP mode is off.

2.19.3. Merge list size

[0247] If considering both non-adjacent and STMVP merge candidates, the size of merge list is signalled in sequence parameter set header and the maximum allowed size of merge list is 8.

2.20. Geometric partitioning mode (GPM)

[0248] In VVC, a geometric partitioning mode is supported for inter prediction. The geometric partitioning mode is signalled using a CU-level flag as one kind of merge mode, with other merge modes including the regular merge mode, the MMVD mode, the CIIP mode and the subblock merge mode. In total 64 partitions are supported by geometric partitioning mode for each possible CU size $w \times h = 2^m \times 2^n$ with $m, n \in \{3 \dots 6\}$ excluding 8x64 and 64x8.

[0249] Fig. 29 shows examples of the GPM splits grouped by identical angles. When this mode is used, a CU is split into two parts by a geometrically located straight line (Fig. 29). The location of the splitting line is mathematically derived from the angle and offset parameters of a specific partition. Each part of a geometric partition in the CU is inter-predicted using its own motion; only uni-prediction is allowed for each partition, that is, each part has one motion vector and one reference index. The uni-prediction motion

constraint is applied to ensure that same as the conventional bi-prediction, only two motion compensated prediction are needed for each CU. The uni-prediction motion for each partition is derived using the process described in 2.20.1.

[0250] If geometric partitioning mode is used for the current CU, then a geometric partition index indicating the partition mode of the geometric partition (angle and offset), and two merge indices (one for each partition) are further signalled. The number of maximum GPM candidate size is signalled explicitly in SPS and specifies syntax binarization for GPM merge indices. After predicting each of part of the geometric partition, the sample values along the geometric partition edge are adjusted using a blending processing with adaptive weights as in 2.20.2. This is the prediction signal for the whole CU, and transform and quantization process will be applied to the whole CU as in other prediction modes. Finally, the motion field of a CU predicted using the geometric partition modes is stored as in 2.20.3.

2.20.1. Uni-prediction candidate list construction

[0251] The uni-prediction candidate list is derived directly from the merge candidate list constructed according to the extended merge prediction process in 2.18. Denote n as the index of the uni-prediction motion in the geometric uni-prediction candidate list. The LX motion vector of the n -th extended merge candidate, with X equal to the parity of n , is used as the n -th uni-prediction motion vector for geometric partitioning mode. These motion vectors are marked with “x” in Fig. 30. In case a corresponding LX motion vector of the n -th extended merge candidate does not exist, the $L(1 - X)$ motion vector of the same candidate is used instead as the uni-prediction motion vector for geometric partitioning mode.

2.20.2. Blending along the geometric partitioning edge

[0252] After predicting each part of a geometric partition using its own motion, blending is applied to the two prediction signals to derive samples around geometric partition edge.

The blending weight for each position of the CU are derived based on the distance between individual position and the partition edge.

[0253] The distance for a position (x, y) to the partition edge are derived as:

$$d(x, y) = (2x + 1 - w) \cos(\varphi_i) + (2y + 1 - h) \sin(\varphi_i) - \rho_j \quad (2-24)$$

$$\rho_j = \rho_{x,j} \cos(\varphi_i) + \rho_{y,j} \sin(\varphi_i) \quad (2-25)$$

$$\rho_{x,j} = \begin{cases} 0 & i \% 16 = 8 \text{ or } (i \% 16 \neq 0 \text{ and } h \geq w) \\ \pm(j \times w) \gg 2 & \text{otherwise} \end{cases} \quad (2-26)$$

$$\rho_{y,j} = \begin{cases} \pm(j \times h) \gg 2 & i \% 16 = 8 \text{ or } (i \% 16 \neq 0 \text{ and } h \geq w) \\ 0 & \text{otherwise} \end{cases} \quad (2-27)$$

where i, j are the indices for angle and offset of a geometric partition, which depend on the signaled geometric partition index. The sign of $\rho_{x,j}$ and $\rho_{y,j}$ depend on angle index i .

The weights for each part of a geometric partition are derived as following:

$$wIdxL(x, y) = partIdx ? 32 + d(x, y) : 32 - d(x, y) \quad (2-28)$$

$$w_0(x, y) = \frac{Clip3(0, 8, (wIdxL(x, y) + 4) \gg 3)}{8} \quad (2-29)$$

$$w_1(x, y) = 1 - w_0(x, y) \quad (2-30)$$

[0254] The $partIdx$ depends on the angle index i . One example of weigh w_0 is illustrated in Fig. 31. Fig. 31 shows exemplified generation of a bending weight w_0 using geometric partitioning mode.

2.20.3. Motion field storage for geometric partitioning mode

[0255] $Mv1$ from the first part of the geometric partition, $Mv2$ from the second part of the geometric partition and a combined Mv of $Mv1$ and $Mv2$ are stored in the motion filed of a geometric partitioning mode coded CU.

[0256] The stored motion vector type for each individual position in the motion filed are determined as:

$$sType = abs(motionIdx) < 32 ? 2 : (motionIdx \leq 0 ? (1 - partIdx) : partIdx) \quad (2-31)$$

where $motionIdx$ is equal to $d(4x + 2, 4y + 2)$, which is recalculated from equation (2-18). The $partIdx$ depends on the angle index i .

[0257] If $sType$ is equal to 0 or 1, $Mv0$ or $Mv1$ are stored in the corresponding motion field, otherwise if $sType$ is equal to 2, a combined Mv from $Mv0$ and $Mv2$ are stored. The combined Mv are generated using the following process:

- 1) If Mv1 and Mv2 are from different reference picture lists (one from L0 and the other from L1), then Mv1 and Mv2 are simply combined to form the bi-prediction motion vectors.

Otherwise, if Mv1 and Mv2 are from the same list, only uni-prediction motion Mv2 is stored.

2.21. Multi-hypothesis prediction

[0258] In multi-hypothesis prediction (MHP), up to two additional predictors are signalled on top of inter AMVP mode, regular merge mode, affine merge and MMVD mode. The resulting overall prediction signal is accumulated iteratively with each additional prediction signal.

$$p_{n+1} = (1 - \alpha_{n+1})p_n + \alpha_{n+1}h_{n+1}$$

[0259] The weighting factor α is specified according to the following Table 2-4:

Table 2-4 – weighting factor for MHP

add_hyp_weight_idx	α
0	1/4
1	-1/8

[0260] For inter AMVP mode, MHP is only applied if non-equal weight in BCW is selected in bi-prediction mode.

[0261] The additional hypothesis can be either merge or AMVP mode. In the case of merge mode, the motion information is indicated by a merge index, and the merge candidate list is the same as in the Geometric Partition Mode. In the case of AMVP mode, the reference index, MVP index, and MVD are signaled.

2.22. Non-adjacent spatial candidate

[0262] Fig. 32 shows spatial neighboring blocks used to derive the spatial merge candidates. The non-adjacent spatial merge candidates are inserted after the TMVP in the regular merge candidate list. The pattern of the spatial merge candidates is shown on Fig. 32. The distances between the non-adjacent spatial candidates and the current coding block are based on the width and height of the current coding block.

2.23. Template matching (TM)

[0263] Template matching (TM) is a decoder-side MV derivation method to refine the motion information of the current CU by finding the closest match between a template (i.e., top and/or left neighbouring blocks of the current CU) in the current picture and a block (i.e., same size to the template) in a reference picture. As illustrated in Fig. 33, a better MV is to be searched around the initial motion of the current CU within a $[-8, +8]$ -pel search range. The template matching that was proposed two modifications: search step size is determined based on AMVR mode and TM can be cascaded with bilateral matching process in merge modes.

[0264] In AMVP mode, an MVP candidate is determined based on template matching error to pick up the one which reaches the minimum difference between current block template and reference block template, and then TM performs only for this particular MVP candidate for MV refinement. TM refines this MVP candidate, starting from full-pel MVD precision (or 4-pel for 4-pel AMVR mode) within a $[-8, +8]$ -pel search range by using iterative diamond search. The AMVP candidate may be further refined by using cross search with full-pel MVD precision (or 4-pel for 4-pel AMVR mode), followed sequentially by half-pel and quarter-pel ones depending on AMVR mode as specified in Table 2-5. This search process ensures that the MVP candidate still keeps the same MV precision as indicated by AMVR mode after TM process.

Table 2-5 – Search patterns of AMVR and merge mode with AMVR.

Search pattern	AMVR mode				Merge mode	
	4-pel	Full-pel	Half-pel	Quarter-pel	AltIF=0	AltIF=1
4-pel diamond	v					
4-pel cross	v					
Full-pel diamond		v	v	v	v	v
Full-pel cross		v	v	v	v	v

Half-pel cross			v	v	v	v
Quarter-pel cross				v	v	
1/8-pel cross					v	

[0265] In merge mode, similar search method is applied to the merge candidate indicated by the merge index. As Table 2-5 shows, TM may perform all the way down to 1/8-pel MVD precision or skipping those beyond half-pel MVD precision, depending on whether the alternative interpolation filter (that is used when AMVR is of half-pel mode) is used according to merged motion information. Besides, when TM mode is enabled, template matching may work as an independent process or an extra MV refinement process between block-based and subblock-based bilateral matching (BM) methods, depending on whether BM can be enabled or not according to its enabling condition check.

2.24. Overlapped block motion compensation (OBMC)

[0266] Overlapped Block Motion Compensation (OBMC) has previously been used in H.263. In the JEM, unlike in H.263, OBMC can be switched on and off using syntax at the CU level. When OBMC is used in the JEM, the OBMC is performed for all motion compensation (MC) block boundaries except the right and bottom boundaries of a CU. Moreover, it is applied for both the luma and chroma components. In the JEM, a MC block is corresponding to a coding block. When a CU is coded with sub-CU mode (includes sub-CU merge, affine and FRUC mode), each sub-block of the CU is a MC block. To process CU boundaries in a uniform fashion, OBMC is performed at sub-block level for all MC block boundaries, where sub-block size is set equal to 4×4, as illustrated in Fig. 34. Fig. 34 is an illustration of sub-blocks where OBMC applies.

[0267] When OBMC applies to the current sub-block, besides current motion vectors, motion vectors of four connected neighbouring sub-blocks, if available and are not identical to the current motion vector, are also used to derive prediction block for the current sub-block. These multiple prediction blocks based on multiple motion vectors are

combined to generate the final prediction signal of the current sub-block.

[0268] Prediction block based on motion vectors of a neighbouring sub-block is denoted as P_N , with N indicating an index for the neighbouring *above*, *below*, *left* and *right* sub-blocks and prediction block based on motion vectors of the current sub-block is denoted as P_C . When P_N is based on the motion information of a neighbouring sub-block that contains the same motion information to the current sub-block, the OBMC is not performed from P_N . Otherwise, every sample of P_N is added to the same sample in P_C , i.e., four rows/columns of P_N are added to P_C . The weighting factors $\{1/4, 1/8, 1/16, 1/32\}$ are used for P_N and the weighting factors $\{3/4, 7/8, 15/16, 31/32\}$ are used for P_C . The exception are small MC blocks, (i.e., when height or width of the coding block is equal to 4 or a CU is coded with sub-CU mode), for which only two rows/columns of P_N are added to P_C . In this case weighting factors $\{1/4, 1/8\}$ are used for P_N and weighting factors $\{3/4, 7/8\}$ are used for P_C . For P_N generated based on motion vectors of vertically (horizontally) neighbouring sub-block, samples in the same row (column) of P_N are added to P_C with a same weighting factor.

[0269] In the JEM, for a CU with size less than or equal to 256 luma samples, a CU level flag is signalled to indicate whether OBMC is applied or not for the current CU. For the CUs with size larger than 256 luma samples or not coded with AMVP mode, OBMC is applied by default. At the encoder, when OBMC is applied for a CU, its impact is taken into account during the motion estimation stage. The prediction signal formed by OBMC using motion information of the top neighbouring block and the left neighbouring block is used to compensate the top and left boundaries of the original signal of the current CU, and then the normal motion estimation process is applied.

2.25. Multiple transform selection (MTS) for core transform

[0270] In addition to DCT-II which has been employed in HEVC, a Multiple Transform Selection (MTS) scheme is used for residual coding both inter and intra coded blocks. It uses multiple selected transforms from the DCT8/DST7. The newly introduced transform

matrices are DST-VII and DCT-VIII. Table 2-6 shows the basis functions of the selected DST/DCT.

Table 2-6 – Transform basis functions of DCT-II/ VIII and DSTVII for N-point input

Transform Type	Basis function $T_i(j)$, $i, j = 0, 1, \dots, N-1$
DCT-II	$T_i(j) = \omega_0 \cdot \sqrt{\frac{2}{N}} \cdot \cos\left(\frac{\pi \cdot i \cdot (2j + 1)}{2N}\right)$ where, $\omega_0 = \begin{cases} \sqrt{\frac{2}{N}} & i = 0 \\ 1 & i \neq 0 \end{cases}$
DCT-VIII	$T_i(j) = \sqrt{\frac{4}{2N + 1}} \cdot \cos\left(\frac{\pi \cdot (2i + 1) \cdot (2j + 1)}{4N + 2}\right)$
DST-VII	$T_i(j) = \sqrt{\frac{4}{2N + 1}} \cdot \sin\left(\frac{\pi \cdot (2i + 1) \cdot (j + 1)}{2N + 1}\right)$

[0271] In order to keep the orthogonality of the transform matrix, the transform matrices are quantized more accurately than the transform matrices in HEVC. To keep the intermediate values of the transformed coefficients within the 16-bit range, after horizontal and after vertical transform, all the coefficients are to have 10-bit.

[0272] In order to control MTS scheme, separate enabling flags are specified at SPS level for intra and inter, respectively. When MTS is enabled at SPS, a CU level flag is signalled to indicate whether MTS is applied or not. Here, MTS is applied only for luma. The MTS signaling is skipped when one of the below conditions is applied.

- The position of the last significant coefficient for the luma TB is less than 1 (i.e., DC only).
- The last significant coefficient of the luma TB is located inside the MTS zero-out region.

[0273] If MTS CU flag is equal to zero, then DCT2 is applied in both directions. However, if MTS CU flag is equal to one, then two other flags are additionally signalled to indicate the transform type for the horizontal and vertical directions, respectively. Transform and signalling mapping table as shown in Table 2-7. Unified the transform selection for ISP

and implicit MTS is used by removing the intra-mode and block-shape dependencies. If current block is ISP mode or if the current block is intra block and both intra and inter explicit MTS is on, then only DST7 is used for both horizontal and vertical transform cores. When it comes to transform matrix precision, 8-bit primary transform cores are used. Therefore, all the transform cores used in HEVC are kept as the same, including 4-point DCT-2 and DST-7, 8-point, 16-point and 32-point DCT-2. Also, other transform cores including 64-point DCT-2, 4-point DCT-8, 8-point, 16-point, 32-point DST-7 and DCT-8, use 8-bit primary transform cores.

Table 2-7 – Transform and signalling mapping table

MTS_CU_flag	MTS_Hor_flag	MTS_Ver_flag	Intra/inter	
g	g	g	Horizontal	Vertical
0			DCT2	
1	0	0	DST7	DST7
	0	1	DCT8	DST7
	1	0	DST7	DCT8
	1	1	DCT8	DCT8

[0274] To reduce the complexity of large size DST-7 and DCT-8, High frequency transform coefficients are zeroed out for the DST-7 and DCT-8 blocks with size (width or height, or both width and height) equal to 32. Only the coefficients within the 16x16 lower-frequency region are retained.

[0275] As in HEVC, the residual of a block can be coded with transform skip mode. To avoid the redundancy of syntax coding, the transform skip flag is not signalled when the CU level MTS_CU_flag is not equal to zero. Note that implicit MTS transform is set to DCT2 when LFNST or MIP is activated for the current CU. Also the implicit MTS can be still enabled when MTS is enabled for inter coded blocks.

2.26. Subblock transform (SBT)

[0276] In VTM, subblock transform is introduced for an inter-predicted CU. In this transform mode, only a sub-part of the residual block is coded for the CU. When inter-predicted CU with `cu_cbf` equal to 1, `cu_sbt_flag` may be signaled to indicate whether the whole residual block or a sub-part of the residual block is coded. In the former case, inter MTS information is further parsed to determine the transform type of the CU. In the latter case, a part of the residual block is coded with inferred adaptive transform and the other part of the residual block is zeroed out.

[0277] When SBT is used for an inter-coded CU, SBT type and SBT position information are signaled in the bitstream. There are two SBT types and two SBT positions, as indicated in Fig. 35. For SBT-V (or SBT-H), the TU width (or height) may equal to half of the CU width (or height) or 1/4 of the CU width (or height), resulting in 2:2 split or 1:3/3:1 split. The 2:2 split is like a binary tree (BT) split while the 1:3/3:1 split is like an asymmetric binary tree (ABT) split. In ABT splitting, only the small region contains the non-zero residual. If one dimension of a CU is 8 in luma samples, the 1:3/3:1 split along that dimension is disallowed. There are at most 8 SBT modes for a CU.

[0278] Position-dependent transform core selection is applied on luma transform blocks in SBT-V and SBT-H (chroma TB always using DCT-2). The two positions of SBT-H and SBT-V are associated with different core transforms. More specifically, the horizontal and vertical transforms for each SBT position is specified in Fig. 35. For example, the horizontal and vertical transforms for SBT-V position 0 is DCT-8 and DST-7, respectively. When one side of the residual TU is greater than 32, the transform for both dimensions is set as DCT-2. Therefore, the subblock transform jointly specifies the TU tiling, cbf, and horizontal and vertical core transform type of a residual block. Fig. 35 shows SBT position, type and transform type.

The SBT is not applied to the CU coded with combined inter-intra mode.

2.27. Template matching based adaptive merge candidate reorder

[0279] To improve the coding efficiency, after the merge candidate list is constructed,

the order of each merge candidate is adjusted according to the template matching cost. The merge candidates are arranged in the list in accordance with the template matching cost of ascending order. It is operated in the form of sub-group.

[0280] The template matching cost is measured by the SAD (Sum of absolute differences) between the neighbouring samples of the current CU and their corresponding reference samples. If a merge candidate includes bi-predictive motion information, the corresponding reference samples are the average of the corresponding reference samples in reference list0 and the corresponding reference samples in reference list1, as illustrated in Fig. 36. If a merge candidate includes sub-CU level motion information, the corresponding reference samples consist of the neighbouring samples of the corresponding reference sub-blocks, as illustrated in Fig. 37.

[0281] The sorting process is operated in the form of sub-group, as illustrated in Fig. 38. The first three merge candidates are sorted together. The following three merge candidates are sorted together.

[0282] The template size (width of the left template or height of the above template) is 1. The sub-group size is 3.

2.28. Adaptive Merge Candidate List

[0283] It can assume the number of the merge candidates is 8. We take the first 5 merge candidates as a first subgroup and take the following 3 merge candidates as a second subgroup (i.e., the last subgroup).

[0284] For the encoder, after the merge candidate list is constructed, some merge candidates are adaptively reordered in an ascending order of costs of merge candidates as shown in Fig. 39.

[0285] More specifically, the template matching costs for the merge candidates in all subgroups except the last subgroup are computed; then reorder the merge candidates in their own subgroups except the last subgroup; finally, the final merge candidate list will

be got.

[0286] For the decoder, after the merge candidate list is constructed, some/no merge candidates are adaptively reordered in ascending order of costs of merge candidates as shown in Fig. 40. In Fig. 40, the subgroup the selected (signaled) merge candidate located in is called the selected subgroup.

[0287] More specifically, if the selected merge candidate is located in the last subgroup, the merge candidate list construction process is terminated after the selected merge candidate is derived, no reorder is performed and the merge candidate list is not changed; otherwise, the execution process is as follows:

[0288] The merge candidate list construction process is terminated after all the merge candidates in the selected subgroup are derived; compute the template matching costs for the merge candidates in the selected subgroup; reorder the merge candidates in the selected subgroup; finally, a new merge candidate list will be got.

[0289] For both encoder and decoder, a template matching cost is derived as a function of T and RT, wherein T is a set of samples in the template and RT is a set of reference samples for the template.

[0290] When deriving the reference samples of the template for a merge candidate, the motion vectors of the merge candidate are rounded to the integer pixel accuracy.

[0291] The reference samples of the template (RT) for bi-directional prediction are derived by weighted averaging of the reference samples of the template in reference list0 (RT_0) and the reference samples of the template in reference list1 (RT_1) as follows.

$$RT = ((8 - w) * RT_0 + w * RT_1 + 4) \gg 3 \quad (2-32)$$

where the weight of the reference template in reference list0 ($8-w$) and the weight of the reference template in reference list1 (w) are decided by the BCW index of the merge candidate. BCW index equal to $\{0,1,2,3,4\}$ corresponds to w equal to $\{-2,3,4,5,10\}$, respectively.

[0292] If the Local Illumination Compensation (LIC) flag of the merge candidate is true, the reference samples of the template are derived with LIC method.

[0293] The template matching cost is calculated based on the sum of absolute differences (SAD) of T and RT.

[0294] The template size is 1. That means the width of the left template and/or the height of the above template is 1.

[0295] If the coding mode is MMVD, the merge candidates to derive the base merge candidates are not reordered.

[0296] If the coding mode is GPM, the merge candidates to derive the uni-prediction candidate list are not reordered.

2.29. IBC with extended reference area

[0297] An IBC reference area design is proposed that does not increase the current memory area required by ECM-3 and tests the performance.

[0298] Fig. 41 illustrates the extended reference area. In the figure, the blue square denotes the current CTU and the green ones denote CTUs that may be used by IBC reference. Specifically, assume that W denotes the maximum horizontal CTU index and the current CTU index is (m, n), for coding units in the current CTU, CTUs with index (0, n)...(m, n) and (m-1, n)...(W, n) defines the reference area that can be used by IBC.

[0299] One reason to have such a design is that in the current ECM, the left, above and upper-left CTUs are being used and thus need to be saved. To achieve this, all CTUs to the right of the above CTU in the above CTU row (for CTUs to be coded in the current CTU row) and all CTUs to the left of the current CTU in the current CTU row (for CTUs to be coded in the next CTU row) must be kept. It means that such a design does not increase the buffer size required by the current ECM.

2.30. IBC with Template Matching

[0300] It is proposed to also use Template Matching with IBC for both IBC merge mode and IBC AMVP mode.

[0301] The IBC-TM merge list has been modified compared to the one used by regular IBC merge mode such that the candidates are selected according to a pruning method with a motion distance between the candidates as in the regular TM merge mode. The ending zero motion fulfillment (which is a nonsense regarding Intra coding) has been replaced by motion vectors to the left $(-W, 0)$, top $(0, -H)$ and top-left $(-W, -H)$ CUs, then, if necessary, the list is fulfilled with the left one without pruning.

[0302] In the IBC-TM merge mode, the selected candidates are refined with the Template Matching method prior to the RDO or decoding process. The IBC-TM merge mode has been put in competition with the regular IBC merge mode and a TM-merge flag is signaled.

[0303] In the IBC-TM AMVP mode, up to 3 candidates are selected from the IBC merge list. Each of those 3 selected candidates are refined using the Template Matching method and sorted according to their resulting Template Matching cost. Only the 2 first ones are then considered in the motion estimation process as usual.

[0304] The Template Matching refinement for both IBC-TM merge and AMVP modes is quite simple since IBC motion vectors are constrained to be integer and within a reference region as shown in Fig. 42. So, in IBC-TM merge mode, all refinements are performed at integer precision, and in IBC-TM AMVP mode, they are performed either at integer or 4-pel precision. In both cases, the refined motion vectors in each refinement step must respect the constraint of the reference region.

2.31. Reconstruction-Reordered IBC (RR-IBC)

[0305] Fig. 43 shows examples of symmetry in screen content pictures. Screen content coding tools like Intra Block Copy (IBC) generate a prediction block by directly copying a prior coded reference region in the same picture. Symmetry is often observed in video

content, especially in text character regions and computer-generated graphics in screen content sequences, as shown in Fig. 43. Therefore, a specific screen content coding tool considering the symmetry would be efficient to compress such kinds of video contents.

[0306] A Reconstruction-Reordered IBC (RR-IBC) mode is proposed for screen content video coding. When it is applied, the samples in a reconstruction block are flipped according to a flip type of the current block. At the encoder side, the original block is flipped before motion search and residual calculation, while the prediction block is derived without flipping. At the decoder side, the reconstruction block is flipped back to restore the original block.

[0307] Two flip methods, horizontal flip and vertical flip, are supported for RR-IBC coded blocks. A syntax flag is firstly signalled for an IBC AMVP coded block, indicating whether the reconstruction is flipped, and if it is flipped, another flag is further signaled specifying the flip type. For IBC merge, the flip type is inherited from neighbouring blocks, without syntax signalling. Considering the horizontal or vertical symmetry, the current block and the reference block are normally aligned horizontally or vertically. Therefore, when a horizontal flip is applied, the vertical component of the BV is not signaled and inferred to be equal to 0. Similarly, the horizontal component of the BV is not signaled and inferred to be equal to 0 when a vertical flip is applied.

[0308] To better utilize the symmetry property, a flip-aware BV adjustment approach is applied to refine the block vector candidate. Fig. 44a is an illustration of BV adjustment for horizontal flip, and Fig. 44b is an illustration of BV adjustment for vertical flip, respectively. For example, as shown in Figs. 44a and 44b, (x_{nbr}, y_{nbr}) and (x_{cur}, y_{cur}) represent the coordinates of the center sample of the neighboring block and the current block, respectively, BV^{nbr} and BV^{cur} denotes the BV of the neighboring block and the current block, respectively. Instead of directly inheriting the BV from a neighbouring block, the horizontal component of BV^{cur} is calculated by adding a motion shift to the horizontal component of BV^{nbr} (denoted as BV^{nbr}_h) in case that the neighbouring block is

coded with a horizontal flip, i.e., $BV^{cur}_h = 2(x_{nbr} - x_{cur}) + BV^{nbr}_h$. Similarly, the vertical component of BV^{cur} is calculated by adding a motion shift to the vertical component of BV^{nbr} (denoted as BV^{nbr}_v) in case that the neighbouring block is coded with a vertical flip, i.e., $BV^{cur}_v = 2(y_{nbr} - y_{cur}) + BV^{nbr}_v$.

2.32. Intra template matching prediction

[0309] Intra template matching prediction (Intra TMP) is a special intra prediction mode that copies the best prediction block from the reconstructed part of the current frame, whose L-shaped template matches the current template. For a predefined search range, the encoder searches for the most similar template to the current template in a reconstructed part of the current frame and uses the corresponding block as a prediction block. The encoder then signals the usage of this mode, and the same prediction operation is performed at the decoder side.

[0310] The prediction signal is generated by matching the L-shaped causal neighbor of the current block with another block in a predefined search area in Fig. 45 consisting of:

- R1: current CTU
- R2: top-left CTU
- R3: above CTU
- R4: left CTU

[0311] SAD is used as a cost function.

[0312] Within each region, the decoder searches for the template that has least SAD with respect to the current one and uses its corresponding block as a prediction block.

[0313] The dimensions of all regions (SearchRange_w, SearchRange_h) are set proportional to the block dimension (BlkW, BlkH) to have a fixed number of SAD comparisons per pixel. That is:

[0314] SearchRange_w = a * BlkW

[0315] SearchRange_h = a * BlkH

[0316] Where ‘ a ’ is a constant that controls the gain/complexity trade-off. In practice, ‘ a ’ is equal to 5.

[0317] Fig. 45 shows intra template matching search area used. The Intra template matching tool is enabled for CUs with size less than or equal to 64 in width and height. This maximum CU size for Intra template matching is configurable.

[0318] The Intra template matching prediction mode is signaled at CU level through a dedicated flag when DIMD is not used for current CU.

2.33. Intra prediction fusion

[0319] Intra prediction fusion method uses multiple predictors generated from different modes/reference lines.

[0320] In sub-test a, multiple intra predictors are generated and then fused by weighted averaging. The process of deriving the predictors to be used in the fusion process is described as follows:

- 1) For angular intra prediction modes including the single mode case of TIMD and DIMD, the proposed method derives intra prediction by weighting intra predictions obtained from multiple reference lines represented as $p_{fusion} = w_0 p_{line} + w_1 p_{line+1}$, where p_{line} is the intra prediction from the default reference line and p_{line+1} is the prediction from the line above the default reference line. The weights are set as $w_0 = 3/4$ and $w_1 = 1/4$.
- 2) For TIMD mode with blending, p_{line} is used for the 1st mode ($w_0 = 1, w_1 = 0$) and p_{line+1} is used for the 2nd mode ($w_0 = 0, w_1 = 1$).
- 3) For DIMD mode with blending, the number of predictors selected for a weighted average is increased from 3 to 6.

[0321] In sub-test b, intra prediction fusion is performed on reference lines instead of prediction blocks. Two reference lines (referred to as r_{line} and r_{line+1}) are used for intra prediction fusion. ΔInt of the corresponding intra prediction angle is considered in the fusion process. Each value in the fused reference line ($r_{fusion}[i]$) is derived from:

$$r_{fusion}[i] = (3 \cdot r_{line}[i] + r_{line+1}[i + \Delta Int]) \gg 2$$

[0322] The proposed intra prediction fusion is applied to luma blocks when angular intra

mode has non-integer slope (required reference samples interpolation) and the block size is greater than 16, it is used with MRL and not applied for ISP coded blocks. In the method studied in the sub-test a, PDPC is applied for the intra prediction mode using the closest to the current block reference line.

2.34. Template-based multiple reference line intra prediction (TMRL)

[0323] The proposed TMRL mode includes the following aspects:

a) Extended reference line candidate list and intra-prediction-mode candidate list

The extended reference line candidate list used in this proposal is {1, 3, 5, 7, 12}. The restriction on the top CTU row is unchanged. The size of the intra-prediction-mode candidate list is 10. The construction of the intra-prediction-mode candidate list is similar to MPM. The differences are:

PLANAR mode is excluded from the proposed intra-prediction-mode candidate list.

DC mode is added after the 5 neighboring PUs' modes and DIMD modes if it has not been included.

The angular modes with delta angles from ± 1 to ± 4 (compared to the existing angular modes in the intra-prediction-mode candidate list) are added.

b) Construction of the TMRL candidate list

There are $5 \times 10 = 50$ combinations of the extended reference line and the allowed intra-prediction modes for a block. Since the extended reference line starts from reference line 1, the area covered by reference line 0 is used for template matching. The SAD costs over the template area (see Fig. 46) are calculated between the predictions (generated by 50 combinations) and the reconstructions. The 20 combinations with the least SAD cost are selected in an ascending order to form the TMRL candidate list.

c) Signaling of TMRL

Instead of coding the reference line and the intra mode directly, an index to the TMRL candidate list is coded to indicate which combination of the reference line and prediction mode is used for coding the current block. In the proposed TMRL mode, a truncated Golomb-Rice coding with a divisor 4 is employed to code selected combinations from the combination list. The binarization process and the codewords are shown in Table 2-8.

Table 2-8 – The binarization process of the TMRL index

Index	Bin string (prefix)	Bin string (suffix)
0	0	00
1	0	01

2	0	10
3	0	11
4	10	00
...	...	
18	1111	10
19	1111	11

d) Encoder-side modifications

Encoder-side modification is tested to further improve the coding efficiency. For intra blocks larger than 8×8 , an additional TMRL RDO is added if no TMRL modes are selected by SATD comparison.

2.35. Convolutional cross-component model (CCCM) for intra prediction

[0324] It is proposed to apply convolutional cross-component model (CCCM) to predict chroma samples from reconstructed luma samples in a similar spirit as done by the current CCLM modes. As with CCLM, the reconstructed luma samples are down-sampled to match the lower resolution chroma grid when chroma sub-sampling is used.

[0325] Also, similarly to CCLM, there is an option of using a single model or multi-model variant of CCCM. The multi-model variant uses two models, one model derived for samples above the average luma reference value and another model for the rest of the samples (following the spirit of the CCLM design). Multi-model CCCM mode can be selected for PUs which have at least 128 reference samples available.

2.35.1. Convolutional filter

[0326] The proposed convolutional 7-tap filter consist of a 5-tap plus sign shape spatial component, a nonlinear term and a bias term. The input to the spatial 5-tap component of the filter consists of a center (C) luma sample which is collocated with the chroma sample to be predicted and its above/north (N), below/south (S), left/west (W) and right/east (E) neighbors as illustrated below in Fig. 47.

[0327] The nonlinear term P is represented as power of two of the center luma sample C

and scaled to the sample value range of the content:

$$P = (C * C + \text{midVal}) \gg \text{bitDepth}$$

That is, for 10-bit content it is calculated as:

$$P = (C * C + 512) \gg 10$$

[0328] The bias term B represents a scalar offset between the input and output (similarly to the offset term in CCLM) and is set to middle chroma value (512 for 10-bit content).

Output of the filter is calculated as a convolution between the filter coefficients c_i and the input values and clipped to the range of valid chroma samples:

$$\text{predChromaVal} = c_0C + c_1N + c_2S + c_3E + c_4W + c_5P + c_6B$$

2.35.2. Calculation of filter coefficients

[0329] The filter coefficients c_i are calculated by minimising MSE between predicted and reconstructed chroma samples in the reference area. Fig. 48 illustrates the reference area which consists of 6 lines of chroma samples above and left of the PU. Reference area extends one PU width to the right and one PU height below the PU boundaries. Area is adjusted to include only available samples. The extensions to the area shown in blue are needed to support the “side samples” of the plus shaped spatial filter and are padded when in unavailable areas.

[0330] The MSE minimization is performed by calculating autocorrelation matrix for the luma input and a cross-correlation vector between the luma input and chroma output. Autocorrelation matrix is LDL decomposed and the final filter coefficients are calculated using back-substitution. The process follows roughly the calculation of the ALF filter coefficients in ECM, however LDL decomposition was chosen instead of Cholesky decomposition to avoid using square root operations. The proposed approach uses only integer arithmetic.

2.35.3. Bitstream signalling

[0331] Usage of the mode is signalled with a CABAC coded PU level flag. One new CABAC context was included to support this. When it comes to signalling, CCCM is considered a sub-mode of CCLM. That is, the CCCM flag is only signalled if intra

prediction mode is LM_CHROMA_IDX (to enable single mode CCCM) or MMLM_CHROMA_IDX (to enable multi-model CCCM).

2.36. Gradient Linear Model (GLM)

[0332] Compared with the CCLM, instead of down-sampled luma values, the GLM utilizes luma sample gradients to derive the linear model. Specifically, when the GLM is applied, the input to the CCLM process, i.e., the down-sampled luma samples L , are replaced by luma sample gradients G . The other parts of the CCLM (e.g., parameter derivation, prediction sample linear transform) are kept unchanged.

$$C = \alpha \cdot G + \beta$$

[0333] For signaling, when the CCLM mode is enabled to the current CU, two flags are signaled separately for Cb and Cr components to indicate whether GLM is enabled to each component; if the GLM is enabled for one component, one syntax element is further signaled to select one of 4 gradient filters for the gradient calculation. Four gradient filters are enabled for the GLM, as illustrated in Fig. 49.

2.37. Temporal block vector prediction

[0334] Temporal BV prediction (TBVP) is proposed to mimic TMVP. Temporal BV candidates are introduced to IBC merge/AMVP candidate lists. The IBC merge candidate list includes the regular IBC merge candidate list, the IBC-TM merge candidate list and IBC-MBVD base candidate list.

[0335] TBVP candidates are derived with full pruning from the same temporal positions as TMVP. They are put before the HMVP candidates.

2.38. Copy-Padding for IBC

[0336] Copy-padding can be applied on the overlapped area. Fig. 50 shows unreconstructed samples (shaded) in the reference block are padded by copying their prediction samples. With copy-padding, an unreconstructed sample in the overlapped region can be padded by copying its prediction sample, as shown in Fig. 50, in an equation way,

$$P'(x,y) = P(x+BVx, y+BVy)$$

wherein $P'(x,y)$ is a padded sample at position (x, y) , $P(x+BVx, y+BVy)$ is a prediction sample, and (BVx, BVy) is the BV of current block.

[0337] Copy-padding is performed only if the horizontal BV component is smaller than or equal to 0 and the vertical BV component is smaller than or equal to 0.

2.39. Direct block vector (DBV) mode for chroma prediction

[0338] In ECM-6.0, the intra prediction modes for chroma components include 6 cross component linear model (LM) modes, convolution cross component model (CCCM) mode, gradient linear model (GLM) mode, DIMD mode, direct mode (DM), and four default intra prediction modes.

In the signaling of chroma intra mode, an `intra_chroma_pred_mode` is signaled to indicate the specific coding mode, as shown in Table 2-9.

Table 2-9— The binarization process for `intra_chroma_pred_mode` in ECM6.0

<code>intra_chroma_pred_mode</code>	bin string	chroma intra mode
0	1100	list[0]
1	1101	list[1]
2	1110	list[2]
3	1111	list[3]
4	10	DIMD chroma
5	0	DM

[0339] ECM6.0 includes a method of chroma prediction using block vector in `MODE_IBC`. In single tree partition, the prediction process of IBC is applied for both luma and chroma components, and the chroma block vector is derived from the corresponding luma block vector depending on the chroma format sampling structure. In dual tree partition, the prediction process of IBC is only applied for luma component.

[0340] This contribution proposes a method to improve the coding efficiency of chroma intra prediction for screen content, namely direct block vector (DBV) mode.

[0341] Fig. 51 shows the five locations in reconstructed luma samples. For chroma components, when chroma dual tree is activated in intra slice, if one of the luma blocks (the following five locations in Fig. 51) is coded with MODE_IBC, its block vector bvL is used and scaled to derive chroma block vector bvC . The scaling factor depends on the chroma format sampling structure.

[0342] Then, by using the position of the current chroma block (x_{Cb} , y_{Cb}) and its bvC , the corresponding offset position ($x_{Cb} + bvC[0]$, $y_{Cb} + bvC[1]$) is determined, and a block copying prediction is performed as shown in Fig. 52. Fig. 52 shows the prediction process of DBV mode.

[0343] A CU level flag is signaled to indicate whether the proposed DBV mode is applied as shown in Table 2-10.

Table 2-10 – The binarization process for `intra_chroma_pred_mode` in the proposed method

<code>intra_chroma_pred_mode</code>	bin string	chroma intra mode
0	11100	list[0]
1	11101	list[1]
2	11110	list[2]
3	11111	list[3]
4	110	DIMD chroma
5	10	DM
6	0	DBV

2.40. Filtering for IBC predicted block

[0344] The proposed 6-tap filter consist of a 5-tap plus sign shape spatial component and a bias term. The input to the spatial 5-tap component of the filter consists of a center (C) sample in the reference block which is at corresponding locations with the sample in the current block to be predicted and its above/north (N), below/south (S), left/west (W) and right/east (E) neighbors as illustrated below. Fig. 53 illustrates a spatial part of the filter.

[0345] The bias term B represents a scalar offset between the input and output and is set to middle luma value (512 for 10-bit content).

[0346] Output of the filter is calculated as follows:

$$\text{predLumaVal} = c0C + c1N + c2S + c3E + c4W + c5B.$$

[0347] The filter coefficients c_i are calculated by minimising the MSE between the reference template and current template, very similar to the CCCM process.

[0348] This filtered mode is used as an additional mode for non-merge IBC blocks. For non-merge blocks, this mode cannot be applied together with IBC-LIC, IBC-CIIP or RR-IBC. For IBC merge modes, this filtering mode is inherited when merge mode list is constructed, so there is no extra signalling.

3. Problems

[0349] In current design of intra block copy (IBC) in video coding standard or video codec (e.g., VVC, ECM), only translational motion is considered within the picture. Similar to inter prediction, affine model could be used to improve the coding performance to handle the textures with zoom-in/out, rotation, perspective motions and the other irregular motions.

4. Detailed solutions

[0350] The detailed solutions below should be considered as examples to explain general concepts. These embodiments should not be interpreted in a narrow way. Furthermore, these solutions can be combined in any manner.

[0351] In the present disclosure, intra block copy (IBC) may not be limited to the current IBC technology, but may be interpreted as the technology that reference (or prediction) block is obtained with samples in the current slice/tile/subpicture/picture/other video unit (e.g., CTU row) excluding the conventional intra prediction methods.

[0352] In the following discussion, IBC may be replaced by other coding tools that rely

on coded/decoded/reconstructed information within the same region, e.g., palette, intra template matching.

[0353] In the following discussion, a block vector (BV) used in the affine model is denoted as control point block vector (CPBV).

Intra block copy with affine model (Denote the proposed method as IBC-affine mode)

1. It is proposed that a video unit may be divided into more than one subblocks, wherein the block vector (BV) for a subblock may be derived using an affine model. Denote the block size of the video unit as $W \times H$.
 - a. In one example, one affine model may be applied to all subblocks.
 - b. In one example, at least two affine models may be applied to two subblocks.
 - i. Alternatively, furthermore, the control point BVs for two subblocks associated with the affine model may be different.
 - ii. Alternatively, furthermore, each subblock may be associated with its own affine model.
 - c. In one example, the video unit may be coded with IBC mode or IntraTMP mode.
 - d. In one example, the same size may be used for all subblocks.
 - i. In one example, the size of the subblock may be $W_1 \times H_1$, wherein W_1 is less than or equal to W , and H_1 is less than or equal to H , except that $W_1 = W$ and $H_1 = H$.
 - 1) In one example, W_1 may be equal to H_1 .
 - a) For example, $W_1 = H_1 = 2$.
 - b) For example, $W_1 = H_1 = 4$.
 - c) For example, $W_1 = H_1 = 8$.
 - d) For example, $W_1 = H_1 = 16$.
 - 2) In one example, a filtering method may be applied to the boundaries of subblocks, such as OBMC.
 - 3) In one example, when the number of samples of a sub-block is larger than 1, a pixel-level refinement process may be applied.
 - ii. In one example, the size of the subblock may depend on coding information such as colour components, and/or chroma format, and/or the video resolution.
 - iii. Alternatively, the size of at least one subblock may be different from other subblocks.
 - iv. In one example, one subblock may be one pixel, i.e., $W_1 = H_1 = 1$.
 - v. In one example, the subblock partitioning may depend on the colour format/colour components.

- 1) In one example, $W_c = \max(1, W_l/xS)$ and $H_c = \max(1, H_l/yS)$, wherein W_l and H_l are width and height of a luma subblock, W_c and H_c are width and height of a chroma subblock, xS and yS are scaling factors depending on the colour format.
 - a) E.g., $xS=yS=2$ for YUV 4:2:0.
 - b) E.g., $xS=yS=1$ for YUV 4:4:4.
 - c) E.g., $xS=2, yS=1$ for YUV 4:2:2.
- vi. In one example, the BV of a chroma sub-block may be derived from at least one BV of at least one collocated luma sub-block.
 - 1) In one example, the collocated luma sub-block may refer to a luma sub-block which consists of at least one collocated luma sample of a chroma sample in the chroma sub-block.
 - 2) In one example, the collocated luma sub-block may refer to the luma sub-block consists of the collocated luma sample of the center, or top-left, or top-right, or left-bottom, or right-bottom sample of the chroma sub-block.
 - 3) In on example, BVs from the collocated luma sub-block may be weighted averaged.
 - 4) In one example, BV of the collocated luma sub-block consists of most of collocated luma samples may be used.
 - 5) In one example, BV of the collocated luma sub-block at a specific position may be used, such as center.
- e. In one example, T_1 -parameter affine model may be used.
 - i. For example, 4-parameter affine model is used (i.e., $T_1 = 4$).
 - 1) In one example, 2 CPBVs may be used in the 4-parameter affine model.
 - 2) In one example, BV for a subblock may be derived using the 4-parameter affine model as follows:
 - a)
$$\begin{cases} bv_x = \frac{bv_{1x}-bv_{0x}}{W}x + \frac{bv_{1y}-bv_{0y}}{W}y + bv_{0x} \\ bv_y = \frac{bv_{1y}-bv_{0y}}{W}x + \frac{bv_{1x}-bv_{0x}}{W}y + bv_{0y} \end{cases}, \text{ where } (bv_{0x}, bv_{0y}) \text{ is}$$

block vector of the top-left corner control point, (bv_{1x}, bv_{1y}) is
block vector of the top-right corner control point.
 - 3) Alternatively, the top-left corner control point and bottom-left corner control point may be used.
 - 4) Alternatively, the top-left corner control point and bottom-right corner control point may be used.
 - 5) Alternatively, the top-right corner control point and bottom-left corner control point may be used.

- 6) Alternatively, the top-right corner control point and bottom-right corner control point may be used.
- 7) Alternatively, the bottom-left corner control point and bottom-right corner control point may be used.
- ii. For example, 6-parameter affine model is used (i.e., $T_1 = 6$).
 - 1) In one example, 3 CPBVs may be used in the 6-parameter affine model.
 - 2) In one example, the BV for a subblock may be derived using the 6-parameter affine model as follows:
 - a)
$$\begin{cases} bv_x = \frac{bv_{1x}-bv_{0x}}{W}x + \frac{bv_{2x}-bv_{0x}}{H}y + bv_{0x} \\ bv_y = \frac{bv_{1y}-bv_{0y}}{W}x + \frac{bv_{2y}-bv_{0y}}{H}y + bv_{0y} \end{cases}, \text{ where } (bv_{0x}, bv_{0y}) \text{ is}$$

block vector of the top-left corner control point, (bv_{1x}, bv_{1y}) is block vector of the top-right corner control point, and (bv_{2x}, bv_{2y}) is block vector of the bottom-left corner control point.
 - 3) Alternatively, the top-left corner control point, and top-right corner control point, and bottom-right corner control point may be used.
 - 4) Alternatively, the top-left corner control point, and bottom-right corner control point, and bottom-left corner control point may be used.
 - 5) Alternatively, the top-right corner control point, and bottom-right corner control point, and bottom-left corner control point may be used.
 - iii. In one example, which affine model is used and/or how to use the affine mode may be pre-defined, or signalled, or derived.
 - iv. In one example, the derived BV may be rounded to a precision.
 - 1) For example, the derived BV may be rounded to integer-pixel precision.
 - 2) For example, the derived BV may be rounded to half-pixel precision.
 - 3) For example, the derived BV may be rounded to quarter-pixel precision.
 - v. In one example, the derived BV may be clipped to a region.
 - 1) In one example, the region may only include samples already reconstructed before the current block.
 - 2) In one example, the region may only include samples in specific CTUs or CTU rows.
 - 3) In one example, the region may refer to IBC buffer.
2. It is proposed that only uni-prediction affine model is applied for IBC-affine coded blocks.

- a. Alternatively, bi-prediction affine models could be applied for IBC-affine coded blocks.
 - i. In one example, each prediction direction may be associated with one affine model.
- 3. It is proposed that one or more CPBV candidate list may be constructed for a video unit coded with IBC-affine mode.
 - a. In one example, IBC-affine mode may be applied to IBC AMVP mode (IBC affine AMVP mode).
 - i. In one example, the CPBV candidate list may refer to the CPBV AMVP candidate list.
 - b. In one example, IBC-affine mode may be applied to IBC merge mode (IBC affine merge mode).
 - i. In one example, the CPBV candidate list may refer to the CPBV merge candidate list.
 - c. In one example, one or more type of CPBV candidate may be used to construct the CPBV candidate list.
 - i. In one example, the inherited CPBV candidate may be used.
 - 1) In one example, the inherited CPBV candidates may be extrapolated from the CPBVs of the neighbour video units.
 - a) In one example, the neighbouring video unit may refer to spatial adjacent, and/or non-adjacent, and/or temporal neighbouring video unit.
 - b) In one example, the extrapolation method may be same as affine for inter prediction.
 - ii. In one example, the constructed CPBV candidate may be used.
 - 1) In one example, the constructed CPBV candidate may refer to the CPBV candidate that is constructed by combining the normal block vector of neighbouring video unit of each control point.
 - a) In one example, the neighbouring video unit may refer to spatial adjacent, and/or non-adjacent, and/or temporal neighbouring video unit.
 - b) In one example, BV from HBVP table may be used to construct the CPBV candidate.
 - c) In one example, temporal BV may be used to construct the CPBV candidate.
 - iii. In one example, normal BV from spatial adjacent and/or non-adjacent neighbouring video units may be used to construct the candidate list.
 - iv. In one example, temporal normal BV may be used to construct the candidate list.

- v. In one example, normal BV from HBVP table may be used to construct the candidate list.
- vi. In one example, the normal BV from spatial adjacent and/or non-adjacent neighbouring video units, and/or temporal normal BV, and/or normal BV from HBVP table may be directly used as the candidates in the CPBV candidate list.
- vii. In one example, pairwise candidates may be used.
- viii. In one example, pre-defined candidates may be used, such as zero CPBVs.
- ix. In one example, the way to construct the CPBV candidate list for IBC AMVP mode may be same as that for IBC merge mode.
 - 1) Alternatively, the way to construct the CPBV candidate list for IBC AMVP mode may be different from that for IBC merge mode.
- d. In one example, pruning may be used during constructing the CPBV candidate list.
 - i. In one example, pruning may be performed for partial or all CPBV candidates.
 - ii. In one example, pruning may be performed when the motion information of two CPBV candidates are same.
 - 1) In one example, BVs of all control points are same.
 - 2) In one example, the motion information may include other coding information such as whether a specific coding tool is used.
 - a) In one example, the specific coding tool may refer to IBC-LIC, RR-IBC, filtering based IBC.
 - iii. In one example, pruning may be performed when the motion information of two CPBV candidates are similar.
 - 1) In one example, when the difference between BVs of one or more control points of the two CPBV candidates are smaller than or equal to a threshold (T_2), the two CPBV candidates may be treated as similar.
- e. In one example, reordering may be used during constructing the CPBV candidate list.
 - i. In one example, reordering may depend on neighbouring prediction/reconstruction samples.
 - ii. In one example, template matching cost may be used for reordering.
 - iii. In one example, one or more reordering operation may be used.
 - iv. In one example, reordering may be performed depending on coding information, such as block size/dimensions or candidate type.
- f. In one example, a method of grouping CPBV candidates may be used.
 - i. In one example, the clustering method may be used.

- ii. In one example, the CPBV candidates may be divided into different groups/classes.
 - iii. In one example, the method of grouping CPBV candidates may be used with reordering method.
- g. In one example, one or more CPBV candidates may be refined.
 - i. In one example, template matching may be used to refine the CPBV candidates.
 - ii. In one example, partial or all CPBV candidates may be refined.
- h. In one example, one or more BV offset/difference may be added to the CPBV candidates.
 - i. In one example, the BV offset/difference may be added to all CPBV candidates.
 - ii. Alternatively, there is no BV offset/difference for at least one CPBV candidate.
 - iii. In one example, the BV offset/difference may be signalled or derived.
 - 1) In one example, the sign and/or magnitude of BV offset/difference may be signalled or derived.
- 4. In one example, CPBVs of the current video unit may be used for the following video unit.
 - a. In one example, the CPBVs of the current video unit may be stored in the HBVP table.
 - i. In one example, the CPBVs may be stored using the same way as normal BV.
 - ii. In one example, the CPBVs may be stored in an individual HBVP table that is different from the HBVP table for normal BV.
 - iii. In one example, the CPBVs may be conditionally stored in the HBVP table.
 - 1) In one example, the condition may depend on coding information, such as block dimensions/size.
 - b. In one example, the parameters of the affine mode and/or positions may be stored and used by the following video unit.
 - i. In one example, the parameters of the IBC-affine mode may be used for the following video unit coded with IBC-affine.
 - ii. In one example, the parameters of the IBC-affine mode may be used for the following video unit coded with inter affine.
 - iii. the parameters of the inter affine mode may be used for the following video unit coded with IBC-affine.
 - c. In one example, the CPBVs may be used to construct the BV candidate list for the following video units.

- i. In one example, the following video units may refer to video units in the current picture/slice/tile or those in the following picture/slice/tile.
 - ii. In one example, CPBVs may be only used to construct the CPBV candidate list for the following units.
 - iii. In one example, the CPBVs from neighbouring (adjacent or non-adjacent) video units may be used.
 - d. In one example, the CPBVs of luma video unit may be used for chroma video unit.
 - i. In one example, the CPBVs may be used to derive the chroma BV for the chroma video unit.
 - ii. In one example, the CPBVs may be used to indicate the area used for training the model for CCCM.
 - e. In one example, the CPBVs may be used for video unit coded with IntraTMP.
 - f. In one example, the CPBVs may be only used for the neighbouring video units.
 - g. Alternatively, the CPBVs are not allowed to be used for the following video unit.
 - h. In one example, whether to and/or how to store/use the CPBVs may be different for the video units at the CTU boundary and video units that are not at the CTU boundary.
5. In one example, IBC-affine may be used with a specific coding tool.
- a. In one example, the coding tool may refer to intra coding tool.
 - i. In one example, intra coding tool may refer to a specific coding tool such as conventional intra prediction, DIMD, TIMD, MRL, ISP, MIP, IntraTMP, intra prediction fusion, SGPM, TMRL, PDPC/gradient PDPC, CCLM, MMLM, CCCM, GLM, chroma fusion, or their variants.
 - b. In one example, the coding tool may refer to inter coding tool.
 - i. In one example, inter coding tool may refer to a specific coding tool such as CIIP (e.g., CIIP-Planar, CIIP-TIMD, CIIP-TM), BCW (e.g., BCW index derived by TM), MMVD (e.g., MMVD or TM based reordering for MMVD), template matching (TM), affine (e.g., affine-MMVD, TM based reordering for affine MMVD), DMVR/multi-pass DMVR, PROF, BDOF/sample based BDOF, adaptive decoder-side motion vector refinement (ADMVR), OBMC or TM based OBMC, MHP, GPM (e.g., GPM, GPM-TM, GPM-MMVD, GPM-intra), bilateral/template matching AMVP-merge mode, or their variants.
 - c. In one example, the coding tool may refer to BDPCM or Palette.
 - d. In one example, the coding tool may refer to IBC coding tool.
 - i. In one example, IBC coding tool refer to IBC AMVP mode may refer to normal IBC AMVP, or TM based IBC AMVP, or RR-IBC AMVP mode, or other IBC AMVP mode wherein a BV predictor is derived and BVD

is signalled/derived, or, IBC merge mode may refer to normal IBC merge mode, or IBC-TM merge mode, or IBC-MBVD mode, or IBC-CIIP, or IBC-GPM, or IBC-LIC, or sub-pel based IBC, filtering based IBC, or their variants.

- e. In one example, the coding tool may refer to an in-loop filter method.
 - i. In one example, the boundaries between different subblocks may be processed by deblocking filter.
- f. In one example, the coding tool may refer to a transform method, such as DCT/DST/MTS/LFNST/NSPT.
- g. In one example, the coding tool may refer to sign prediction, and/or dependent quantization.
- h. In one example, when IBC-affine is used with TM based IBC AMVP, one or more CPBVs of the bi-prediction may be refined using TM.
 - i. In one example, fractional precision of BV may be derived using TM.
- i. In one example, one or more CPBVs may be refined using template matching and/or bilateral matching.
- j. In one example, when IBC-affine is used with IBC-TM merge mode, one or more BV offsets may be added to the CPBVs.
- k. In one example, when IBC-affine is used with RR-IBC AMVP mode, all CPBVs are flipped.
 - i. Alternatively, one of CPBVs may be not flipped.
- l. In one example, when IBC-affine is used with IBC-LIC/filtering based IBC, partial or all subblocks may be refined using the IBC-LIC/filtering based IBC parameters.
- m. In one example, when IBC-affine is used with IBC-CIIP, the prediction signal generated by IBC may be obtained by CPBVs.
- n. In one example, when IBC-affine is used with IBC-GPM, prediction signals of one or more sub-partitions may be obtained by IBC-affine.
- o. In one example, when IBC-affine is used with IBC-MBVD, the BV offsets may be derived for one or more CPBVs.
- p. In one example, IBC-affine may be used with sub-pel/fractional based IBC.
 - i. In one example, one or more CPBVs may be sub-pel/fractional precision.
 - ii. In one example, all CPBVs may be sub-pel/fractional.
 - iii. In one example, at most one CPBV is allowed to be sub-pel/fractional.
 - iv. In one example, the derived BV for each subblock may be fractional precision or integer.
 - v. In one example, the sub-pel/fractional precision may refer to 1/2, 1/4, 1/8, 1/16, 1/32, 1/64.

- vi. In one example, the interpolation for fractional BV in IBC-affine may be same as that for non IBC-affine.
 - 1) Alternatively, the interpolation for fractional BV in IBC-affine may be different from that for non IBC-affine.
- q. Alternatively, IBC-affine is not allowed to be used with a specific coding tool.
 - i. In one example, the coding tool may refer to IBC AMVP mode may refer to normal IBC AMVP, or TM based IBC AMVP, or RR-IBC AMVP mode, or other IBC AMVP mode wherein a BV predictor is derived and BVD is signalled/derived, or, IBC merge mode may refer to normal IBC merge mode, or IBC-TM merge mode, or IBC-MBVD mode, or IBC-CIIP, or IBC-GPM, or IBC-LIC, or sub-pel based IBC, filtering based IBC, or their variants.
- 6. Whether to and/or how to apply IBC-affine may depend on coding information, the coding information may refer to:
 - a. whether a specific coding method is allowed,
 - b. block dimensions and/or block size,
 - c. block depth,
 - d. slice/picture type and/or partition tree type (single, or dual tree, or local dual tree),
 - e. temporal layer identification,
 - f. block location,
 - g. video content (e.g., camera-captured content, and/or screen content, and/or mixed content),
 - h. colour format,
 - i. colour component.
 - i. In one example, IBC-affine may apply to all colour components.
 - ii. In one example, when IBC-affine is applied to chroma components, it may be different from that for luma component.
 - iii. In one example, whether to and/or how to apply IBC-affine to a first component may depend on whether to apply IBC-affine to a second component.
 - 1) In one example, the first component may refer to chroma component (e.g., Cb and/or Cr), and the second component may refer to luma component (e.g., Y).
 - 2) In one example, the way to apply IBC-affine to the first component may be same as the second component.
 - a) Alternatively, the way to apply IBC-affine to the first component may be different from the second component.
 - iv. In one example, IBC-affine may apply to luma component, but not to chroma components.

- 1) In one example, luma component may refer to Y in YCbCr colour space or G in RGB colour space.
 - 2) In one example, chroma components may refer to Cb and/or Cr in YCbCr colour space or R and/or B in RGB colour space.
7. Indication of IBC-affine may be conditionally signalled wherein the condition may include:
- a. block dimensions and/or block size,
 - b. block depth,
 - c. slice/picture type and/or partition tree type (single, or dual tree, or local dual tree),
 - d. temporal layer identification,
 - e. block location,
 - f. colour format,
 - g. colour component.
8. Whether current block is coded with IBC-affine mode may be signalled using one or more syntax elements.
- a. In one example, the syntax element may be binarized with fixed length coding, or truncated unary coding, or unary coding, or EG coding, or coded a flag.
 - b. In one example, the syntax element may be bypass coded or context coded.
 - i. The context may depend on coded information, such as block dimensions, and/or block size, and/or slice/picture types, and/or the information of neighbouring blocks (adjacent or non-adjacent), and/or the information of other coding tools used for current block, and/or the information of temporal layer.
 - c. In one example, the one or more syntax elements may be signalled at sequence header/picture header/SPS/VPS/DPS/DCI/PPS/APS/slice header/tile group header.
 - d. In one example, the syntax element may be coded in a predictive way.
 - e. For example, the syntax element of the current block may be predicted by that of a neighboring block.
9. Whether the disclosed method is used may be dependent on the content characteristics, e.g., screen contents or nature contents.
10. The disclosed methods above may apply to other coding tools, such as IntraTMP.

General aspects

11. In above examples, the video unit may refer to the video unit may refer to colour component/sub-picture/slice/tile/coding tree unit (CTU)/CTU row/groups of CTU/coding unit (CU)/prediction unit (PU)/transform unit (TU)/ coding tree block (CTB)/ coding block (CB)/prediction block(PB)/transform block (TB)/a block/sub-block of a block/sub-region within a block/ any other region that contains more than one sample or pixel.

12. Whether to and/or how to apply the disclosed methods above may be signalled at sequence level/group of pictures level/picture level/slice level/tile group level, such as in sequence header/picture header/SPS/VPS/DPS/DCI/PPS/APS/slice header/tile group header.
13. Whether and/or how to apply the above methods may depend on the following information:
 - a. A message signalled in the DPS/SPS/VPS/PPS/APS/picture header/slice header/tile group header/coding tree unit (CTU)/Coding unit (CU)/CTU row/group of CTUs/TU/PU block/Video coding unit;
 - b. Position of CU/PU/TU/block/Video coding unit;
 - c. Block dimension of current block and/or its neighbouring blocks;
 - d. Block shape of current block and/or its neighbouring blocks;
 - e. coded mode of a block, e.g., IBC or non-IBC inter mode or non-IBC subblock mode;
 - f. Indication of the colour format (such as 4:2:0, 4:4:4);
 - g. Coding tree structure;
 - h. Slice/tile group type and/or picture type;
 - i. Colour component (e.g., may be only applied on chroma components or luma component);
 - j. Temporal layer ID;
 - k. Profiles/Levels/Tiers of a standard.
14. A syntax element disclosed above may be binarized as a flag, a fixed length code, an EG(x) code, a unary code, a truncated unary code, a truncated binary code, etc. It can be signed or unsigned.
15. A syntax element disclosed above may be coded with at least one context model. Or it may be bypass coded.
16. A syntax element disclosed above may be signaled in a conditional way.
 - a. The SE is signaled only if the corresponding function is applicable.
 - b. The SE is signaled only if the dimensions (width and/or height) of the block satisfy a condition.
17. A syntax element disclosed above may be signaled at block level/ sequence level/group of pictures level/picture level/slice level/tile group level, such as in coding structures of CTU/CU/TU/PU/CTB/CB/TB/PB, or sequence header/picture header/SPS/VPS/DPS/DCI/PPS/APS/slice header/tile group header.
18. The proposed method(s) may be combined with another coding tool such as affine/MTS/LFNST/MMVD/MIP/ISP/CCLM/CCCM/SMVD/BDOF/DMVR/HMVP/T emplate Matching/IBC/Palette/etc.
19. The proposed method(s) may be excluded with another coding tool such as affine/MTS/LFNST/MMVD/MIP/ISP/CCLM/CCCM/SMVD/BDOF/DMVR/HMVP/T emplate Matching/IBC/Palette/etc.

- a. In one example, if the proposed method(s) is used, the excluded coding tool is disabled implicitly without signaling.
- b. In one example, if the excluded coding tool is used, the proposed method(s) is disabled implicitly without signaling.

[0354] As used herein, the term “video unit” or “video block” may be a sequence, a picture, a slice, a tile, a brick, a subpicture, a coding tree unit (CTU)/coding tree block (CTB), a CTU/CTB row, one or multiple coding units (CUs)/coding blocks (CBs), one or multiple CTUs/CTBs, one or multiple Virtual Pipeline Data Unit (VPDU), a sub-region within a picture/slice/tile/brick.

[0355] Fig. 54 illustrates a flowchart of a method 5400 for video processing in accordance with embodiments of the present disclosure. The method 5400 is implemented during a conversion between a video unit of a video and a bitstream of the video.

[0356] At block 5410, for a conversion between a video unit of a video and a bitstream of the video, a block vector (BV) for at least one subblock of a plurality of subblocks of the video unit is derived using an affine model. In this case, in some embodiments, a same block size is used for all of the plurality of subblocks. Alternatively, or in addition, at least one of: a parameter of an affine mode or a position of the video unit is applied to a following video unit of the video unit.

[0357] At block 5420, the conversion is performed based on the BV. In some embodiments, the conversion may comprise encoding the video unit into the bitstream. Alternatively, the conversion may comprise decoding the video unit from the bitstream.

[0358] The method 5400 enables the affine model to be used to improve the coding performance to handle the textures with zoom-in/out, rotation, perspective motions and the other irregular motions.

[0359] In some embodiments, a filtering approach may be applied to boundaries of the plurality of subblocks. For example, the filtering approach may be overlapped block motion compensation (OBMC). In some other embodiments, if the number of samples of

a subblock is larger than a threshold, a pixel-level refinement process may be applied to the subblock. In this case, the threshold equals to 1.

[0360] In some embodiments, a block size of one of the plurality subblocks may be one pixel. In this case, the block size of one of the plurality subblocks is $W \times H$, and $W=H=1$, W presents a width of the subblock and H represents a height of the subblock. In some other embodiments, a subblock partitioning of the plurality subblocks may depend on at least one of: colour format, or colour components. In some examples, W_c may equal to a maximum of 1 and W_l/xS , and H_c may equal to a maximum of 1 and H_l/yS . In this case, W_l represents a width of a luma subblock, H_l represents a height of the luma subblock, W_c represents a width of a chroma subblock, H_c represents a height of the chroma subblock, and xS and yS represent scaling factors depending on the colour format. For example, for YUV 4:2:0 of the colour format, $xS=yS=2$. In some examples, for YUV 4:4:4 of the colour format, $xS=yS=1$. In some examples, for YUV 4:2:2 of the colour format, $xS=2$ and $yS=1$.

[0361] In some embodiments, a BV of a chroma sub-block may be derived from at least one BV of at least one collocated luma sub-block. In some examples, the at least one collocated luma sub-block may comprise a luma sub-block. In this case, the luma sub-block may comprise at least one collocated luma sample of a chroma sample in the chroma sub-block. In some embodiments, the at least one collocated luma sub-block may comprise a luma sub-block. In this case, the luma sub-block may comprise at least one of: a collocated luma sample of center of the chroma sub-block, a collocated luma sample of top-left of the chroma sub-block, a collocated luma sample of top-right of the chroma sub-block, a collocated luma sample of left-bottom of the chroma sub-block, or a collocated luma sample of right-bottom sample of the chroma sub-block. In some other embodiments, the at least one BV of the at least one collocated luma sub-block may be weighted averaged. In some embodiments, the at least one BV of the at least one collocated luma sub-block including most of collocated luma samples may be used. Alternatively, the at least one BV of the at least one collocated luma sub-block at a position may be used. For

example, the position may be center.

[0362] In some embodiments, a derived BV may be rounded to a precision. For example, the precision may be at least one of: integer-pixel precision, half-pixel precision, or quarter-pixel precision. In some other embodiments, a derived BV may be clipped to a region. For example, the region may comprise a sample already reconstructed before a current block. In some examples, the region may comprise at least one of: a sample in a coding tree unit (CTU), or a sample in CTU row. For example, the region may comprise an IBC buffer.

[0363] In some embodiments, the parameter of an IBC-affine mode may be applied by the following video unit coded with IBC-affine. In some other embodiments, the parameter of an IBC-affine mode may be applied by the following video unit coded with inter affine. Alternatively, a parameter of an inter affine mode may be applied by the following video unit coded with IBC-affine.

[0364] In some embodiments, an IBC-affine may be used with a coding tool, and the coding tool may comprise an in-loop filter approach. In some examples, a boundary between different subblocks may be processed by a deblocking filter. In some other embodiments, an IBC-affine may be used with a coding tool, and the coding tool may comprise a transform approach. For example, the transform approach may comprise at least one of: discrete cosine transform (DCT), discrete sine transform (DST), multiple transform selection (MTS), low-frequency non-separable transform (LFNST), or non-separable primary transform (NSPT).

[0365] In some embodiments, an IBC-affine may be used with a coding tool, and the coding tool may comprise at least one of: sign prediction, or dependent quantization. In some other embodiments, one or more control point block vectors (CPBVs) may be refined by at least one of: template matching, or bilateral matching.

[0366] In some embodiments, an indication of whether to and/or how to apply IBC-affine may depend on coding information. In this case, the coding information may comprise

video content. For example, the video content may comprise at least one of: camera-captured content, screen content, or mixed content.

[0367] In some embodiments, the video unit may comprise at least one of the followings: a colour component, a sub-picture, a slice, a tile, a coding tree unit (CTU), a CTU row, a group of CTU, a coding unit (CU), a prediction unit (PU), a transform unit (TU), a coding tree block (CTB), a coding block (CB), a prediction block (PB), a transform block (TB), a block, a sub-block of a block, a sub-region within a block, or a region containing more than one sample or pixel.

[0368] In some embodiments, an indication of whether to and/or how to derive the BV of the at least one subblock using the affine model may be indicated at one of the followings: sequence level, group of pictures level, picture level, slice level, or tile group level.

[0369] In some embodiments, an indication of whether to and/or how to derive the BV of the at least one subblock using the affine model may be indicated in one of the followings: a sequence header, a picture header, a sequence parameter set (SPS), a video parameter set (VPS), a dependency parameter set (DPS), a decoding capability information (DCI), a picture parameter set (PPS), an adaptation parameter sets (APS), a slice header, or a tile group header.

[0370] In some embodiments, whether to and/or how to derive the BV of the at least one subblock using the affine model may depend on at least one of the followings: a message indicated in one of: DPS, SPS, VPS, PPS, APS, picture header, slice header, tile group header, coding tree unit (CTU), coding unit (CU), CTU row, group of CTUs, TU, PU block, video coding unit, a position of a CU block, a position of a PU block, a position of a TU block, a position of a video coding unit, block dimension of a current block, block dimension of a neighbouring block of the current block, a coded mode of a block, an indication of a colour format, a coding tree structure, a slice group type, a tile group type, a slice picture type, a tile picture type, a colour component, an ID of a temporal layer, a

profile of a standard, a level of a standard, or a tier of a standard. In some embodiments, the coded mode of the block may be at least one of: IBC inter mode, non-IBC inter mode, or non-IBC subblock mode. In some embodiments, the indication of a colour format may be 4:2:0 or 4:4:4. In some other embodiments, the colour component may be applied on a chroma component or a luma component.

[0371] In some embodiments, the syntax element may be binarized as a flag, a fixed length code, an EG(x) code, a unary code, a truncated unary code, or a truncated binary code. In this case, the syntax element may be signed or unsigned. In some other embodiments, the syntax element may be coded with at least one context model, or bypass coded. In some embodiments, the syntax element may be signaled in at least one of following conditions: if a corresponding function is applicable, or if dimensions of the block satisfy a condition. In some embodiments, the dimensions may be width and/or height.

[0372] In some embodiments, the syntax element may be signaled at one of the followings: a block level, a sequence level, a group of pictures level, a picture level, a slice level, or a tile group level.

[0373] In some embodiments, the syntax element may be signaled in one of the followings: a coding structure of CTU, a coding structure of CU, a coding structure of TU, a coding structure of PU, a coding structure of CTB, a coding structure of CB, a coding structure of TB, a coding structure of PB, a sequence header, a picture header, a sequence parameter set (SPS), a video parameter set (VPS), a dependency parameter set (DPS), a decoding capability information (DCI), a picture parameter set (PPS), an adaptation parameter sets (APS), a slice header, or a tile group header.

[0374] In some embodiments, IBC with an affine model may be combined with at least one of the following coding tools: affine, multiple transform selection (MTS), low-frequency non-separable transform (LFNST), merge mode with motion vector difference (MMVD), MIP, ISP, cross-component linear model (CCLM), convolutional cross-

component model (CCCM), symmetric motion vector differences (SMVD), bi-directional optical flow (BDOF), decoder side motion vector refinement (DMVR), history-based MVP (HMVP), template matching, intra block copy (IBC), or Palette. In some other embodiments, IBC with an affine model may be excluded with at least one of the following coding tools: affine, multiple transform selection (MTS), LFNST, merge mode with motion vector difference (MMVD), MIP, ISP, cross-component linear model (CCLM), convolutional cross-component model (CCCM), SMVD, bi-directional optical flow (BDOF), decoder side motion vector refinement (DMVR), history-based MVP (HMVP), template matching, intra block copy (IBC), or Palette. In some embodiments, if the IBC with an affine model is used, the excluded coding tool may be disabled without signaling. Alternatively, if the excluded coding tool is used, the IBC with an affine model may be disabled without signaling.

[0375] According to further embodiments of the present disclosure, a non-transitory computer-readable recording medium is provided. The non-transitory computer-readable recording medium stores a bitstream of a video which is generated by a method performed by an apparatus for video processing. The method comprises: deriving a block vector (BV) for at least one subblock of a plurality of subblocks of a video unit of the video using an affine model, wherein a same block size is used for all of the plurality of subblocks, and/or at least one of: a parameter of an affine mode or a position of the video unit is applied to a following video unit of the video unit; and generating the bitstream based on the BV.

[0376] According to still further embodiments of the present disclosure, a method for storing bitstream of a video is provided. The method comprises: deriving a block vector (BV) for at least one subblock of a plurality of subblocks of a video unit of the video using an affine model, wherein a same block size is used for all of the plurality of subblocks, and/or at least one of: a parameter of an affine mode or a position of the video unit is applied to a following video unit of the video unit; generating the bitstream based on the BV; and storing the bitstream in a non-transitory computer-readable recording medium.

[0377] Implementations of the present disclosure can be described in view of the following clauses, the features of which can be combined in any reasonable manner.

[0378] Clause 1. A method for video processing, comprising: deriving, for a conversion between a video unit of a video and a bitstream of the video, a block vector (BV) for at least one subblock of a plurality of subblocks of the video unit using an affine model, wherein a same block size is used for all of the plurality of subblocks, and/or at least one of: a parameter of an affine mode or a position of the video unit is applied to a following video unit of the video unit; and performing the conversion based on the BV.

[0379] Clause 2. The method of clause 1, wherein a filtering approach is applied to boundaries of the plurality of subblocks.

[0380] Clause 3. The method of clause 2, wherein the filtering approach is overlapped block motion compensation (OBMC).

[0381] Clause 4. The method of clause 1, wherein if the number of samples of a subblock is larger than a threshold, a pixel-level refinement process is applied to the subblock, wherein the threshold equals to 1.

[0382] Clause 5. The method of clause 1, wherein a block size of one of the plurality subblocks is one pixel, wherein the block size of one of the plurality subblocks is WH , and $W=H=1$, W presents a width of the subblock and H represents a height of the subblock.

[0383] Clause 6. The method of clause 1, wherein a subblock partitioning of the plurality subblocks depends on at least one of: colour format, or colour components.

[0384] Clause 7. The method of clause 6, wherein W_c equals to a maximum of 1 and W_l/x_S , and H_c equals to a maximum of 1 and H_l/y_S , wherein W_l represents a width of a luma subblock, H_l represents a height of the luma subblock, W_c represents a width of a chroma subblock, H_c represents a height of the chroma subblock, and x_S and y_S represent scaling factors depending on the colour format.

[0385] Clause 8. The method of clause 7, wherein for YUV 4:2:0 of the colour format,

$xS=yS=2$.

[0386] Clause 9. The method of clause 7, wherein for YUV 4:4:4 of the colour format, $xS=yS=1$.

[0387] Clause 10. The method of clause 7, wherein for YUV 4:2:2 of the colour format, $xS=2$ and $yS=1$.

[0388] Clause 11. The method of clause 1, wherein a BV of a chroma sub-block is derived from at least one BV of at least one collocated luma sub-block.

[0389] Clause 12. The method of clause 11, wherein the at least one collocated luma sub-block comprises a luma sub-block, wherein the luma sub-block comprises at least one collocated luma sample of a chroma sample in the chroma sub-block.

[0390] Clause 13. The method of clause 11, wherein the at least one collocated luma sub-block comprises a luma sub-block, wherein the luma sub-block comprises at least one of: a collocated luma sample of center of the chroma sub-block, a collocated luma sample of top-left of the chroma sub-block, a collocated luma sample of top-right of the chroma sub-block, a collocated luma sample of left-bottom of the chroma sub-block, or a collocated luma sample of right-bottom sample of the chroma sub-block.

[0391] Clause 14. The method of clause 11, wherein the at least one BV of the at least one collocated luma sub-block is weighted averaged.

[0392] Clause 15. The method of clause 11, wherein the at least one BV of the at least one collocated luma sub-block including most of collocated luma samples is used.

[0393] Clause 16. The method of clause 11, wherein the at least one BV of the at least one collocated luma sub-block at a position is used.

[0394] Clause 17. The method of clause 11, wherein the position is center.

[0395] Clause 18. The method of clause 1, wherein a derived BV is rounded to a precision.

[0396] Clause 19. The method of clause 18, wherein the precision is at least one of:

integer-pixel precision, half-pixel precision, or quarter-pixel precision.

[0397] Clause 20. The method of clause 1, wherein a derived BV is clipped to a region.

[0398] Clause 21. The method of clause 20, wherein the region comprises a sample already reconstructed before a current block.

[0399] Clause 22. The method of clause 20, wherein the region comprises at least one of: a sample in a coding tree unit (CTU), or a sample in CTU row.

[0400] Clause 23. The method of clause 20, wherein the region comprises an IBC buffer.

[0401] Clause 24. The method of clause 1, wherein the parameter of an IBC-affine mode is applied by the following video unit coded with IBC-affine.

[0402] Clause 25. The method of clause 1, wherein the parameter of an IBC-affine mode is applied by the following video unit coded with inter affine.

[0403] Clause 26. The method of clause 1, wherein a parameter of an inter affine mode is applied by the following video unit coded with IBC-affine.

[0404] Clause 27. The method of clause 1, wherein an IBC-affine is used with a coding tool, and the coding tool comprises an in-loop filter approach.

[0405] Clause 28. The method of clause 27, wherein a boundary between different subblocks is processed by a deblocking filter.

[0406] Clause 29. The method of clause 1, wherein an IBC-affine is used with a coding tool, and the coding tool comprises a transform approach.

[0407] Clause 30. The method of clause 29, wherein the transform approach comprises at least one of: discrete cosine transform (DCT), discrete sine transform (DST), multiple transform selection (MTS), low-frequency non-separable transform (LFNST), or non-separable primary transform (NSPT).

[0408] Clause 31. The method of clause 1, wherein an IBC-affine is used with a coding tool, and the coding tool comprises at least one of: sign prediction, or dependent

quantization.

[0409] Clause 32. The method of clause 1, wherein one or more control point block vectors (CPBVs) are refined by at least one of: template matching, or bilateral matching.

[0410] Clause 33. The method of clause 1, wherein an indication of whether to and/or how to apply IBC-affine depends on coding information, wherein the coding information comprises video content.

[0411] Clause 34. The method of any of clauses 1-33, wherein the video unit comprises at least one of the followings: a colour component, a sub-picture, a slice, a tile, a coding tree unit (CTU), a CTU row, a group of CTU, a coding unit (CU), a prediction unit (PU), a transform unit (TU), a coding tree block (CTB), a coding block (CB), a prediction block (PB), a transform block (TB), a block, a sub-block of a block, a sub-region within a block, a region containing more than one sample or pixel.

[0412] Clause 35. The method of any of clauses 1-33, wherein an indication of whether to and/or how to derive the BV of the at least one subblock using the affine model is indicated at one of the followings: sequence level, group of pictures level, picture level, slice level, or tile group level.

[0413] Clause 36. The method of any of clauses 1-33, wherein an indication of whether to and/or how to derive the BV of the at least one subblock using the affine model is indicated in one of the followings: a sequence header, a picture header, a sequence parameter set (SPS), a video parameter set (VPS), a dependency parameter set (DPS), a decoding capability information (DCI), a picture parameter set (PPS), an adaptation parameter sets (APS), a slice header, or a tile group header.

[0414] Clause 37. The method of any of clauses 1-33, wherein whether to and/or how to derive the BV of the at least one subblock using the affine model depends on at least one of the followings: a message indicated in one of: DPS, SPS, VPS, PPS, APS, picture header, slice header, tile group header, coding tree unit (CTU), coding unit (CU), CTU

row, group of CTUs, TU, PU block, video coding unit, a position of a CU block, a position of a PU block, a position of a TU block, a position of a video coding unit, block dimension of a current block, block dimension of a neighbouring block of the current block, a coded mode of a block, an indication of a colour format, a coding tree structure, a slice group type, a tile group type, a slice picture type, a tile picture type, a colour component, an ID of a temporal layer, a profile of a standard, a level of a standard, or a tier of a standard.

[0415] Clause 38. The method of clause 37, wherein the coded mode of the block is at least one of: IBC inter mode, non-IBC inter mode, or non-IBC subblock mode.

[0416] Clause 39. The method of clause 37, wherein the indication of a colour format is 4:2:0 or 4:4:4.

[0417] Clause 40. The method of clause 37, wherein the colour component is applied on a chroma component or a luma component.

[0418] Clause 41. The method of any of clauses 1-33, wherein the syntax element is binarized as a flag, a fixed length code, an EG(x) code, a unary code, a truncated unary code, or a truncated binary code.

[0419] Clause 42. The method of any of clauses 1-33, wherein the syntax element is signed or unsigned.

[0420] Clause 43. The method of any of clauses 1-33, wherein the syntax element is coded with at least one context model, or bypass coded.

[0421] Clause 44. The method of any of clauses 1-33, wherein the syntax element is signaled in at least one of following conditions: if a corresponding function is applicable, or if dimensions of the block satisfy a condition.

[0422] Clause 45. The method of clause 44, wherein the dimensions are width and/or height.

[0423] Clause 46. The method of any of clauses 1-33, wherein the syntax element is

signaled at one of the followings: a block level, a sequence level, a group of pictures level, a picture level, a slice level, or a tile group level.

[0424] Clause 47. The method of any of clauses 1-33, wherein the syntax element is signaled in one of the followings: a coding structure of CTU, a coding structure of CU, a coding structure of TU, a coding structure of PU, a coding structure of CTB, a coding structure of CB, a coding structure of TB, a coding structure of PB, a sequence header, a picture header, a sequence parameter set (SPS), a video parameter set (VPS), a dependency parameter set (DPS), a decoding capability information (DCI), a picture parameter set (PPS), an adaptation parameter sets (APS), a slice header, or a tile group header.

[0425] Clause 48. The method of any of clauses 1-33, wherein IBC with an affine model is combined with at least one of the following coding tools: affine, multiple transform selection (MTS), low-frequency non-separable transform (LFNST), merge mode with motion vector difference (MMVD), MIP, ISP, cross-component linear model (CCLM), convolutional cross-component model (CCCM), symmetric motion vector differences (SMVD), bi-directional optical flow (BDOF), decoder side motion vector refinement (DMVR), history-based MVP (HMVP), template matching, intra block copy (IBC), or Palette.

[0426] Clause 49. The method of any of clauses 1-33, wherein IBC with an affine model is excluded with at least one of the following coding tools: affine, multiple transform selection (MTS), LFNST, merge mode with motion vector difference (MMVD), MIP, ISP, cross-component linear model (CCLM), convolutional cross-component model (CCCM), SMVD, bi-directional optical flow (BDOF), decoder side motion vector refinement (DMVR), history-based MVP (HMVP), template matching, intra block copy (IBC), or Palette.

[0427] Clause 50. The method of clause 49, wherein if the IBC with an affine model is used, the excluded coding tool is disabled without signaling.

[0428] Clause 51. The method of clause 49, wherein if the excluded coding tool is used, the IBC with an affine model is disabled without signaling.

[0429] Clause 52. The method of any of clauses 1-51, wherein the conversion includes encoding the video unit into the bitstream.

[0430] Clause 53. The method of any of clauses 1-51, wherein the conversion includes decoding the video unit from the bitstream.

[0431] Clause 54. An apparatus for video processing comprising a processor and a non-transitory memory with instructions thereon, wherein the instructions upon execution by the processor, cause the processor to perform a method in accordance with any of clauses 1-53.

[0432] Clause 55. A non-transitory computer-readable storage medium storing instructions that cause a processor to perform a method in accordance with any of clauses 1-53.

[0433] Clause 56. A non-transitory computer-readable recording medium storing a bitstream of a video which is generated by a method performed by an apparatus for video processing, wherein the method comprises: deriving a block vector (BV) for at least one subblock of a plurality of subblocks of a video unit of the video using an affine model, wherein a same block size is used for all of the plurality of subblocks, and/or at least one of: a parameter of an affine mode or a position of the video unit is applied to a following video unit of the video unit; and generating the bitstream based on the BV.

[0434] Clause 57. A method for storing a bitstream of a video, comprising: deriving a block vector (BV) for at least one subblock of a plurality of subblocks of a video unit of the video using an affine model, wherein a same block size is used for all of the plurality of subblocks, and/or at least one of: a parameter of an affine mode or a position of the video unit is applied to a following video unit of the video unit; generating the bitstream based on the BV; and storing the bitstream in a non-transitory computer-readable

recording medium.

Example Device

[0435] Fig. 55 illustrates a block diagram of a computing device 5500 in which various embodiments of the present disclosure can be implemented. The computing device 5500 may be implemented as or included in the source device 110 (or the video encoder 114 or 200) or the destination device 120 (or the video decoder 124 or 300).

[0436] It would be appreciated that the computing device 5500 shown in Fig. 55 is merely for purpose of illustration, without suggesting any limitation to the functions and scopes of the embodiments of the present disclosure in any manner.

[0437] As shown in Fig. 55, the computing device 5500 includes a general-purpose computing device 5500. The computing device 5500 may at least comprise one or more processors or processing units 5510, a memory 5520, a storage unit 5530, one or more communication units 5540, one or more input devices 5550, and one or more output devices 5560.

[0438] In some embodiments, the computing device 5500 may be implemented as any user terminal or server terminal having the computing capability. The server terminal may be a server, a large-scale computing device or the like that is provided by a service provider. The user terminal may for example be any type of mobile terminal, fixed terminal, or portable terminal, including a mobile phone, station, unit, device, multimedia computer, multimedia tablet, Internet node, communicator, desktop computer, laptop computer, notebook computer, netbook computer, tablet computer, personal communication system (PCS) device, personal navigation device, personal digital assistant (PDA), audio/video player, digital camera/video camera, positioning device, television receiver, radio broadcast receiver, E-book device, gaming device, or any combination thereof, including the accessories and peripherals of these devices, or any combination thereof. It would be contemplated that the computing device 5500 can support any type of interface to a user (such as “wearable” circuitry and the like).

[0439] The processing unit 5510 may be a physical or virtual processor and can implement various processes based on programs stored in the memory 5520. In a multi-processor system, multiple processing units execute computer executable instructions in parallel so as to improve the parallel processing capability of the computing device 5500. The processing unit 5510 may also be referred to as a central processing unit (CPU), a microprocessor, a controller or a microcontroller.

[0440] The computing device 5500 typically includes various computer storage medium. Such medium can be any medium accessible by the computing device 5500, including, but not limited to, volatile and non-volatile medium, or detachable and non-detachable medium. The memory 5520 can be a volatile memory (for example, a register, cache, Random Access Memory (RAM)), a non-volatile memory (such as a Read-Only Memory (ROM), Electrically Erasable Programmable Read-Only Memory (EEPROM), or a flash memory), or any combination thereof. The storage unit 5530 may be any detachable or non-detachable medium and may include a machine-readable medium such as a memory, flash memory drive, magnetic disk or another other media, which can be used for storing information and/or data and can be accessed in the computing device 5500.

[0441] The computing device 5500 may further include additional detachable/non-detachable, volatile/non-volatile memory medium. Although not shown in Fig. 55, it is possible to provide a magnetic disk drive for reading from and/or writing into a detachable and non-volatile magnetic disk and an optical disk drive for reading from and/or writing into a detachable non-volatile optical disk. In such cases, each drive may be connected to a bus (not shown) via one or more data medium interfaces.

[0442] The communication unit 5540 communicates with a further computing device via the communication medium. In addition, the functions of the components in the computing device 5500 can be implemented by a single computing cluster or multiple computing machines that can communicate via communication connections. Therefore, the computing device 5500 can operate in a networked environment using a logical connection

with one or more other servers, networked personal computers (PCs) or further general network nodes.

[0443] The input device 5550 may be one or more of a variety of input devices, such as a mouse, keyboard, tracking ball, voice-input device, and the like. The output device 5560 may be one or more of a variety of output devices, such as a display, loudspeaker, printer, and the like. By means of the communication unit 5540, the computing device 5500 can further communicate with one or more external devices (not shown) such as the storage devices and display device, with one or more devices enabling the user to interact with the computing device 5500, or any devices (such as a network card, a modem and the like) enabling the computing device 5500 to communicate with one or more other computing devices, if required. Such communication can be performed via input/output (I/O) interfaces (not shown).

[0444] In some embodiments, instead of being integrated in a single device, some or all components of the computing device 5500 may also be arranged in cloud computing architecture. In the cloud computing architecture, the components may be provided remotely and work together to implement the functionalities described in the present disclosure. In some embodiments, cloud computing provides computing, software, data access and storage service, which will not require end users to be aware of the physical locations or configurations of the systems or hardware providing these services. In various embodiments, the cloud computing provides the services via a wide area network (such as Internet) using suitable protocols. For example, a cloud computing provider provides applications over the wide area network, which can be accessed through a web browser or any other computing components. The software or components of the cloud computing architecture and corresponding data may be stored on a server at a remote position. The computing resources in the cloud computing environment may be merged or distributed at locations in a remote data center. Cloud computing infrastructures may provide the services through a shared data center, though they behave as a single access point for the users. Therefore, the cloud computing architectures may be used to provide the

components and functionalities described herein from a service provider at a remote location. Alternatively, they may be provided from a conventional server or installed directly or otherwise on a client device.

[0445] The computing device 5500 may be used to implement video encoding/decoding in embodiments of the present disclosure. The memory 5520 may include one or more video coding modules 5525 having one or more program instructions. These modules are accessible and executable by the processing unit 5510 to perform the functionalities of the various embodiments described herein.

[0446] In the example embodiments of performing video encoding, the input device 5550 may receive video data as an input 5570 to be encoded. The video data may be processed, for example, by the video coding module 5525, to generate an encoded bitstream. The encoded bitstream may be provided via the output device 5560 as an output 5580.

[0447] In the example embodiments of performing video decoding, the input device 5550 may receive an encoded bitstream as the input 5570. The encoded bitstream may be processed, for example, by the video coding module 5525, to generate decoded video data. The decoded video data may be provided via the output device 5560 as the output 5580.

[0448] While this disclosure has been particularly shown and described with references to preferred embodiments thereof, it will be understood by those skilled in the art that various changes in form and details may be made therein without departing from the spirit and scope of the present application as defined by the appended claims. Such variations are intended to be covered by the scope of this present application. As such, the foregoing description of embodiments of the present application is not intended to be limiting.

I/We Claim:

1. A method for video processing, comprising:
deriving, for a conversion between a video unit of a video and a bitstream of the video, a block vector (BV) for at least one subblock of a plurality of subblocks of the video unit using an affine model, wherein a same block size is used for all of the plurality of subblocks, and/or at least one of: a parameter of an affine mode or a position of the video unit is applied to a following video unit of the video unit; and
performing the conversion based on the BV.
2. The method of claim 1, wherein a filtering approach is applied to boundaries of the plurality of subblocks.
3. The method of claim 2, wherein the filtering approach is overlapped block motion compensation (OBMC).
4. The method of claim 1, wherein if the number of samples of a subblock is larger than a threshold, a pixel-level refinement process is applied to the subblock, wherein the threshold equals to 1.
5. The method of claim 1, wherein a block size of one of the plurality subblocks is one pixel, wherein the block size of one of the plurality subblocks is WH , and $W=H=1$, W presents a width of the subblock and H represents a height of the subblock.
6. The method of claim 1, wherein a subblock partitioning of the plurality subblocks depends on at least one of: colour format, or colour components.
7. The method of claim 6, wherein W_c equals to a maximum of 1 and W_l/xS , and H_c equals to a maximum of 1 and H_l/yS , wherein W_l represents a width of a luma subblock, H_l represents a height of the luma subblock, W_c represents a width of a chroma subblock, H_c represents a height of the chroma subblock, and xS and yS represent scaling factors depending on the colour format.
8. The method of claim 7, wherein for YUV 4:2:0 of the colour format, $xS=yS=2$.

9. The method of claim 7, wherein for YUV 4:4:4 of the colour format, $xS=yS=1$.
10. The method of claim 7, wherein for YUV 4:2:2 of the colour format, $xS=2$ and $yS=1$.
11. The method of claim 1, wherein a BV of a chroma sub-block is derived from at least one BV of at least one collocated luma sub-block.
12. The method of claim 11, wherein the at least one collocated luma sub-block comprises a luma sub-block, wherein the luma sub-block comprises at least one collocated luma sample of a chroma sample in the chroma sub-block.
13. The method of claim 11, wherein the at least one collocated luma sub-block comprises a luma sub-block, wherein the luma sub-block comprises at least one of: a collocated luma sample of center of the chroma sub-block, a collocated luma sample of top-left of the chroma sub-block, a collocated luma sample of top-right of the chroma sub-block, a collocated luma sample of left-bottom of the chroma sub-block, or a collocated luma sample of right-bottom sample of the chroma sub-block.
14. The method of claim 11, wherein the at least one BV of the at least one collocated luma sub-block is weighted averaged.
15. The method of claim 11, wherein the at least one BV of the at least one collocated luma sub-block including most of collocated luma samples is used.
16. The method of claim 11, wherein the at least one BV of the at least one collocated luma sub-block at a position is used.
17. The method of claim 11, wherein the position is center.
18. The method of claim 1, wherein a derived BV is rounded to a precision.
19. The method of claim 18, wherein the precision is at least one of: integer-pixel precision, half-pixel precision, or quarter-pixel precision.

20. The method of claim 1, wherein a derived BV is clipped to a region.
21. The method of claim 20, wherein the region comprises a sample already reconstructed before a current block.
22. The method of claim 20, wherein the region comprises at least one of: a sample in a coding tree unit (CTU), or a sample in CTU row.
23. The method of claim 20, wherein the region comprises an IBC buffer.
24. The method of claim 1, wherein the parameter of an IBC-affine mode is applied by the following video unit coded with IBC-affine.
25. The method of claim 1, wherein the parameter of an IBC-affine mode is applied by the following video unit coded with inter affine.
26. The method of claim 1, wherein a parameter of an inter affine mode is applied by the following video unit coded with IBC-affine.
27. The method of claim 1, wherein an IBC-affine is used with a coding tool, and the coding tool comprises an in-loop filter approach.
28. The method of claim 27, wherein a boundary between different subblocks is processed by a deblocking filter.
29. The method of claim 1, wherein an IBC-affine is used with a coding tool, and the coding tool comprises a transform approach.
30. The method of claim 29, wherein the transform approach comprises at least one of: discrete cosine transform (DCT), discrete sine transform (DST), multiple transform selection (MTS), low-frequency non-separable transform (LFNST), or non-separable primary transform (NSPT).

31. The method of claim 1, wherein an IBC-affine is used with a coding tool, and the coding tool comprises at least one of: sign prediction, or dependent quantization.

32. The method of claim 1, wherein one or more control point block vectors (CPBVs) are refined by at least one of: template matching, or bilateral matching.

33. The method of claim 1, wherein an indication of whether to and/or how to apply IBC-affine depends on coding information, wherein the coding information comprises video content.

34. The method of any of claims 1-33, wherein the video unit comprises at least one of the followings:

- a colour component,
- a sub-picture,
- a slice,
- a tile,
- a coding tree unit (CTU),
- a CTU row,
- a group of CTU,
- a coding unit (CU),
- a prediction unit (PU),
- a transform unit (TU),
- a coding tree block (CTB),
- a coding block (CB),
- a prediction block (PB),
- a transform block (TB),
- a block,
- a sub-block of a block,
- a sub-region within a block,
- a region containing more than one sample or pixel.

35. The method of any of claims 1-33, wherein an indication of whether to and/or how to derive the BV of the at least one subblock using the affine model is indicated at one of the followings:

sequence level,
group of pictures level,
picture level,
slice level, or
tile group level.

36. The method of any of claims 1-33, wherein an indication of whether to and/or how to derive the BV of the at least one subblock using the affine model is indicated in one of the followings:

a sequence header,
a picture header,
a sequence parameter set (SPS),
a video parameter set (VPS),
a dependency parameter set (DPS),
a decoding capability information (DCI),
a picture parameter set (PPS),
an adaptation parameter sets (APS),
a slice header, or
a tile group header.

37. The method of any of claims 1-33, wherein whether to and/or how to derive the BV of the at least one subblock using the affine model depends on at least one of the followings:

a message indicated in one of: DPS, SPS, VPS, PPS, APS, picture header, slice header, tile group header, coding tree unit (CTU), coding unit (CU), CTU row, group of CTUs, TU, PU block, video coding unit,

a position of a CU block,
a position of a PU block,
a position of a TU block,
a position of a video coding unit,
block dimension of a current block,
block dimension of a neighbouring block of the current block,
a coded mode of a block,
an indication of a colour format,
a coding tree structure,

a slice group type,
a tile group type,
a slice picture type,
a tile picture type,
a colour component,
an ID of a temporal layer,
a profile of a standard,
a level of a standard, or
a tier of a standard.

38. The method of claim 37, wherein the coded mode of the block is at least one of: IBC inter mode, non-IBC inter mode, or non-IBC subblock mode.

39. The method of claim 37, wherein the indication of a colour format is 4:2:0 or 4:4:4.

40. The method of claim 37, wherein the colour component is applied on a chroma component or a luma component.

41. The method of any of claims 1-33, wherein the syntax element is binarized as a flag, a fixed length code, an EG(x) code, a unary code, a truncated unary code, or a truncated binary code.

42. The method of any of claims 1-33, wherein the syntax element is signed or unsigned.

43. The method of any of claims 1-33, wherein the syntax element is coded with at least one context model, or bypass coded.

44. The method of any of claims 1-33, wherein the syntax element is signaled in at least one of following conditions:

if a corresponding function is applicable, or
if dimensions of the block satisfy a condition.

45. The method of claim 44, wherein the dimensions are width and/or height.

46. The method of any of claims 1-33, wherein the syntax element is signaled at one of the followings:

- a block level,
- a sequence level,
- a group of pictures level,
- a picture level,
- a slice level, or
- a tile group level.

47. The method of any of claims 1-33, wherein the syntax element is signaled in one of the followings:

- a coding structure of CTU,
- a coding structure of CU,
- a coding structure of TU,
- a coding structure of PU,
- a coding structure of CTB,
- a coding structure of CB,
- a coding structure of TB,
- a coding structure of PB,
- a sequence header,
- a picture header,
- a sequence parameter set (SPS),
- a video parameter set (VPS),
- a dependency parameter set (DPS),
- a decoding capability information (DCI),
- a picture parameter set (PPS),
- an adaptation parameter sets (APS),
- a slice header, or
- a tile group header.

48. The method of any of claims 1-33, wherein IBC with an affine model is combined with at least one of the following coding tools:

- affine,
- multiple transform selection (MTS),

low-frequency non-separable transform (LFNST),
merge mode with motion vector difference (MMVD),
MIP,
ISP,
cross-component linear model (CCLM),
convolutional cross-component model (CCCM),
symmetric motion vector differences (SMVD),
bi-directional optical flow (BDOF),
decoder side motion vector refinement (DMVR),
history-based MVP (HMVP),
template matching,
intra block copy (IBC), or
Palette.

49. The method of any of claims 1-33, wherein IBC with an affine model is excluded with at least one of the following coding tools:

affine,
multiple transform selection (MTS),
LFNST,
merge mode with motion vector difference (MMVD),
MIP,
ISP,
cross-component linear model (CCLM),
convolutional cross-component model (CCCM),
SMVD,
bi-directional optical flow (BDOF),
decoder side motion vector refinement (DMVR),
history-based MVP (HMVP),
template matching,
intra block copy (IBC), or
Palette.

50. The method of claim 49, wherein if the IBC with an affine model is used, the excluded coding tool is disabled without signaling.

51. The method of claim 49, wherein if the excluded coding tool is used, the IBC with an affine model is disabled without signaling.

52. The method of any of claims 1-51, wherein the conversion includes encoding the video unit into the bitstream.

53. The method of any of claims 1-51, wherein the conversion includes decoding the video unit from the bitstream.

54. An apparatus for video processing comprising a processor and a non-transitory memory with instructions thereon, wherein the instructions upon execution by the processor, cause the processor to perform a method in accordance with any of claims 1-53.

55. A non-transitory computer-readable storage medium storing instructions that cause a processor to perform a method in accordance with any of claims 1-53.

56. A non-transitory computer-readable recording medium storing a bitstream of a video which is generated by a method performed by an apparatus for video processing, wherein the method comprises:

deriving a block vector (BV) for at least one subblock of a plurality of subblocks of a video unit of the video using an affine model, wherein a same block size is used for all of the plurality of subblocks, and/or at least one of: a parameter of an affine mode or a position of the video unit is applied to a following video unit of the video unit; and

generating the bitstream based on the BV.

57. A method for storing a bitstream of a video, comprising:

deriving a block vector (BV) for at least one subblock of a plurality of subblocks of a video unit of the video using an affine model, wherein a same block size is used for all of the plurality of subblocks, and/or at least one of: a parameter of an affine mode or a position of the video unit is applied to a following video unit of the video unit;

generating the bitstream based on the BV; and

storing the bitstream in a non-transitory computer-readable recording medium.

DRAWINGS

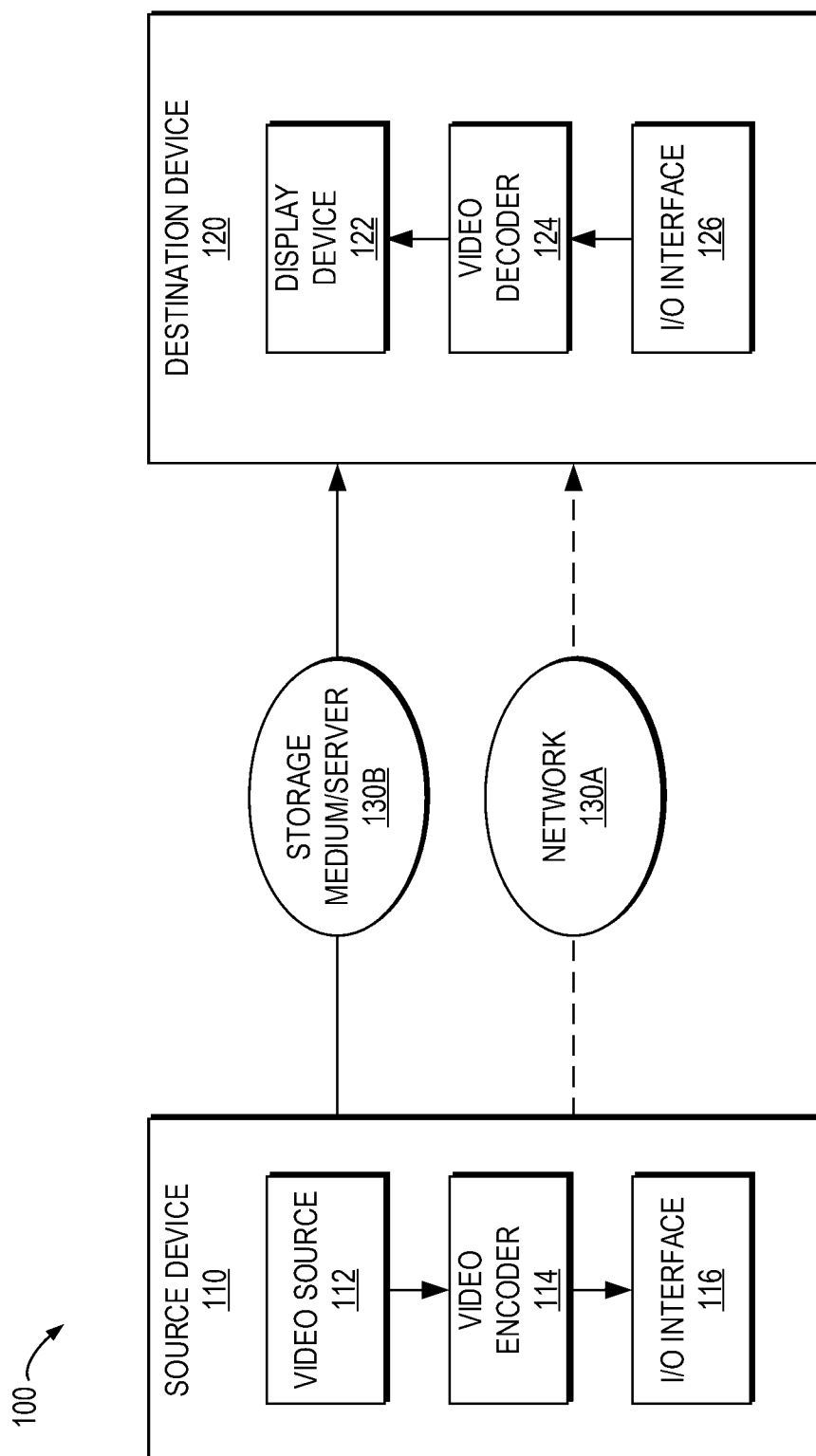


Fig. 1

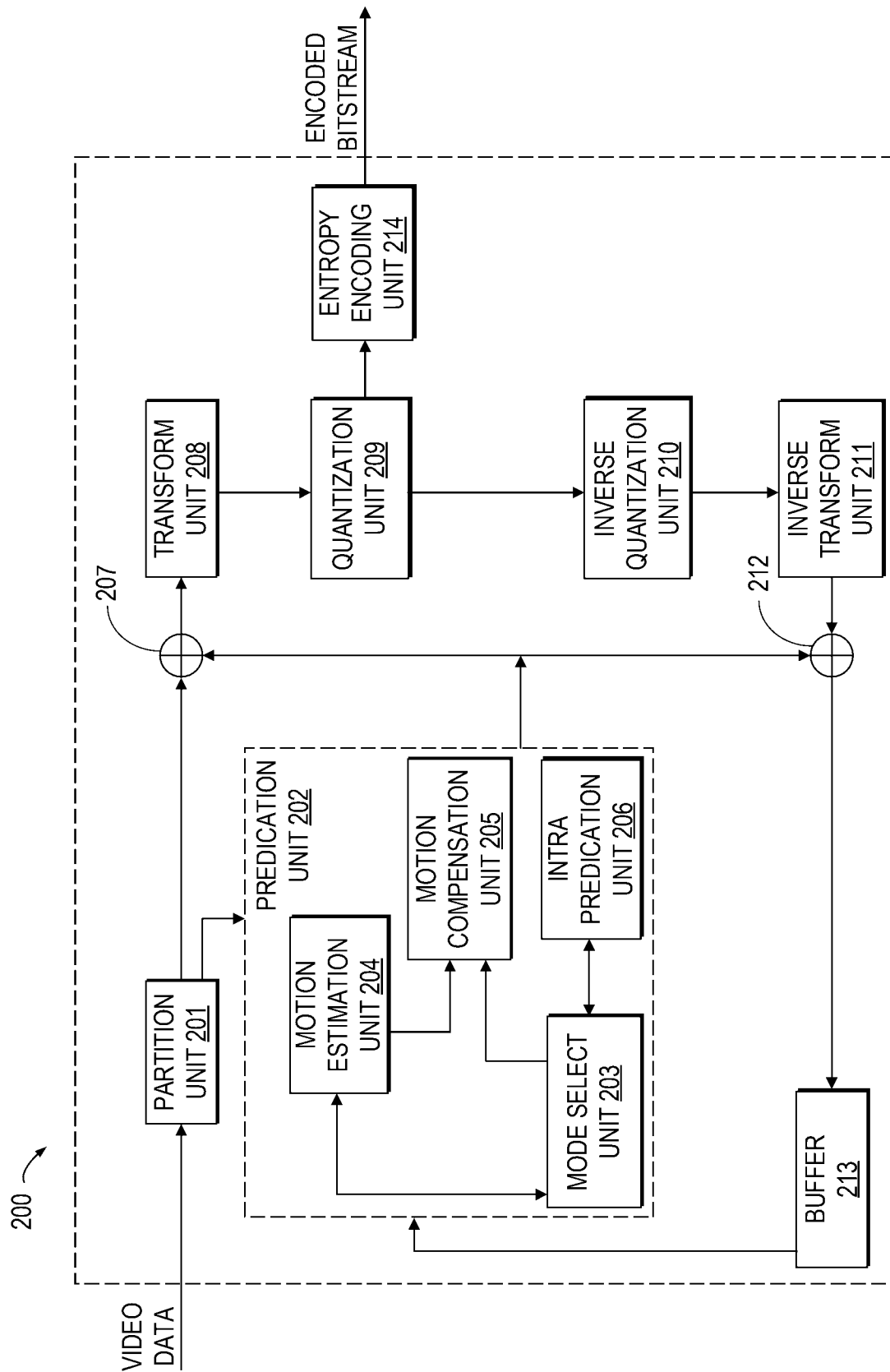


Fig. 2

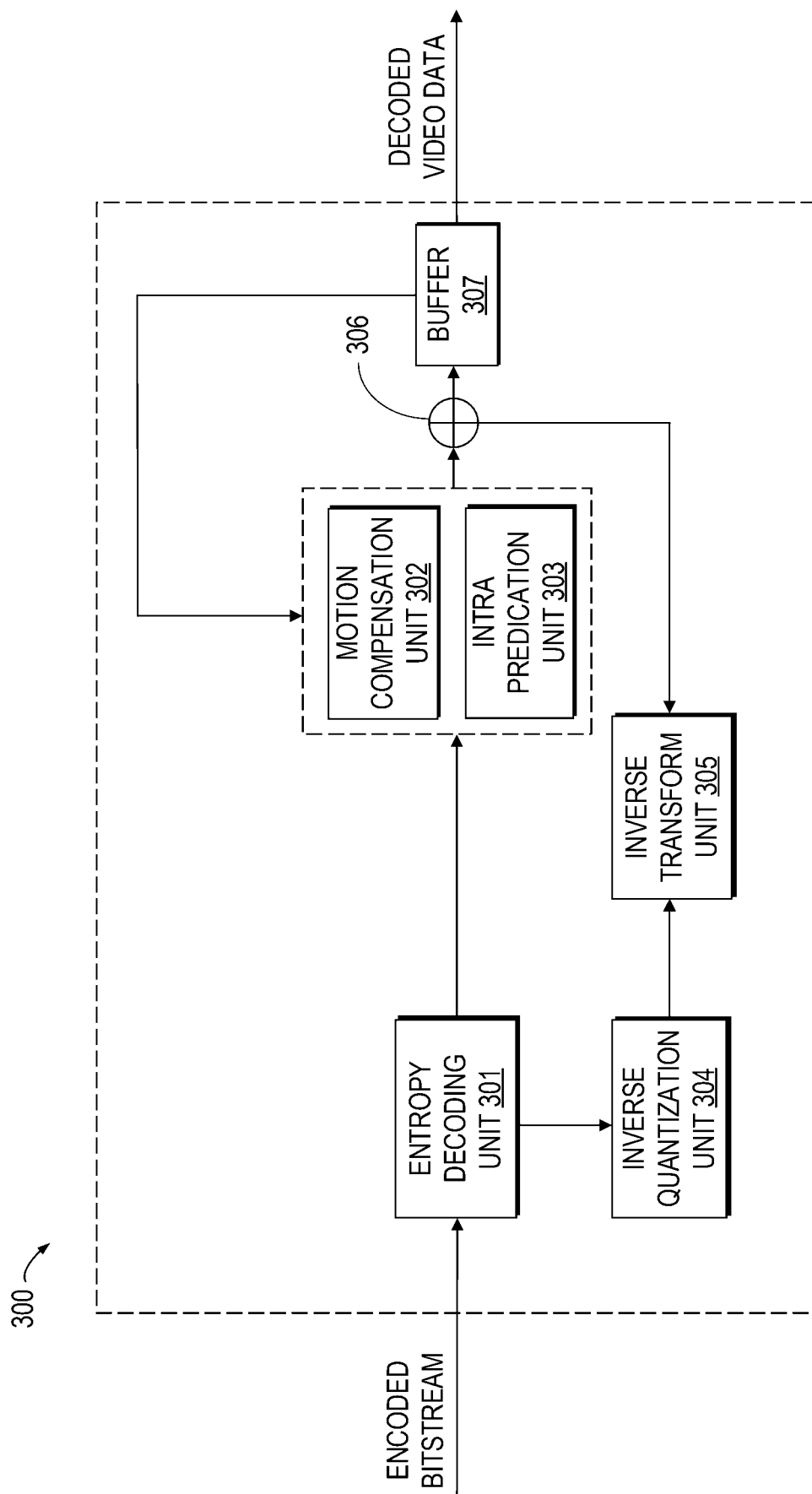


Fig. 3

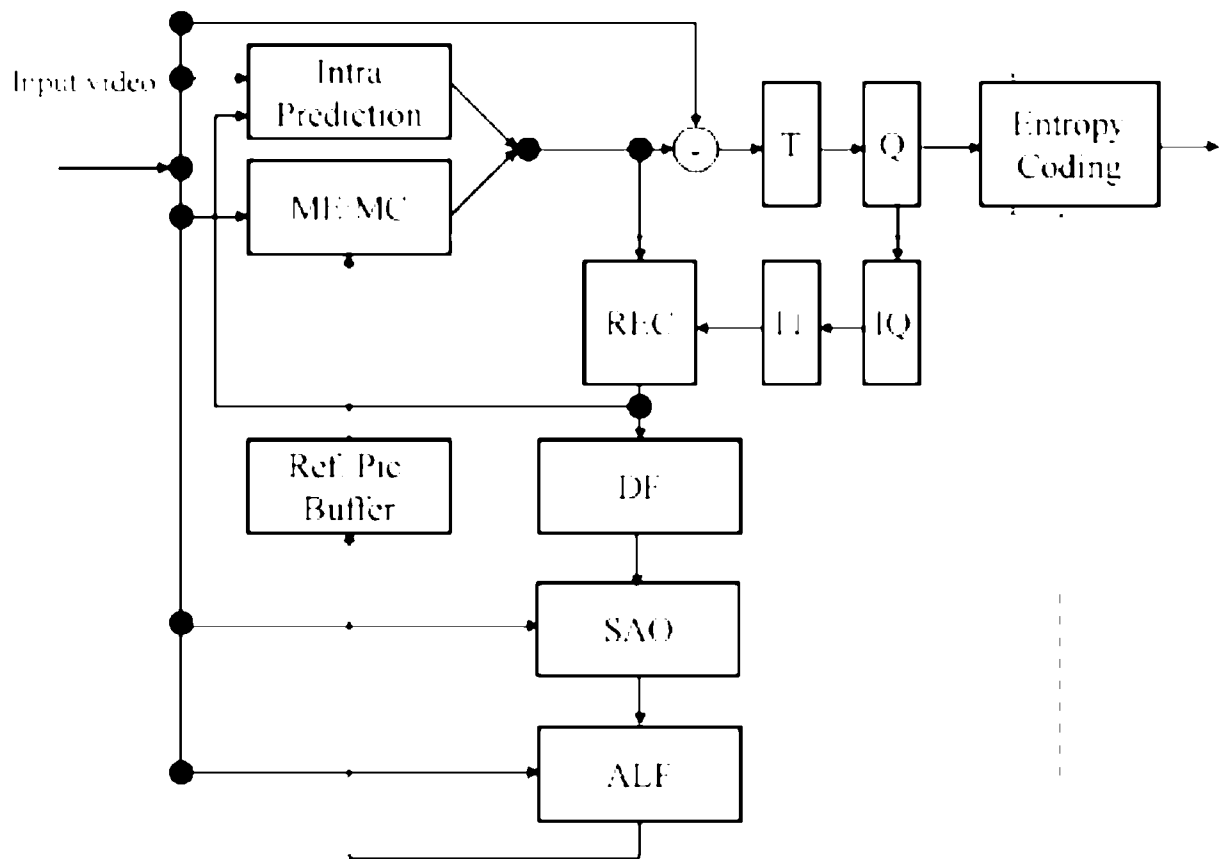


Fig. 4

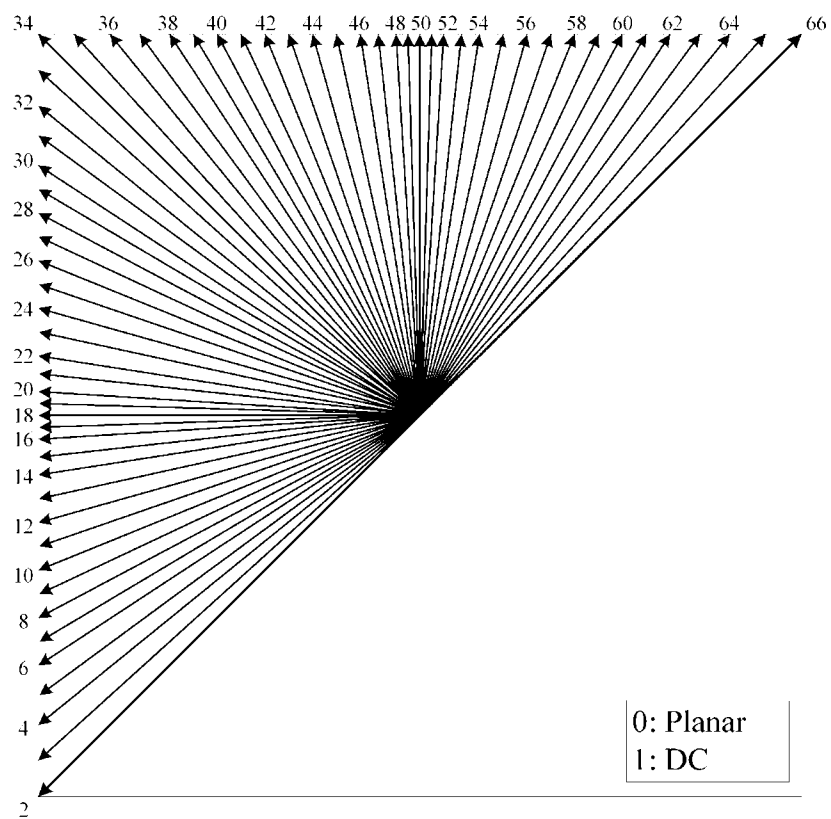


Fig. 5

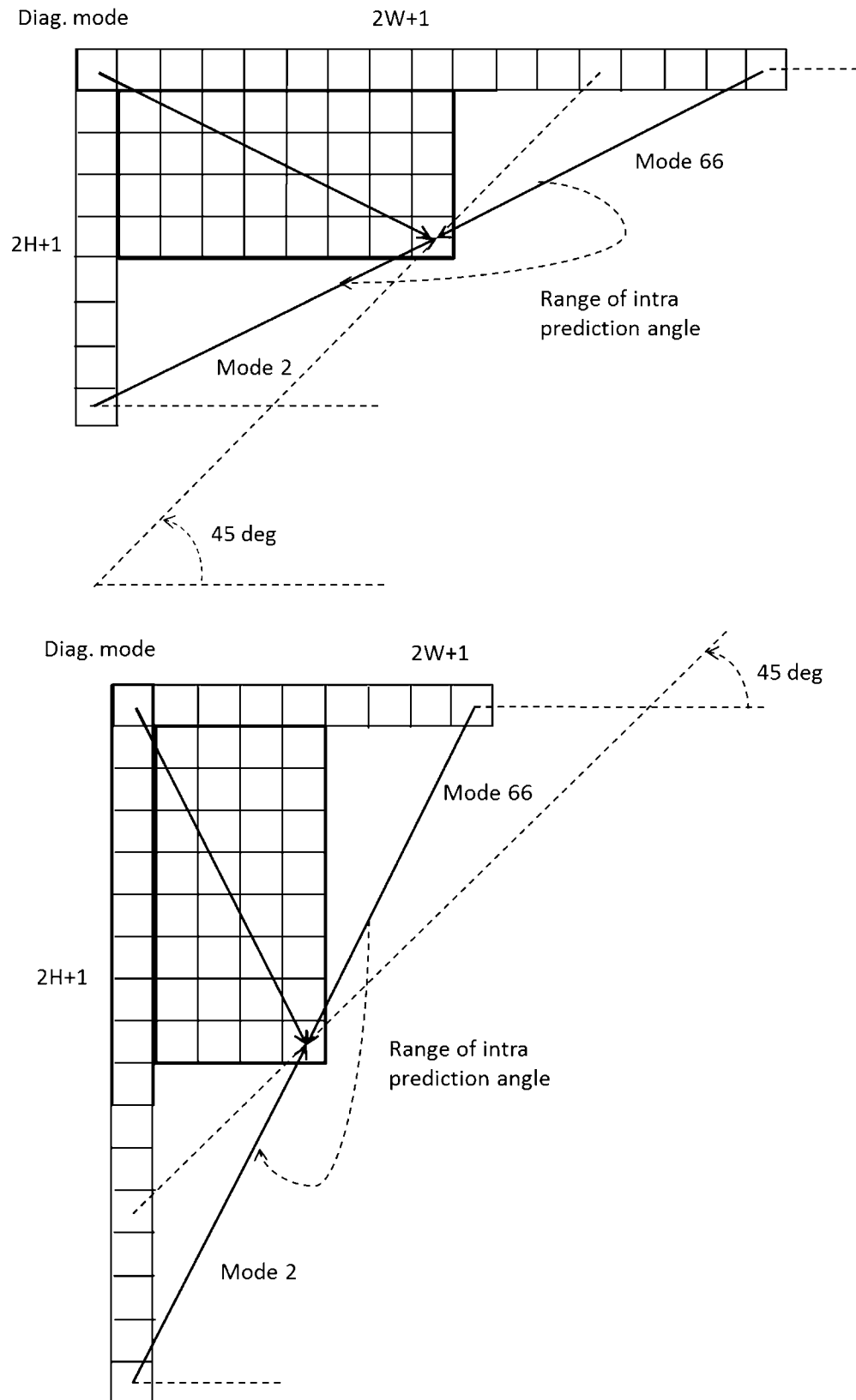


Fig. 6

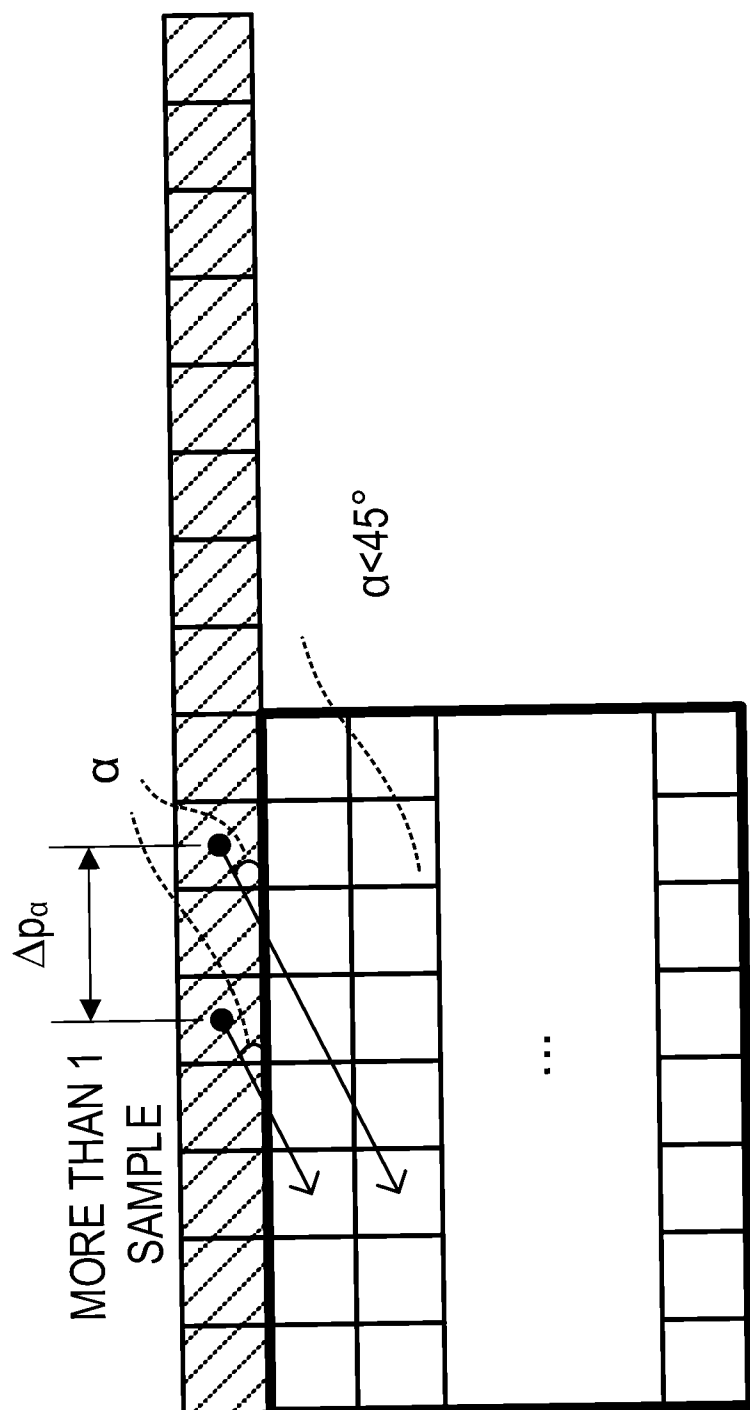


Fig. 7

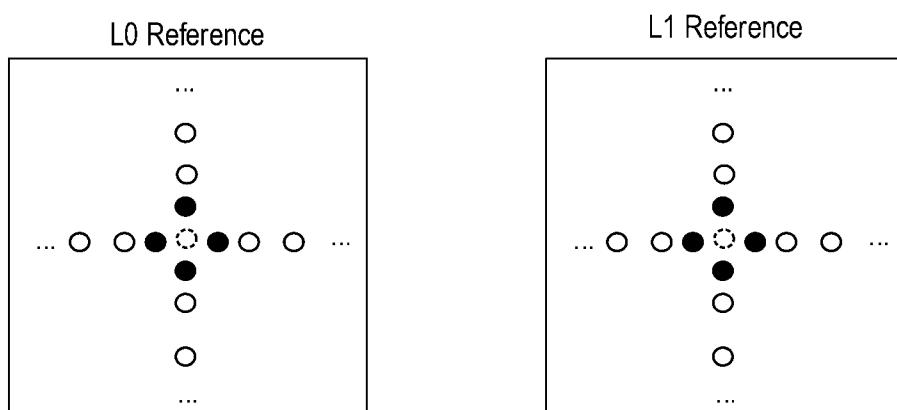


Fig. 8

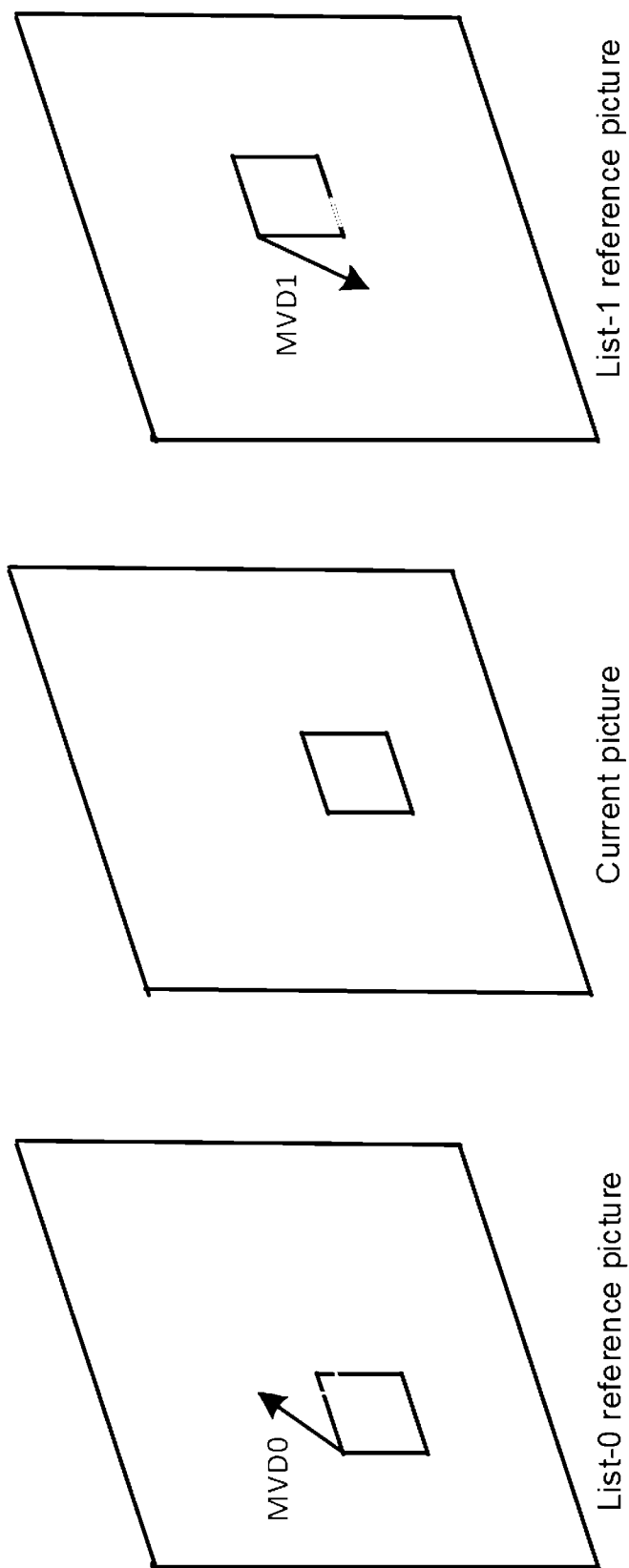


Fig. 9

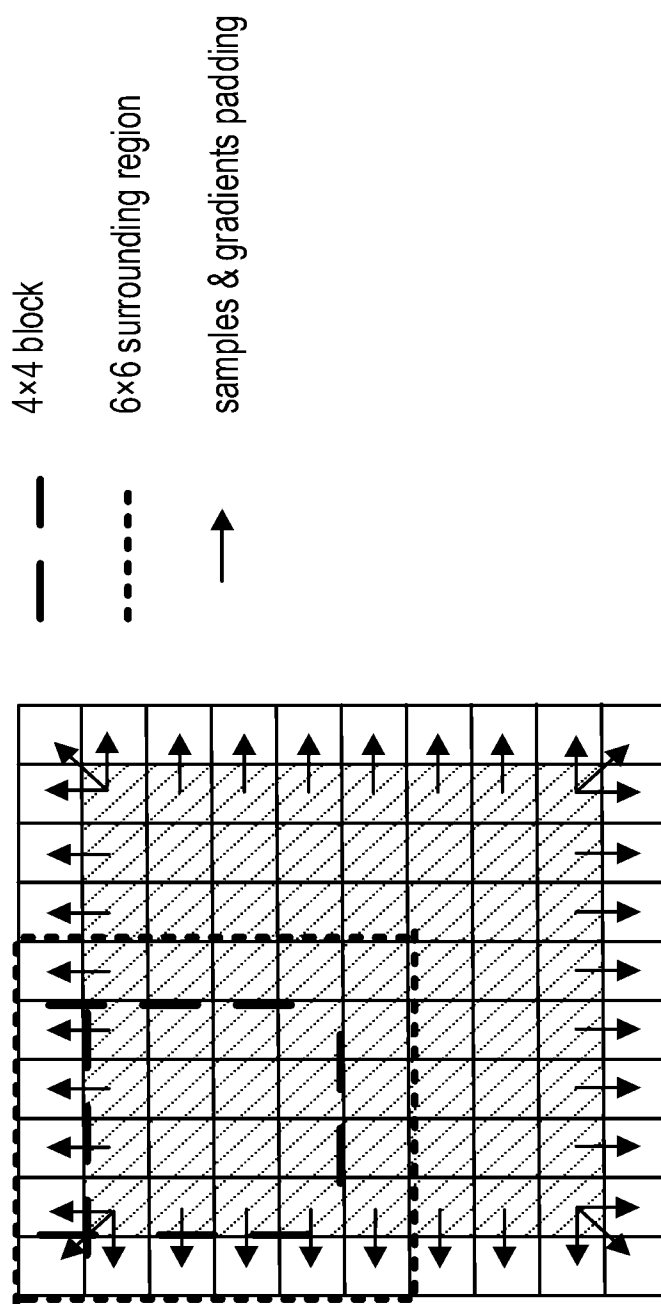


Fig. 10

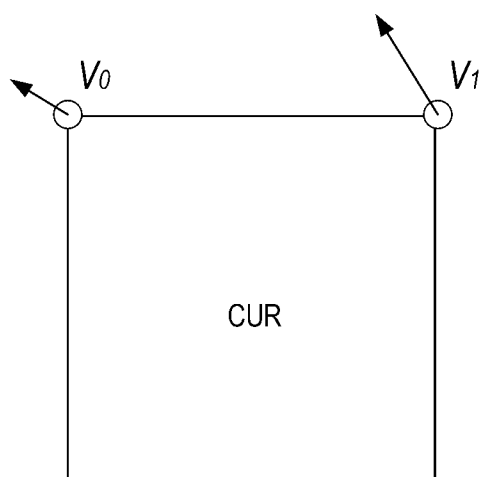


Fig. 11a

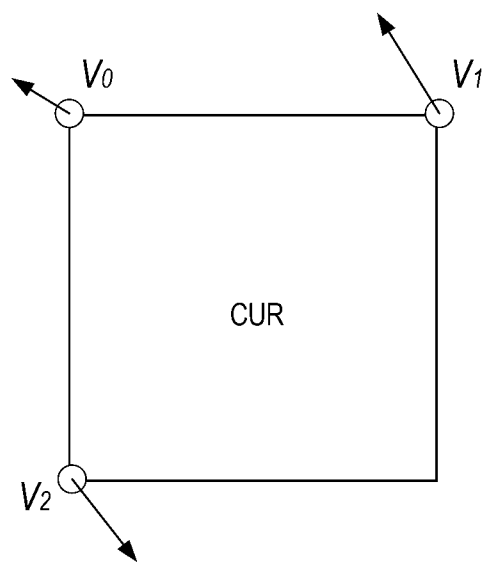


Fig. 11b

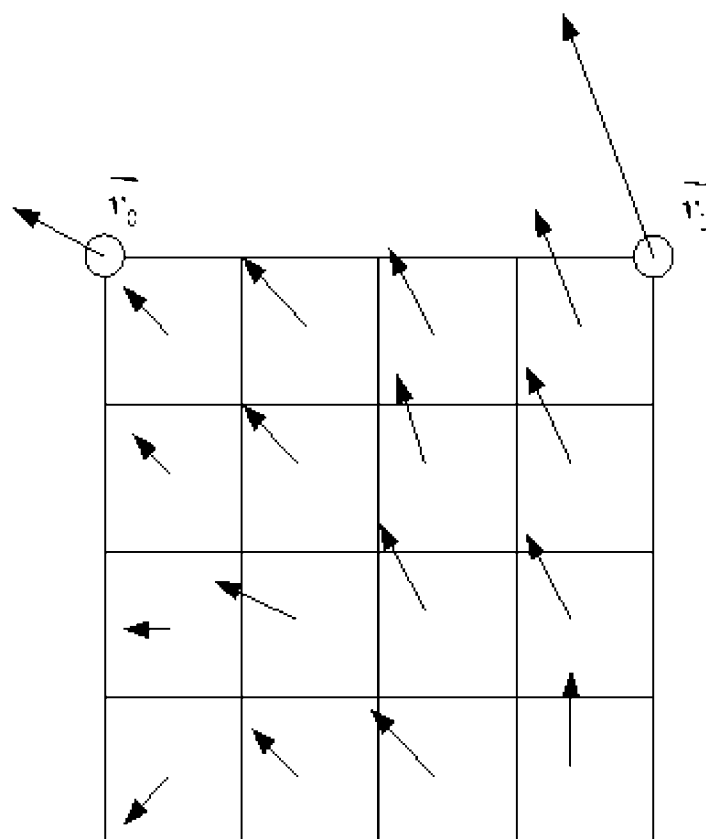


Fig. 12

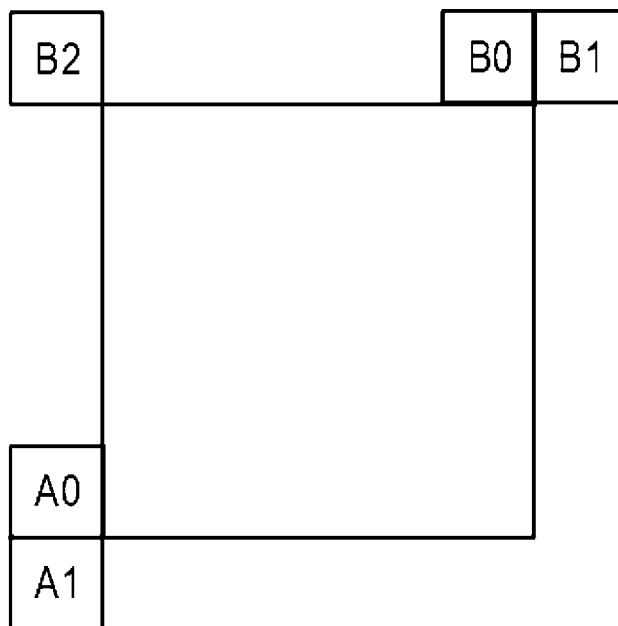


Fig. 13

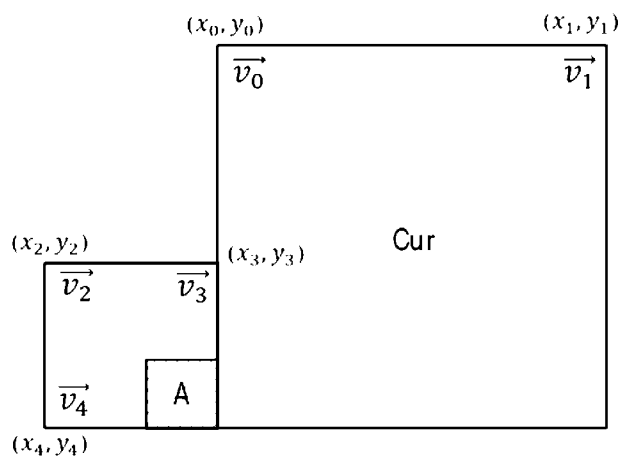


Fig. 14

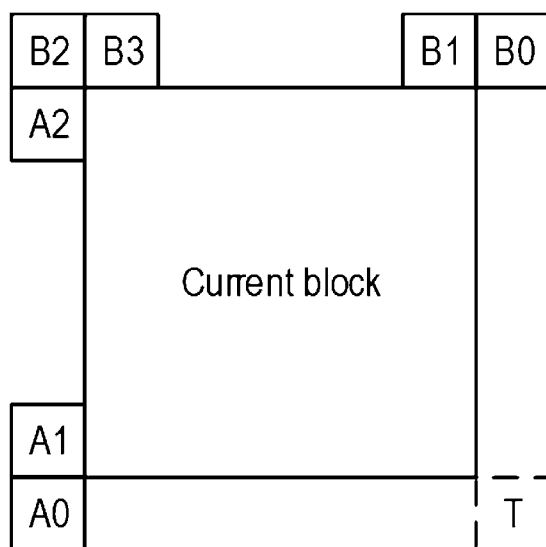


Fig. 15

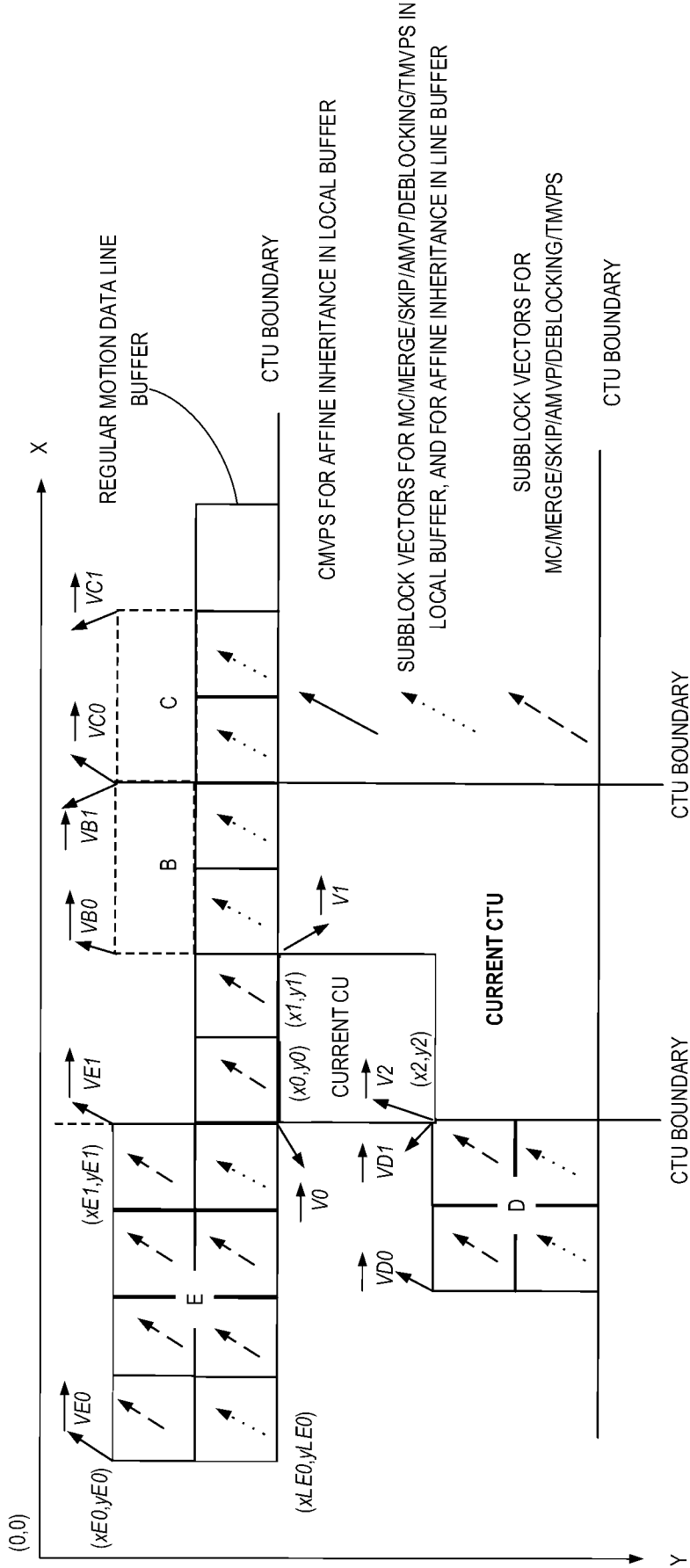


Fig. 16

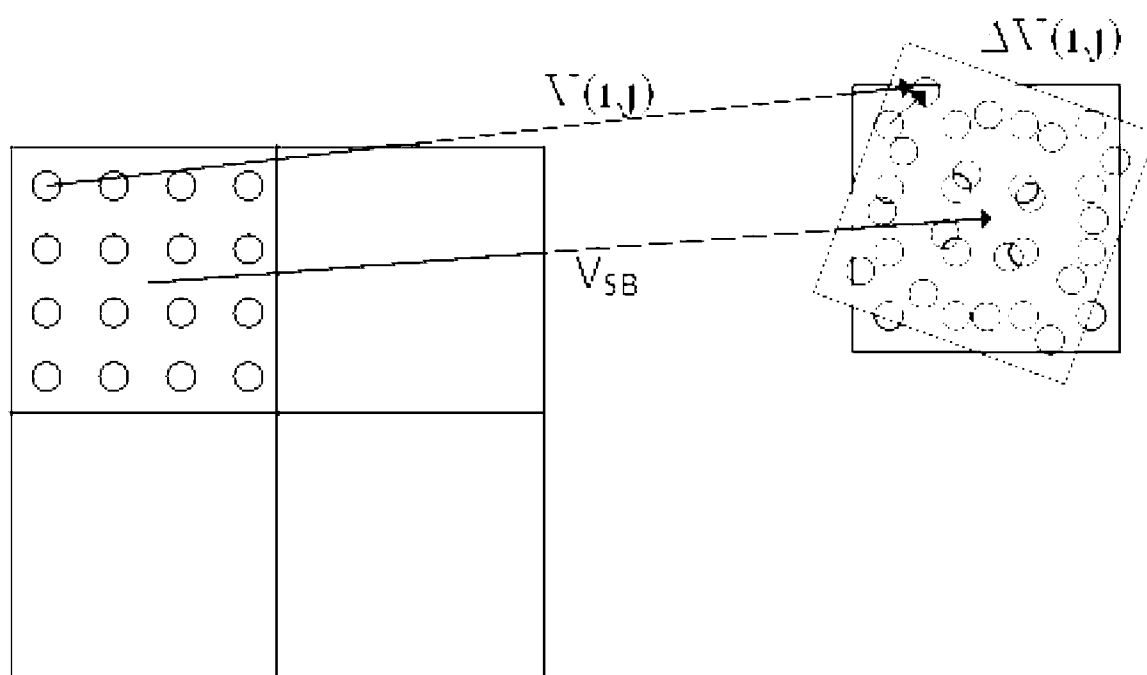


Fig. 17

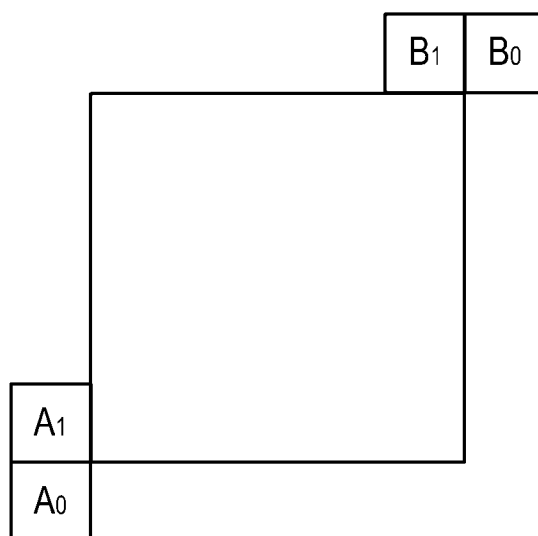


Fig. 18a

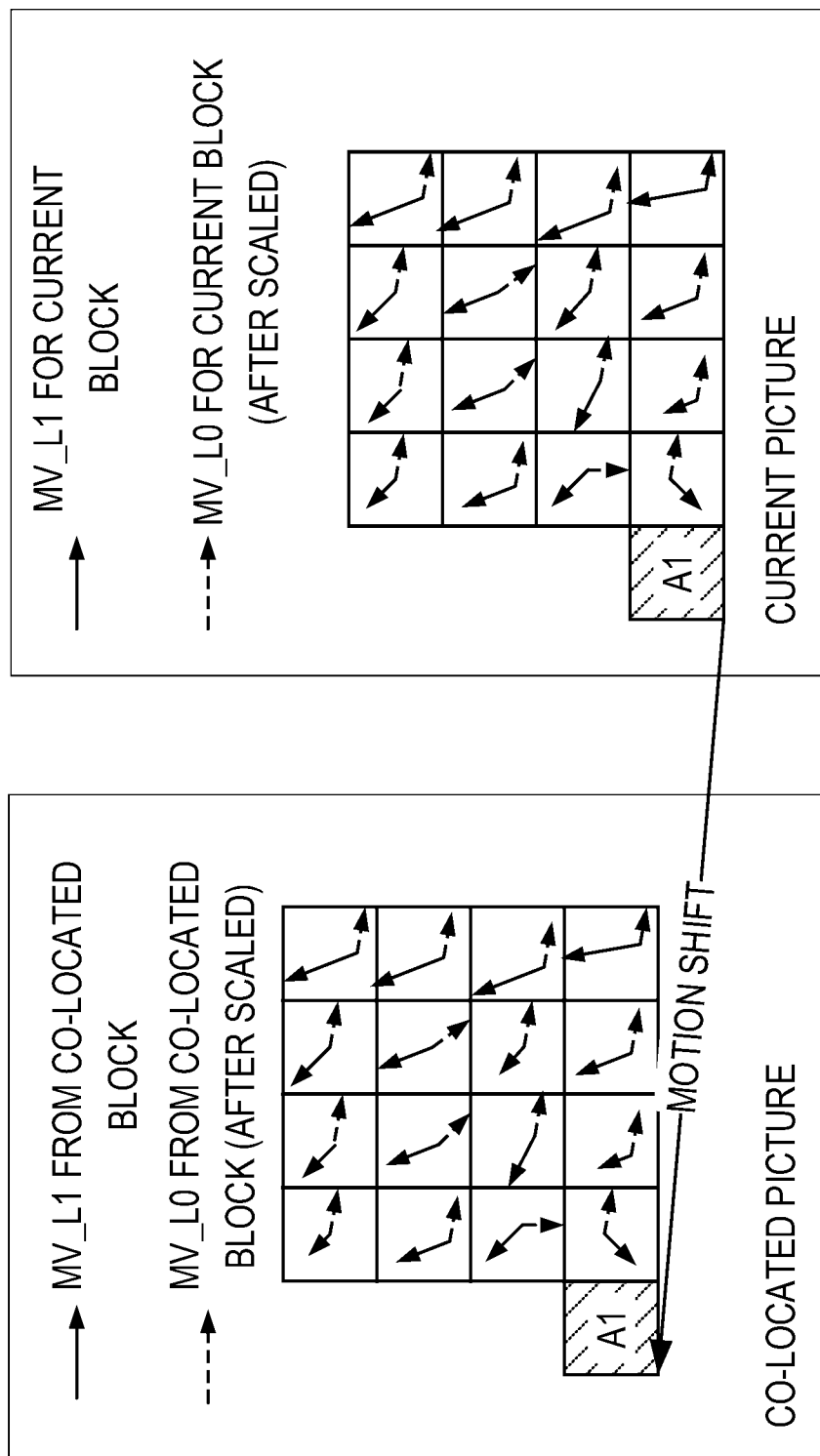


Fig. 18b

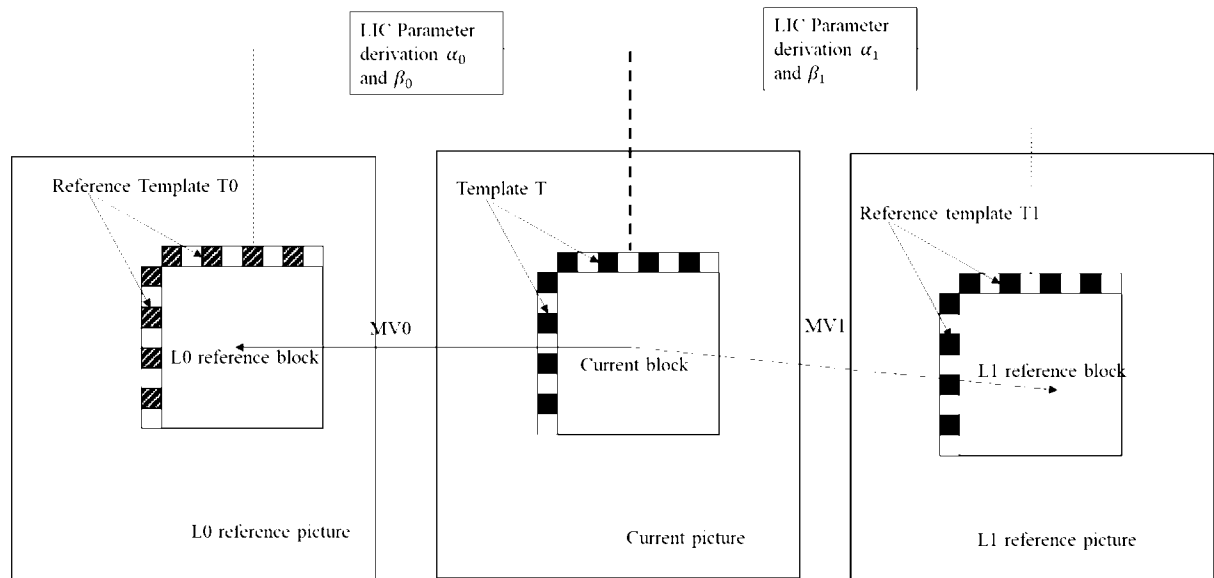


Fig. 19

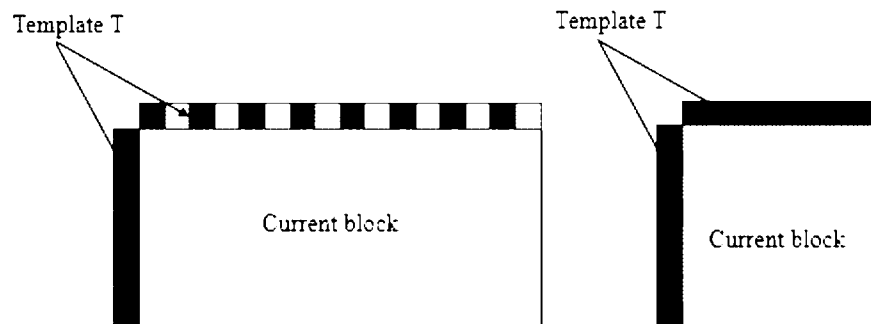


Fig. 20

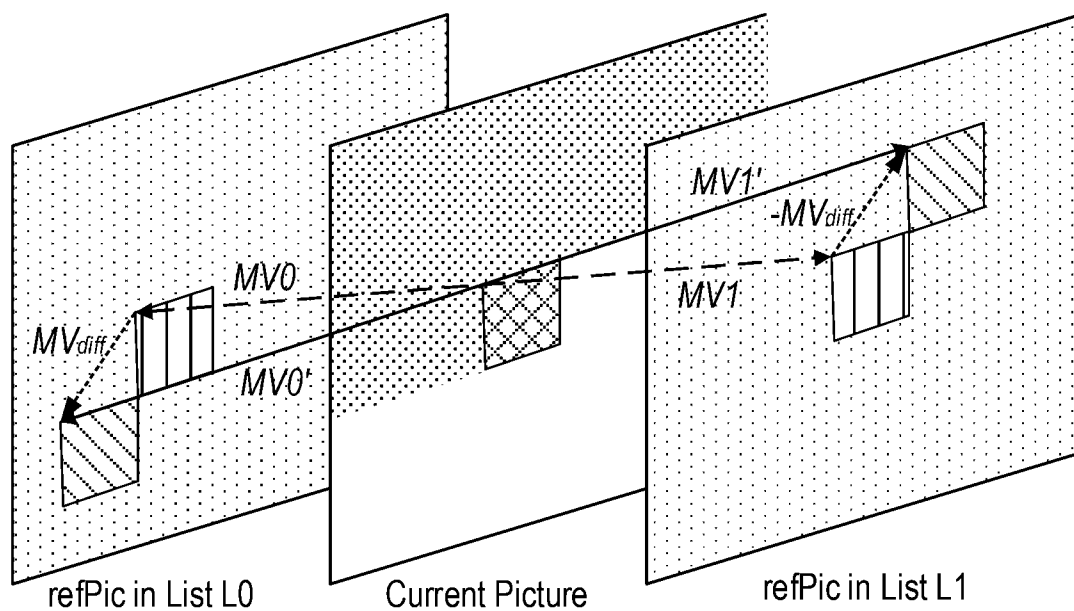


Fig. 21

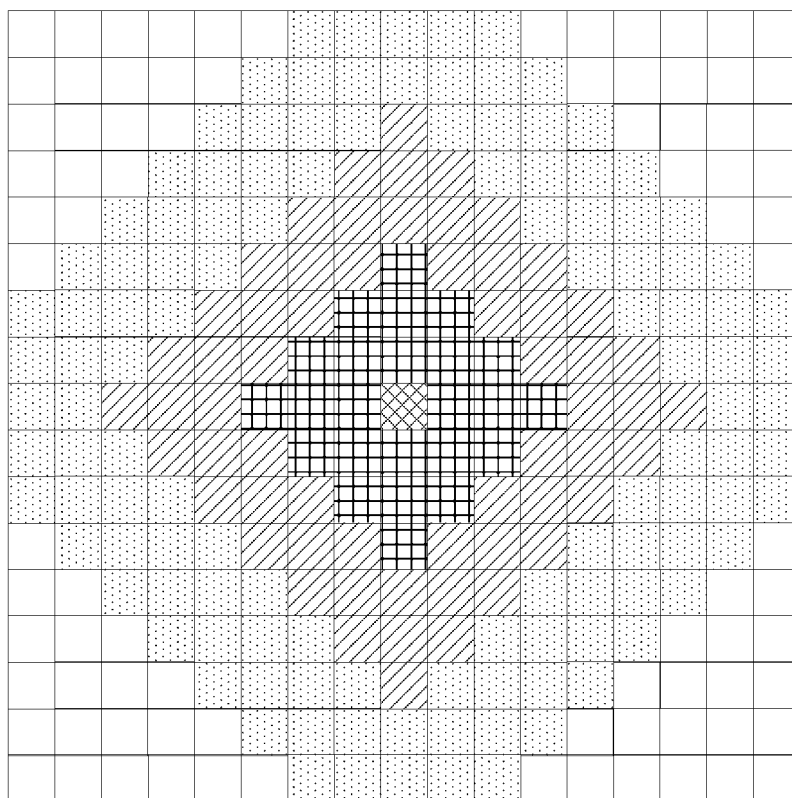


Fig. 22

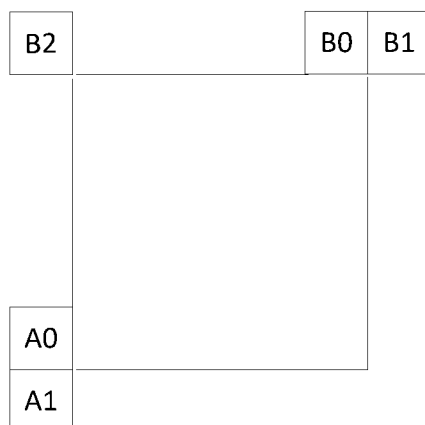


Fig. 23

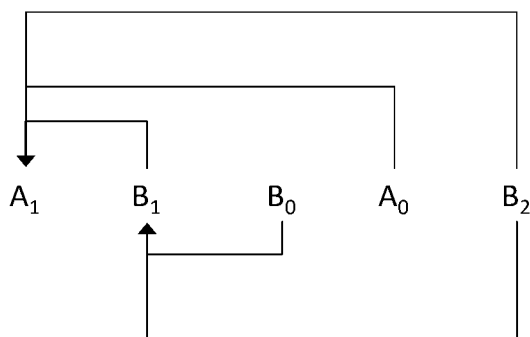


Fig. 24

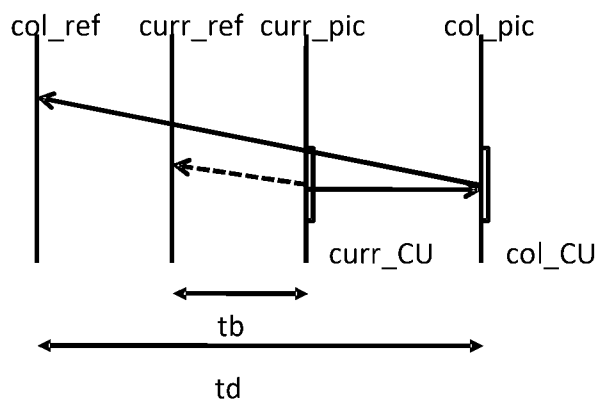


Fig. 25

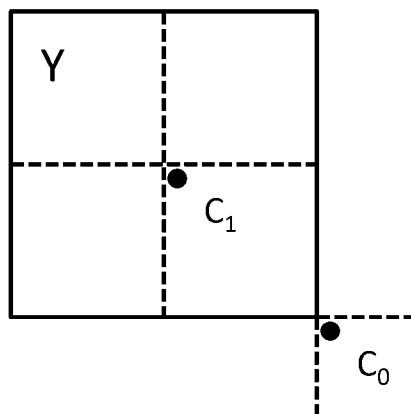


Fig. 26

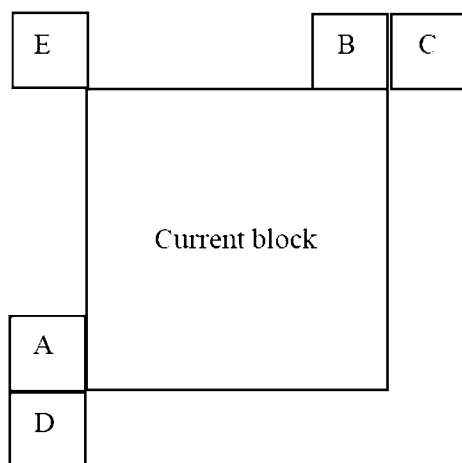


Fig. 27

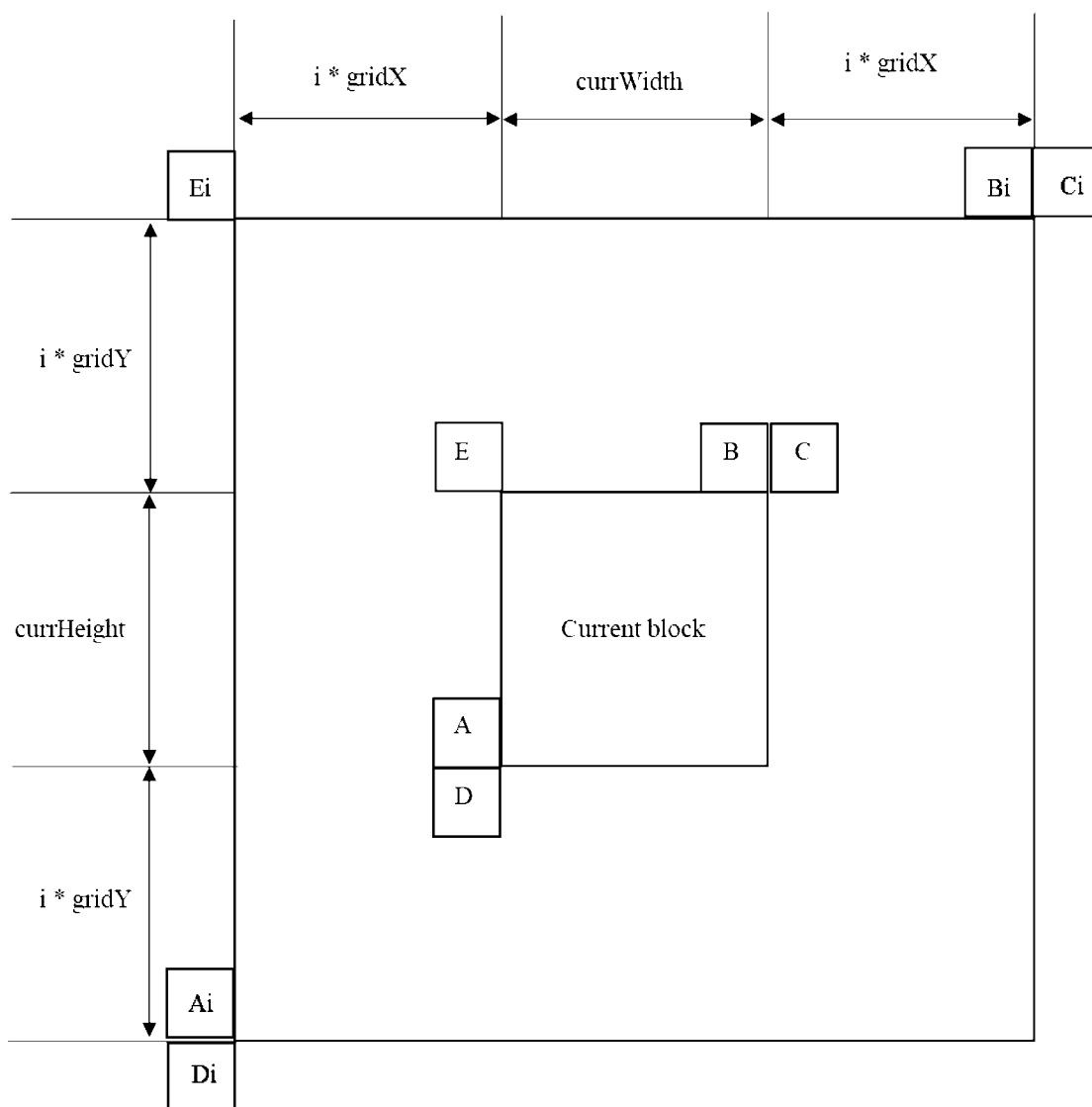


Fig. 28

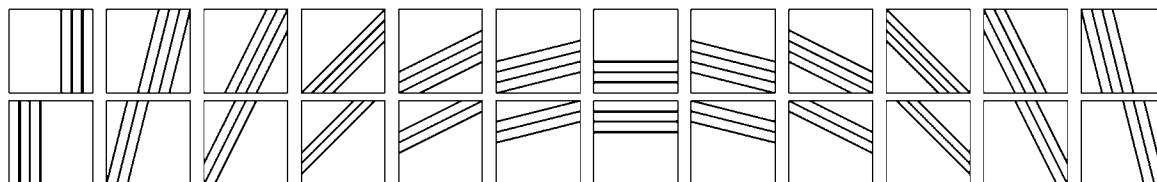


Fig. 29

	L0 MV	L1 MV
Merge Index		
0	x	
1		x
2	x	
3		x
4	x	

Fig. 30

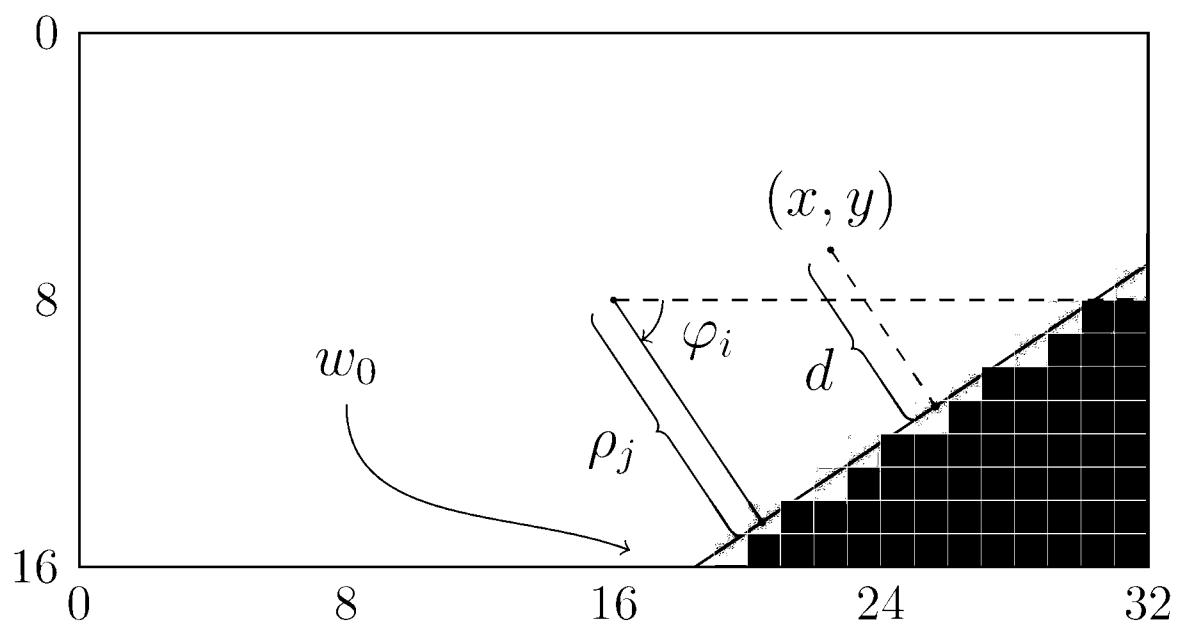


Fig. 31

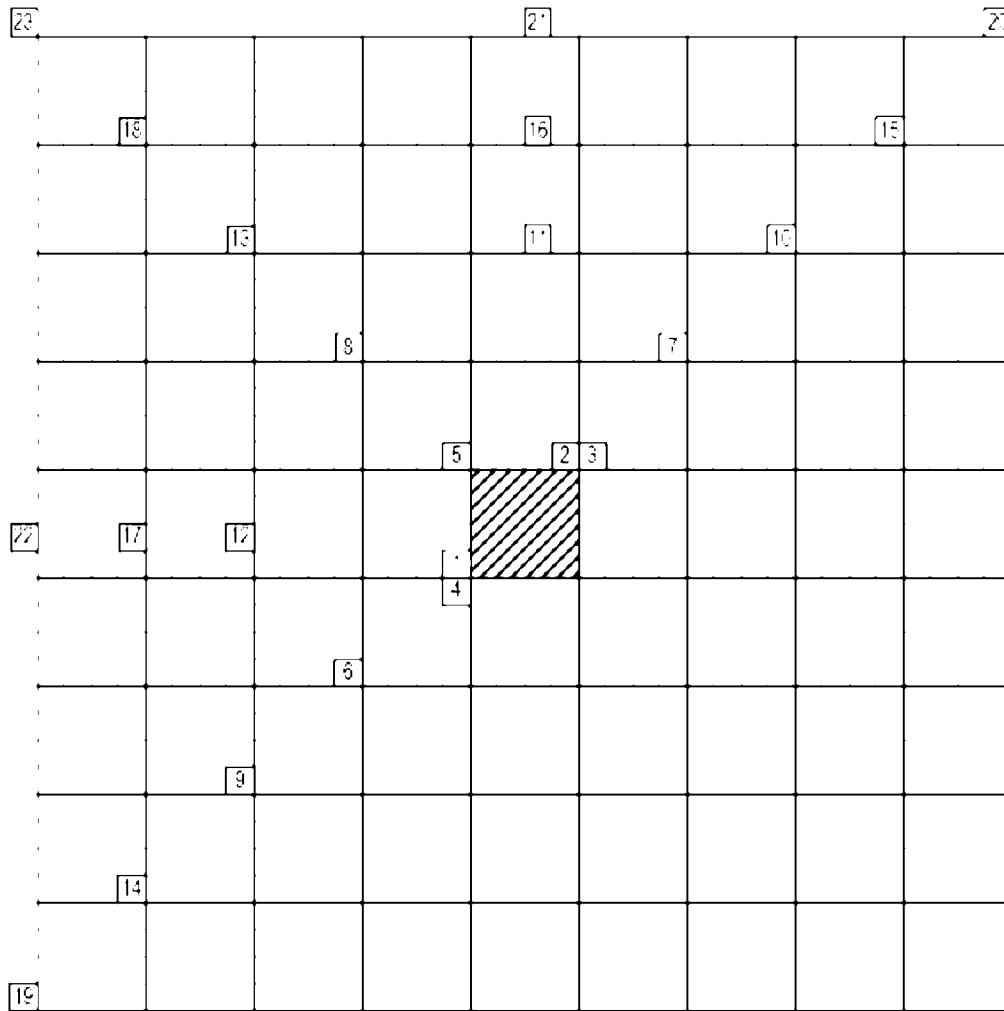


Fig. 32

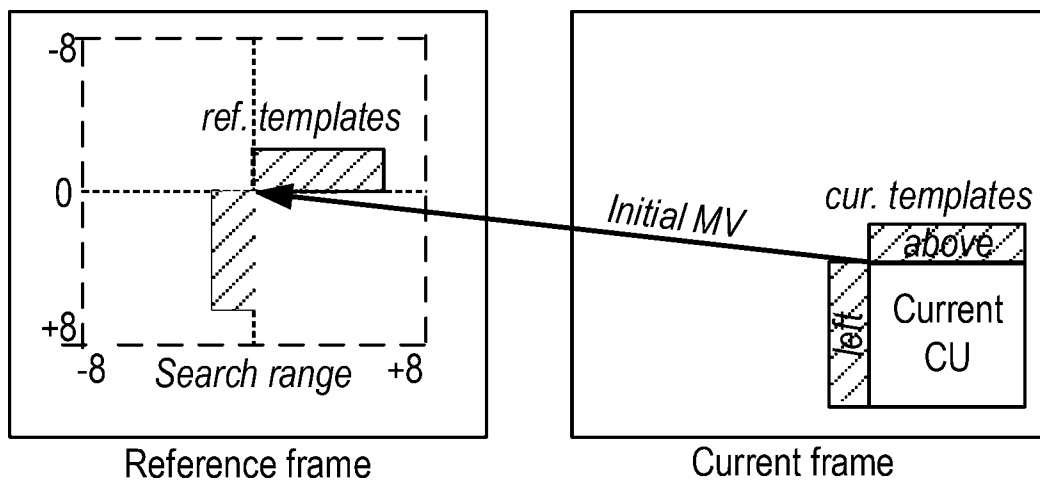


Fig. 33

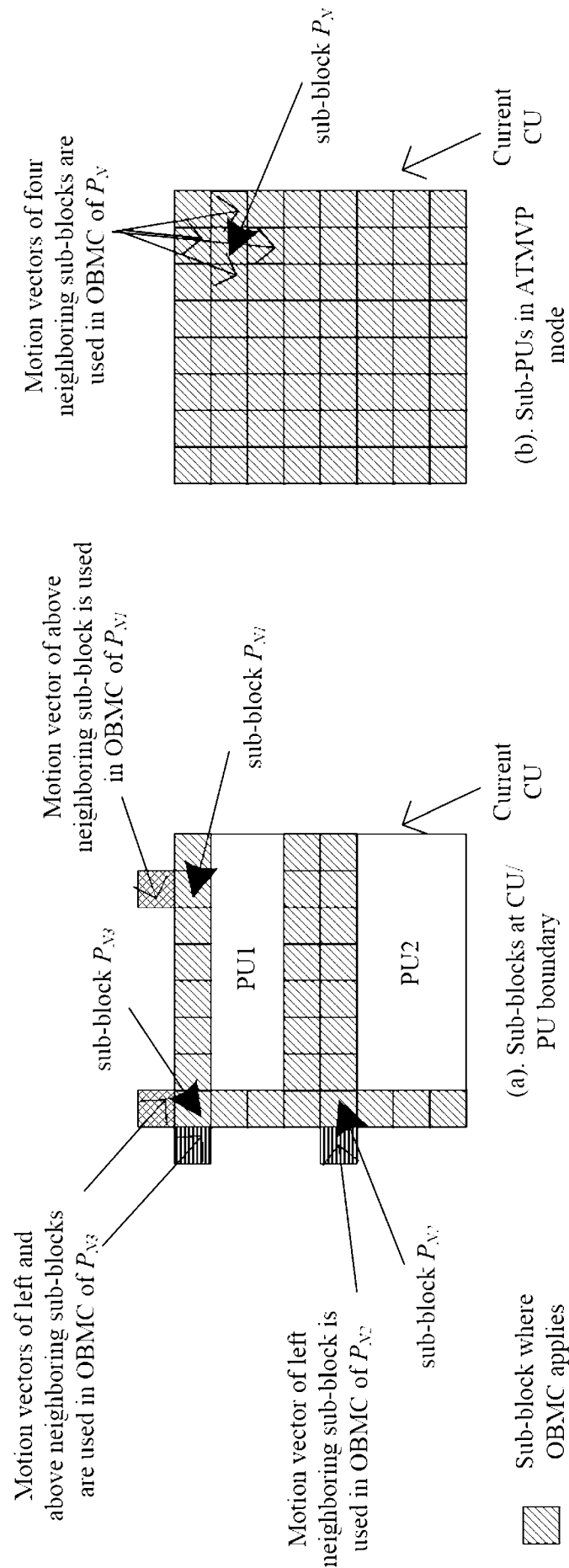


Fig. 34

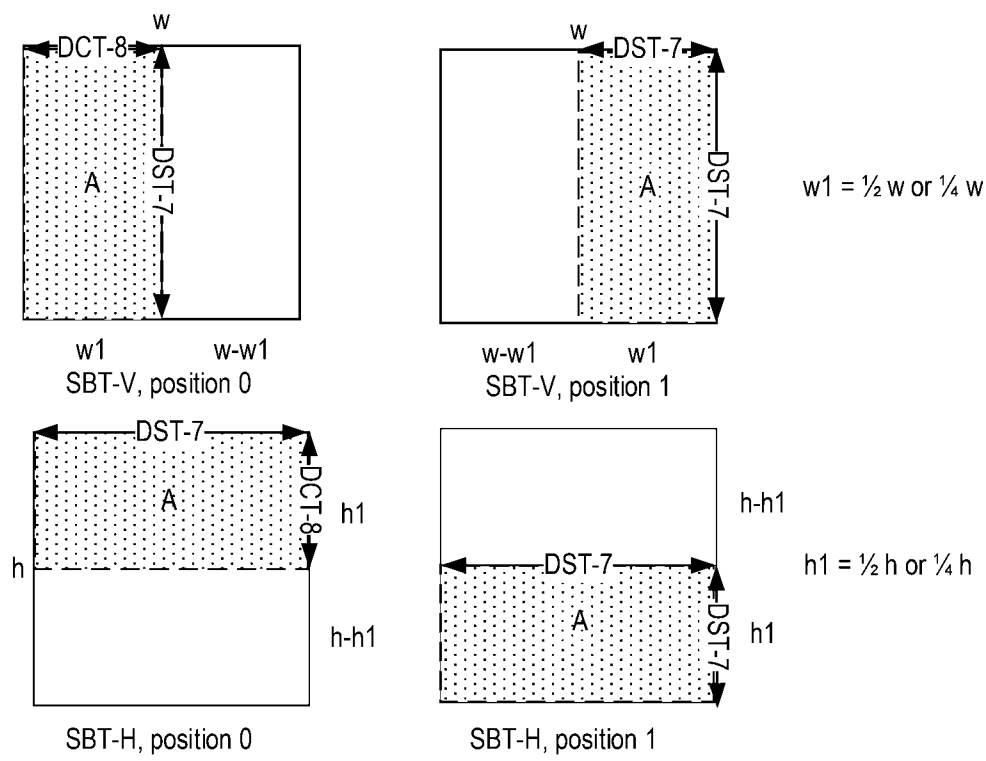


Fig. 35

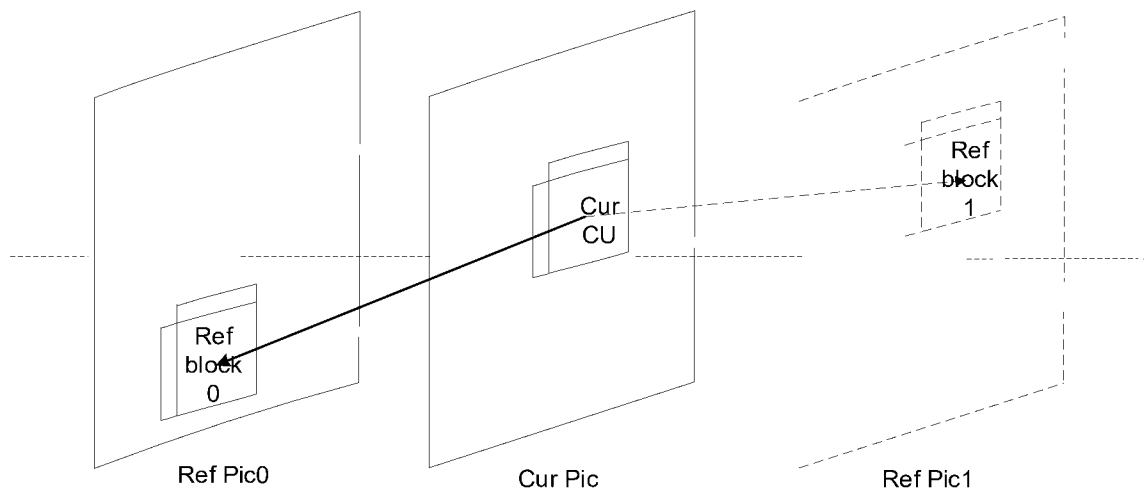


Fig. 36

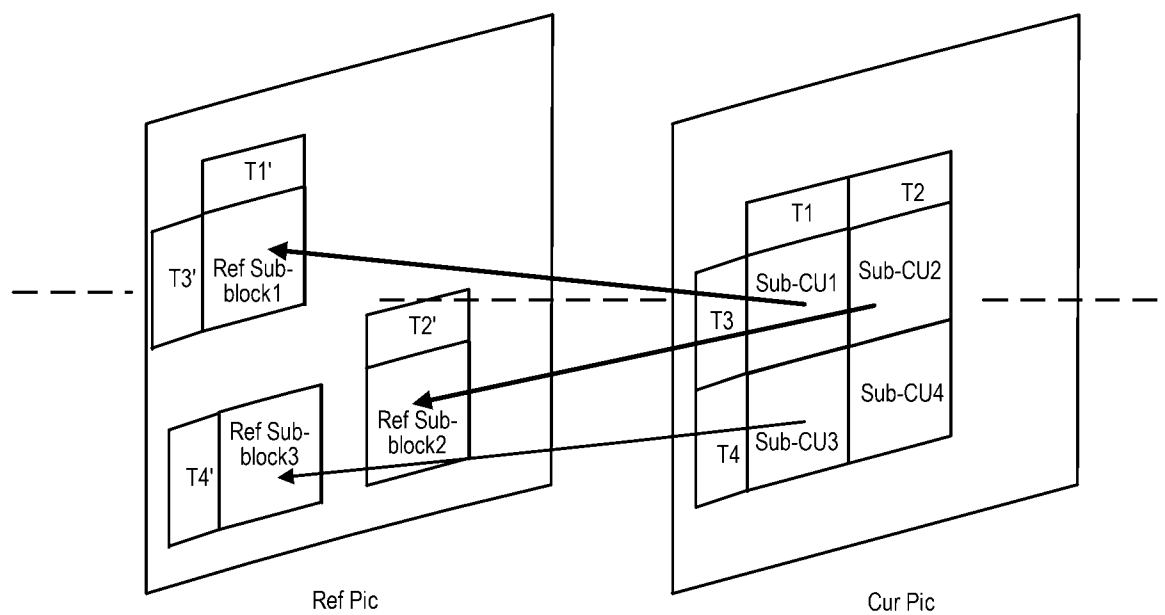


Fig. 37

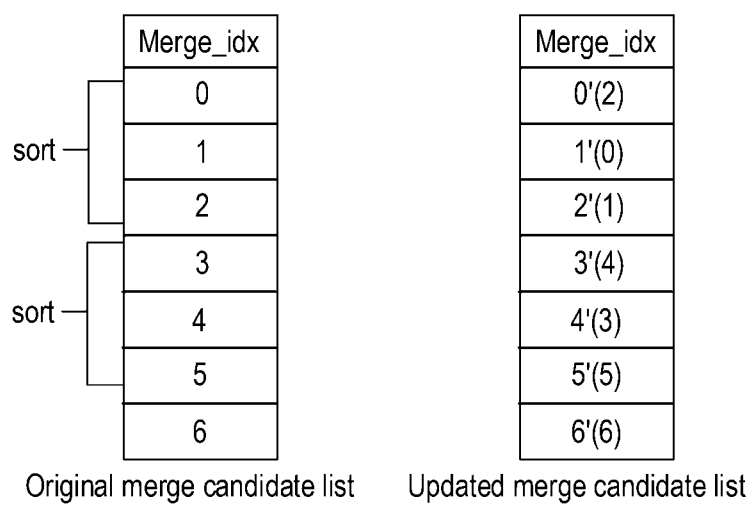


Fig. 38

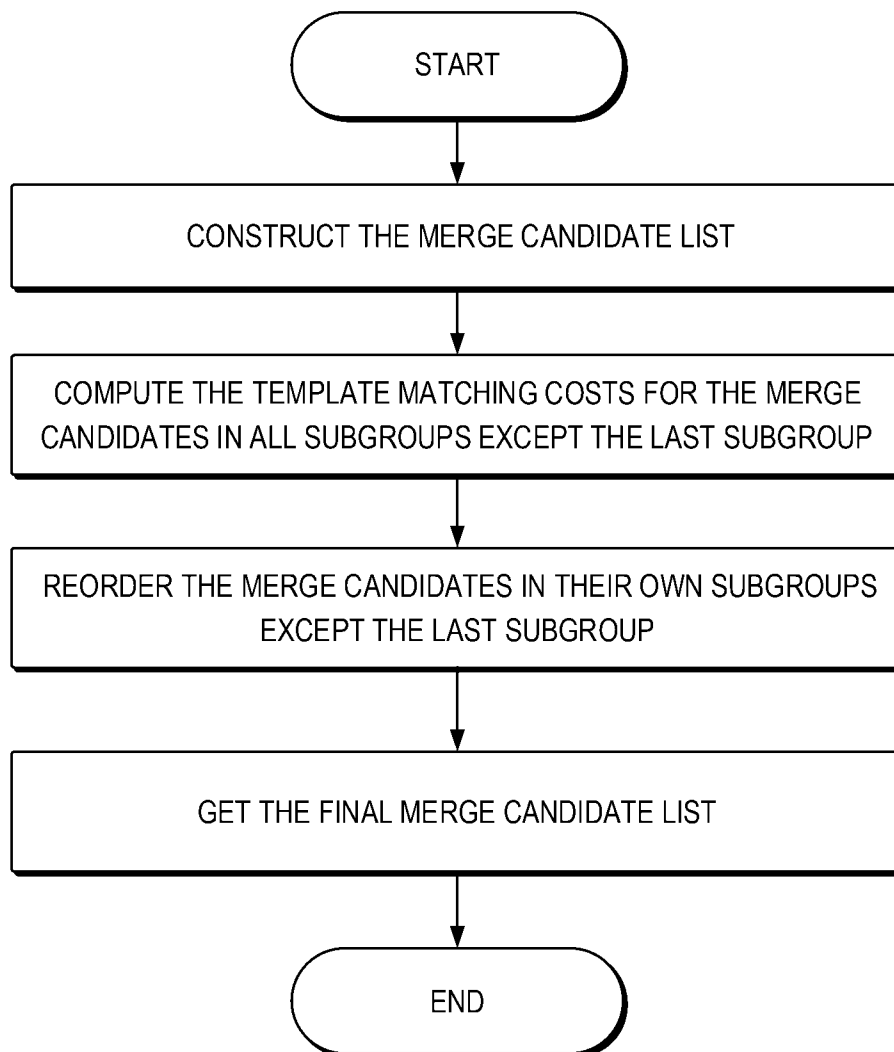


Fig. 39

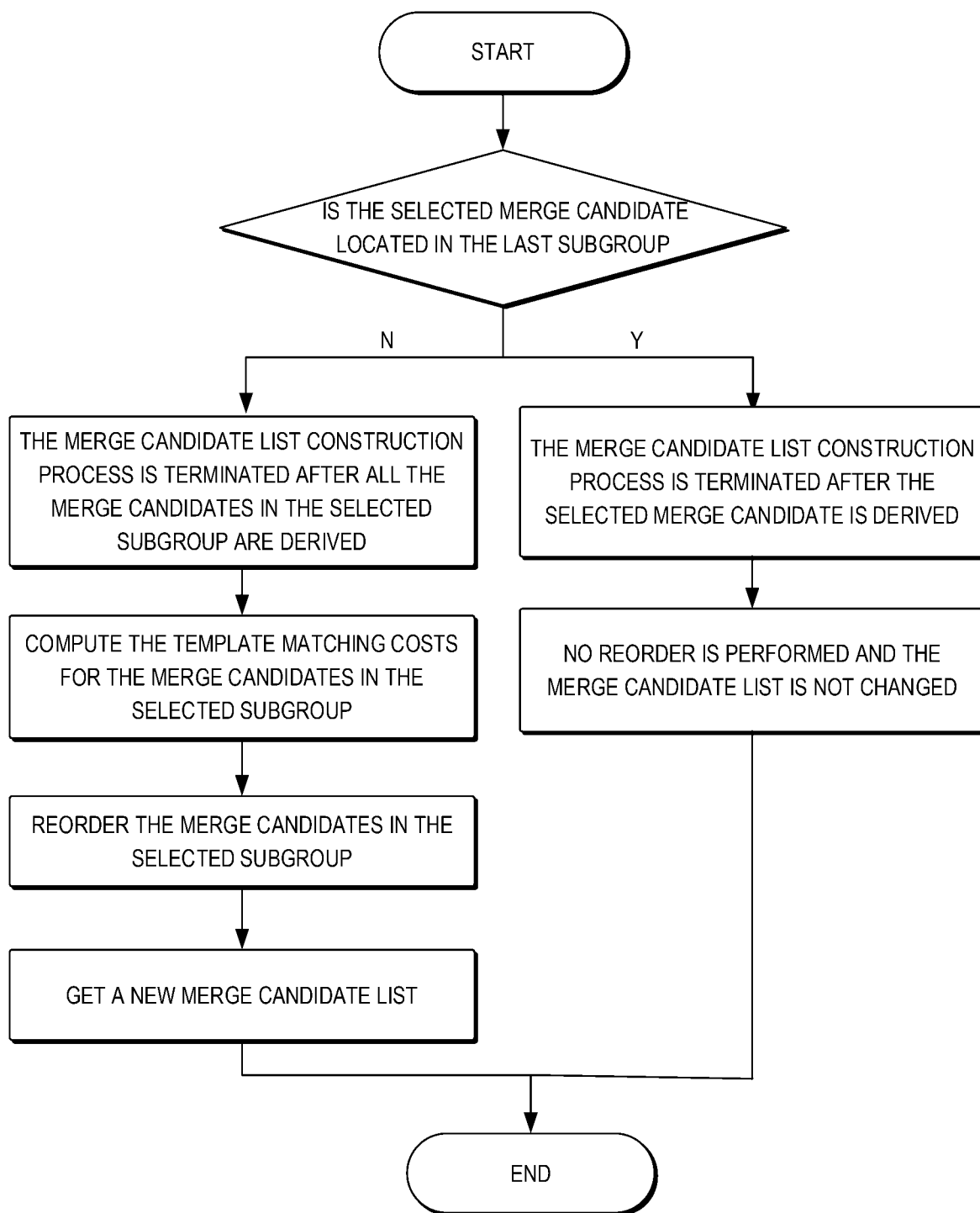


Fig. 40

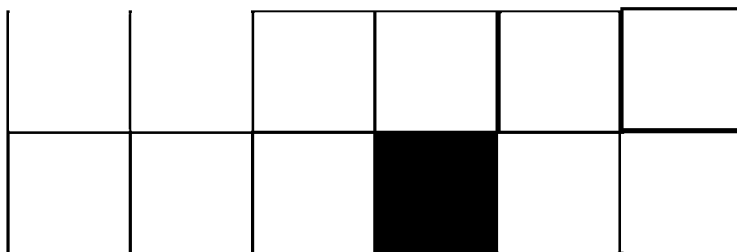


Fig. 41

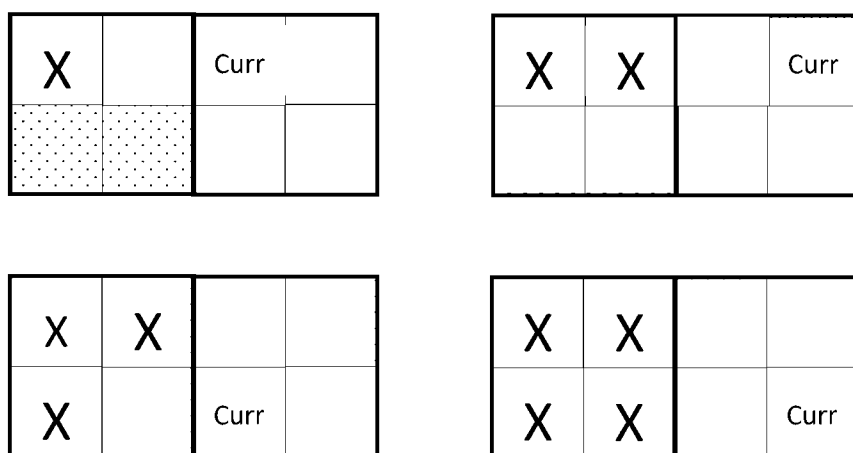


Fig. 42

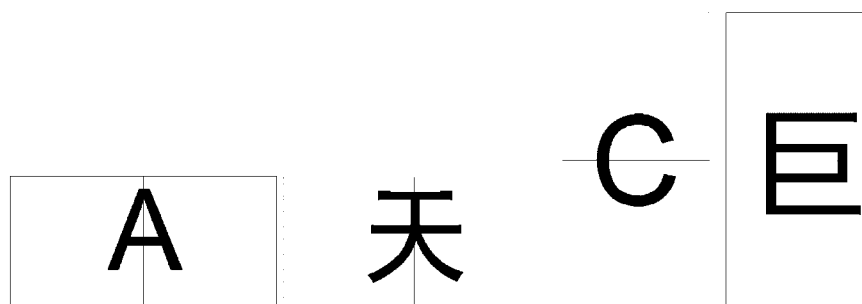


Fig. 43

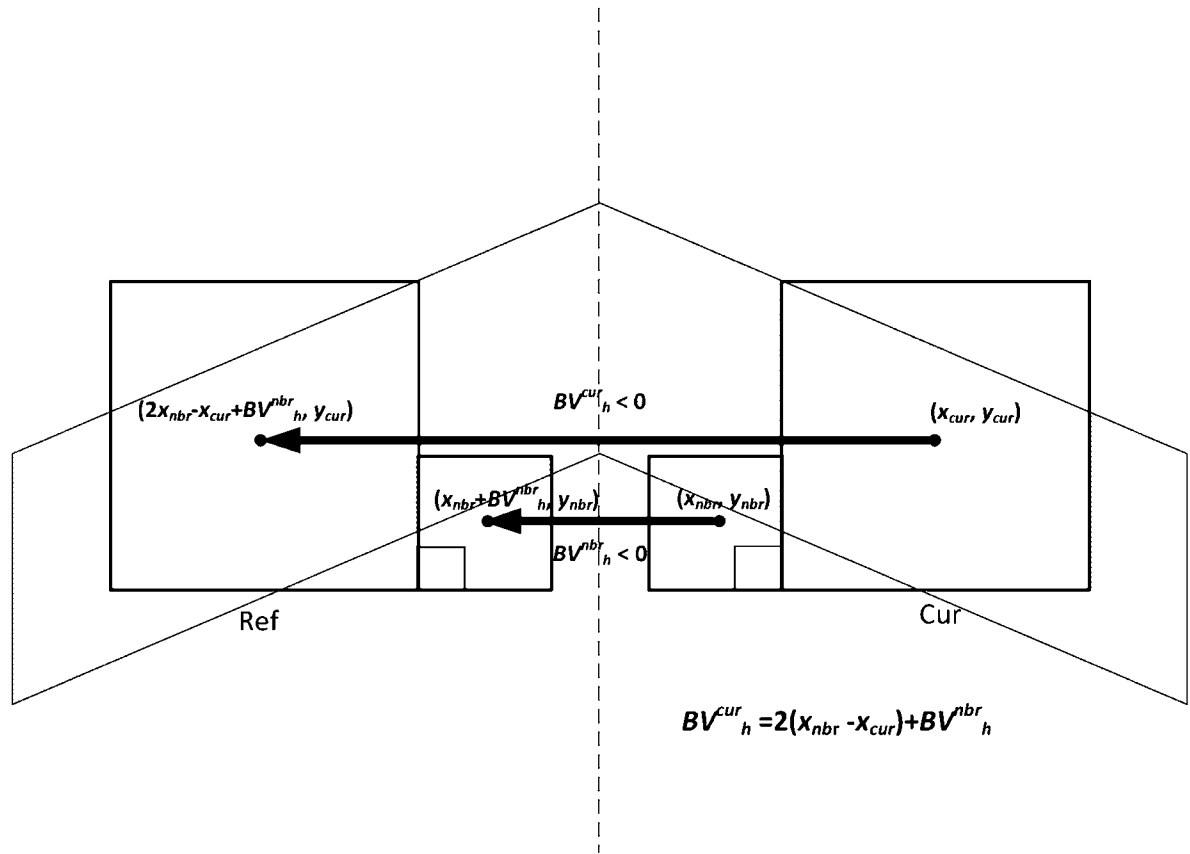


Fig. 44a

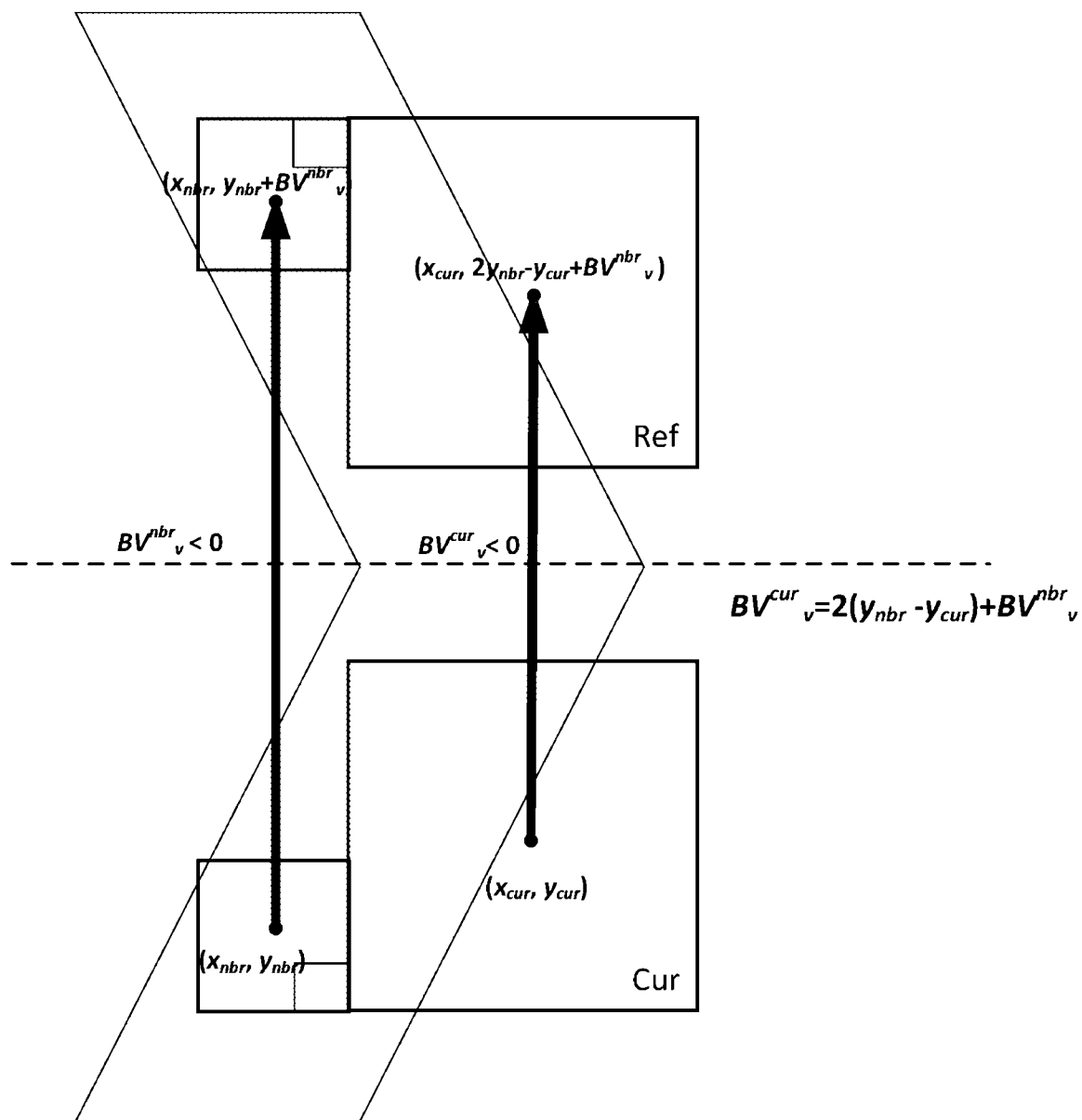


Fig. 44b

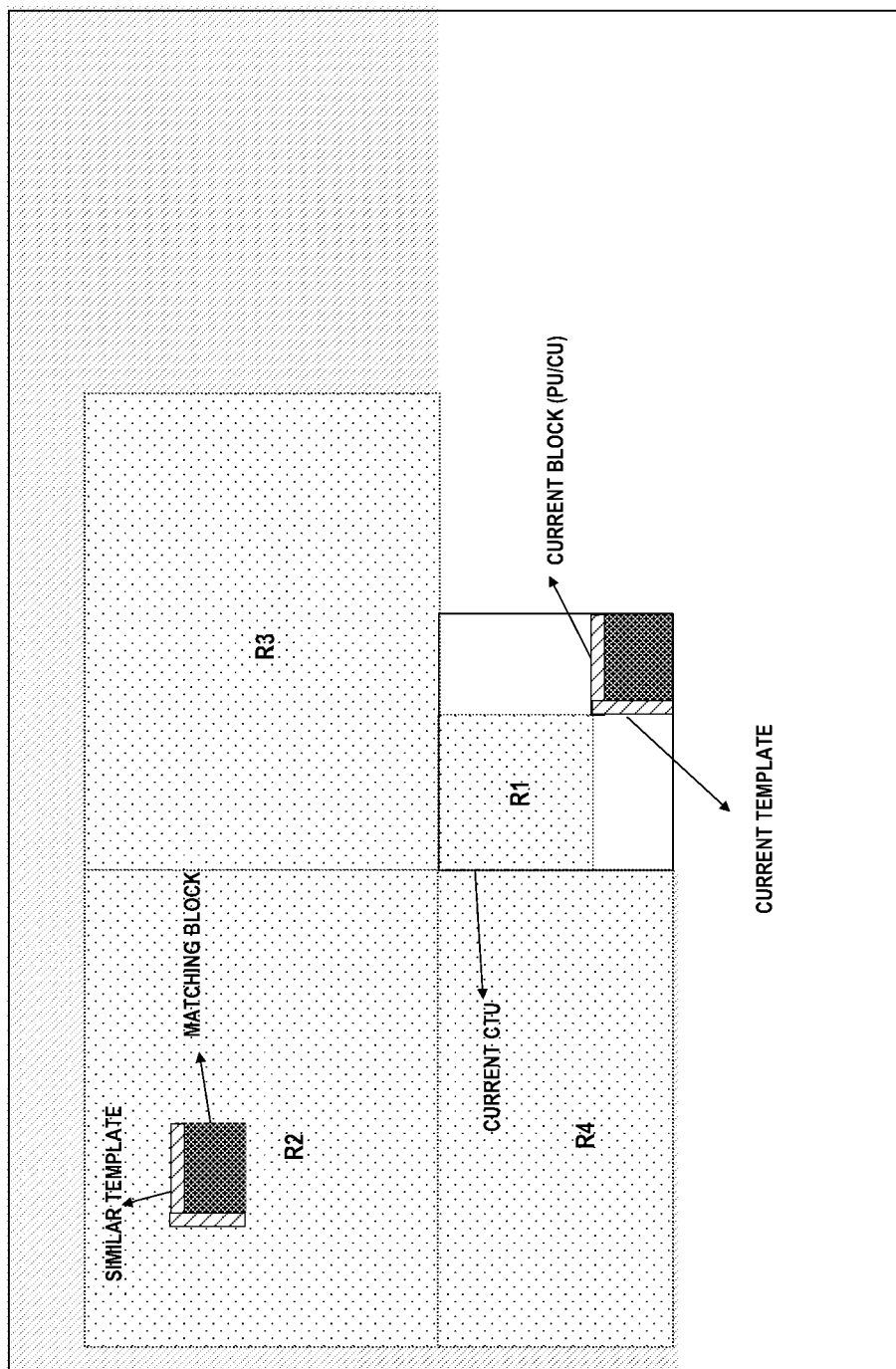


Fig. 45

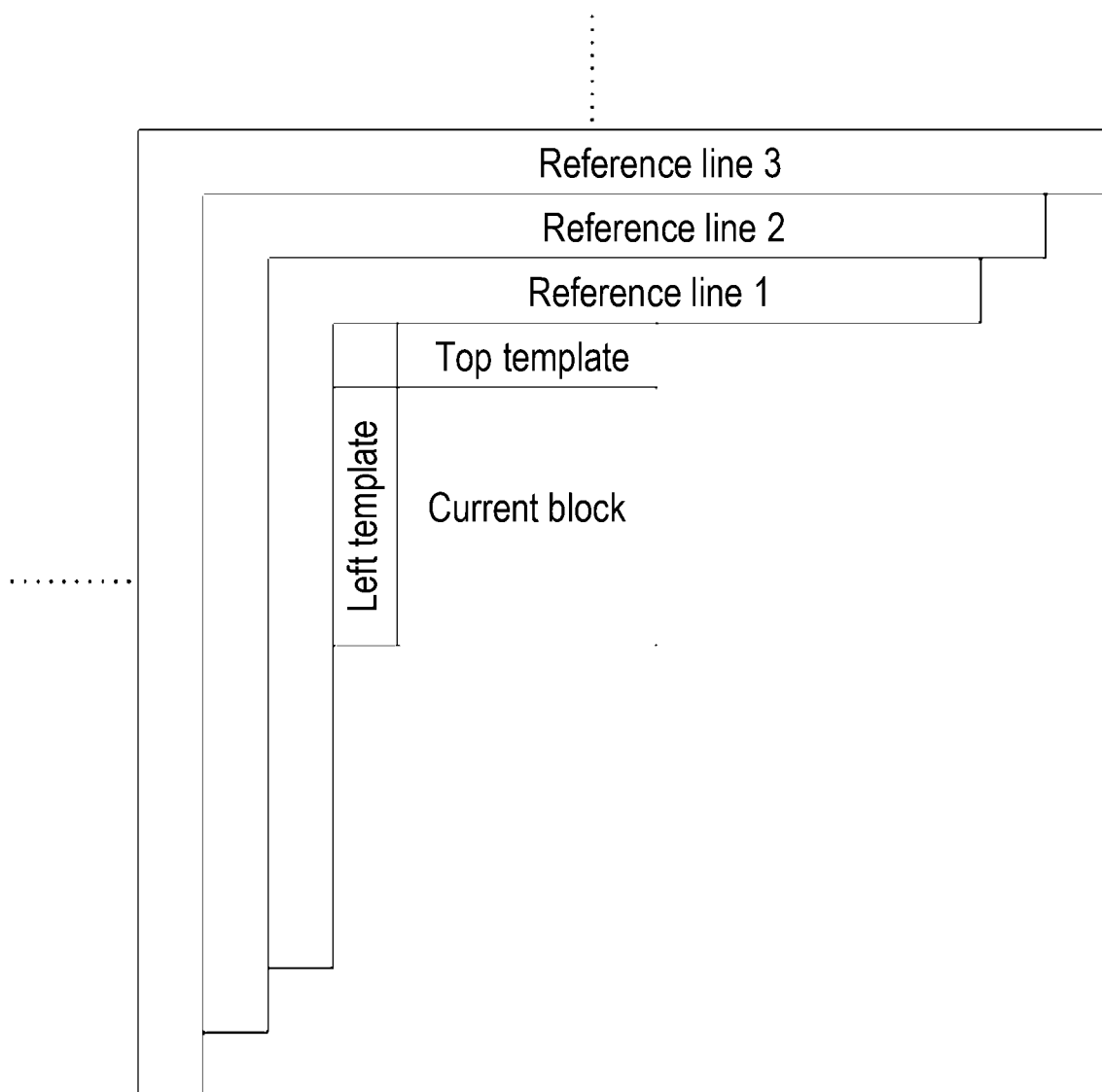


Fig. 46

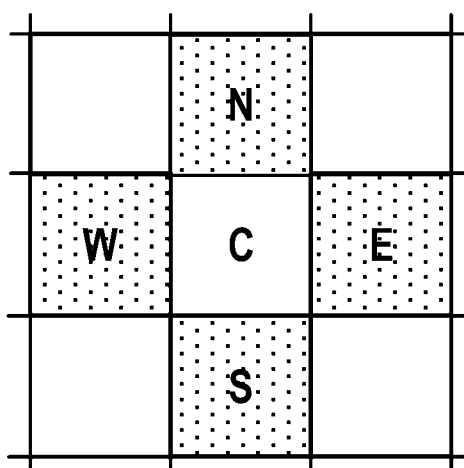


Fig. 47

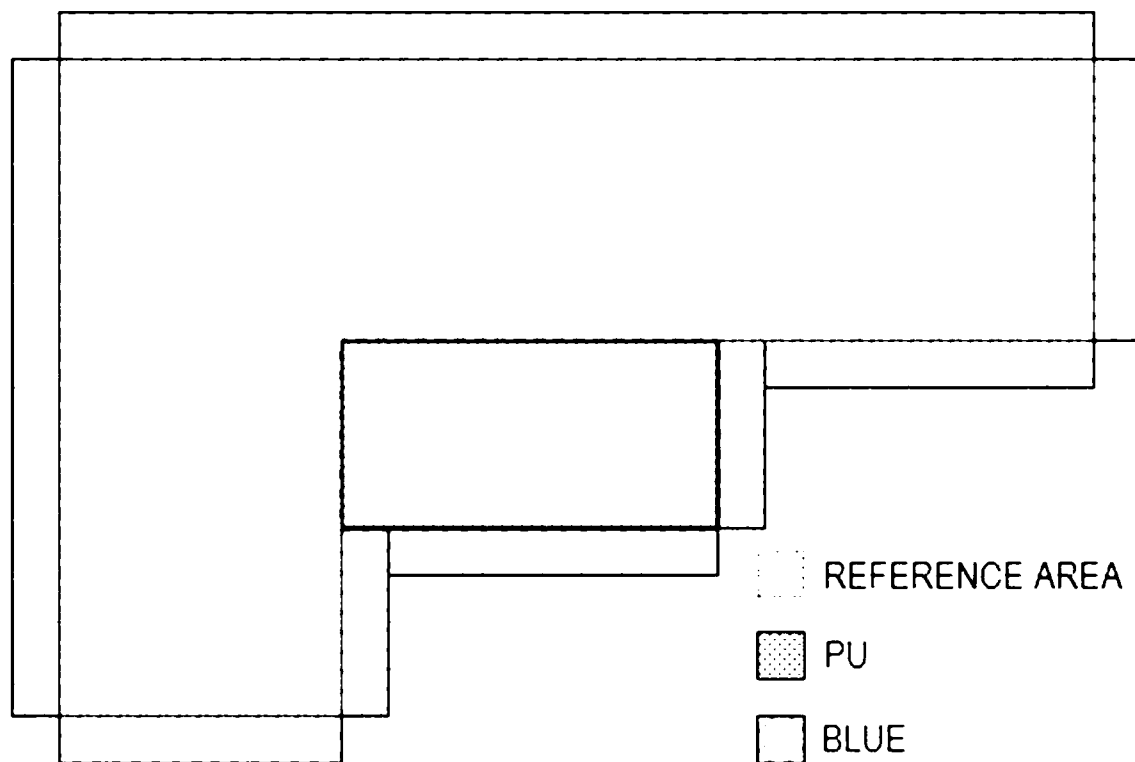


Fig. 48

1	0	-1		1	2	1		2	1	-1		-1	1	2
1	0	-1		-1	-2	-1		1	-1	-2		-2	-1	1

Fig. 49

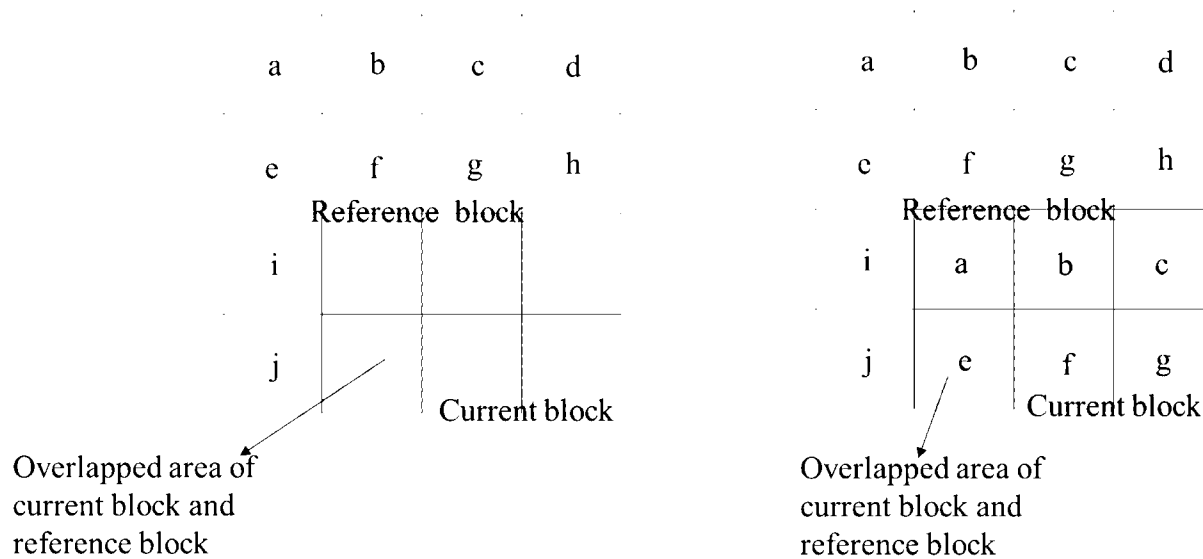


Fig. 50

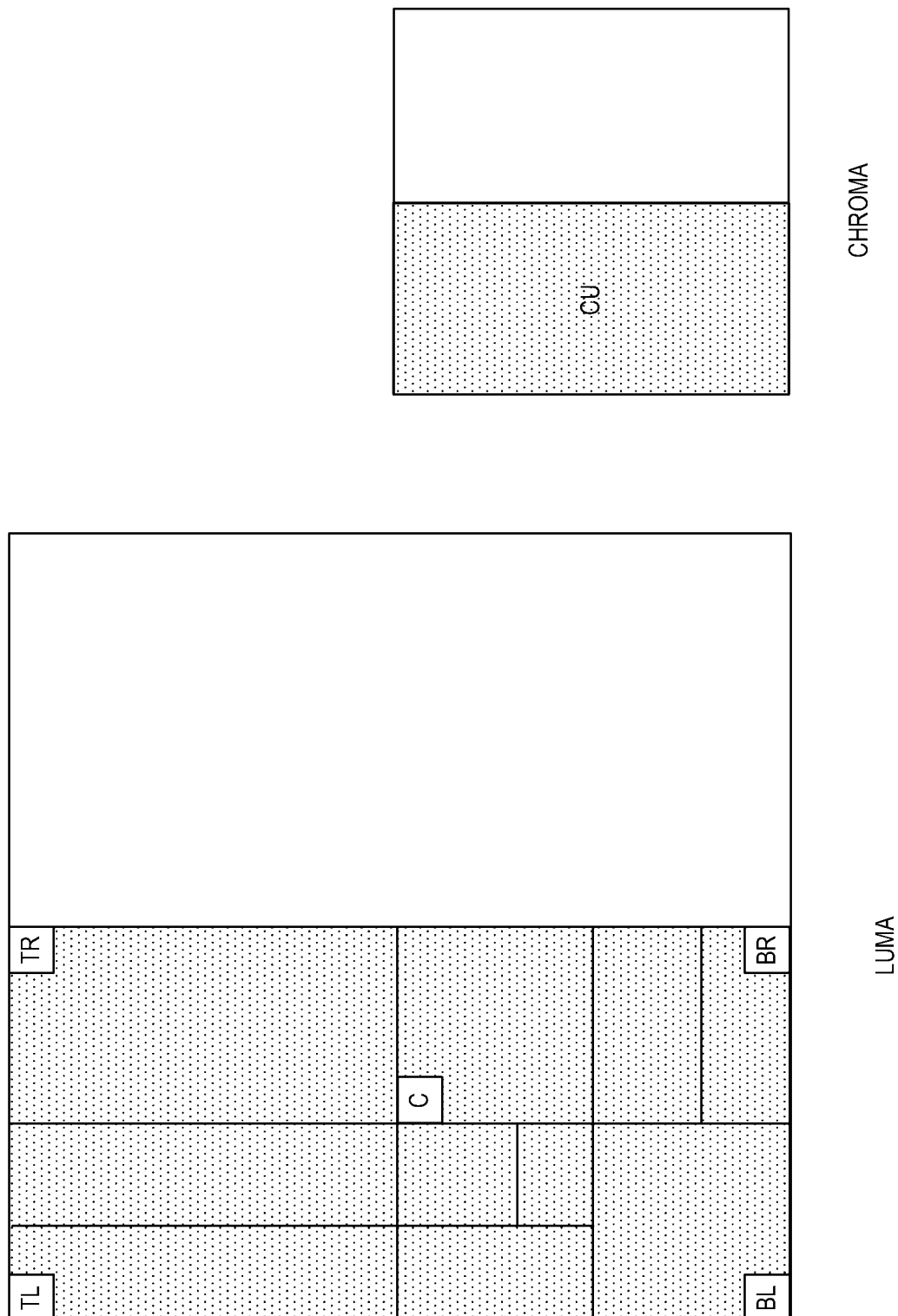


Fig. 51

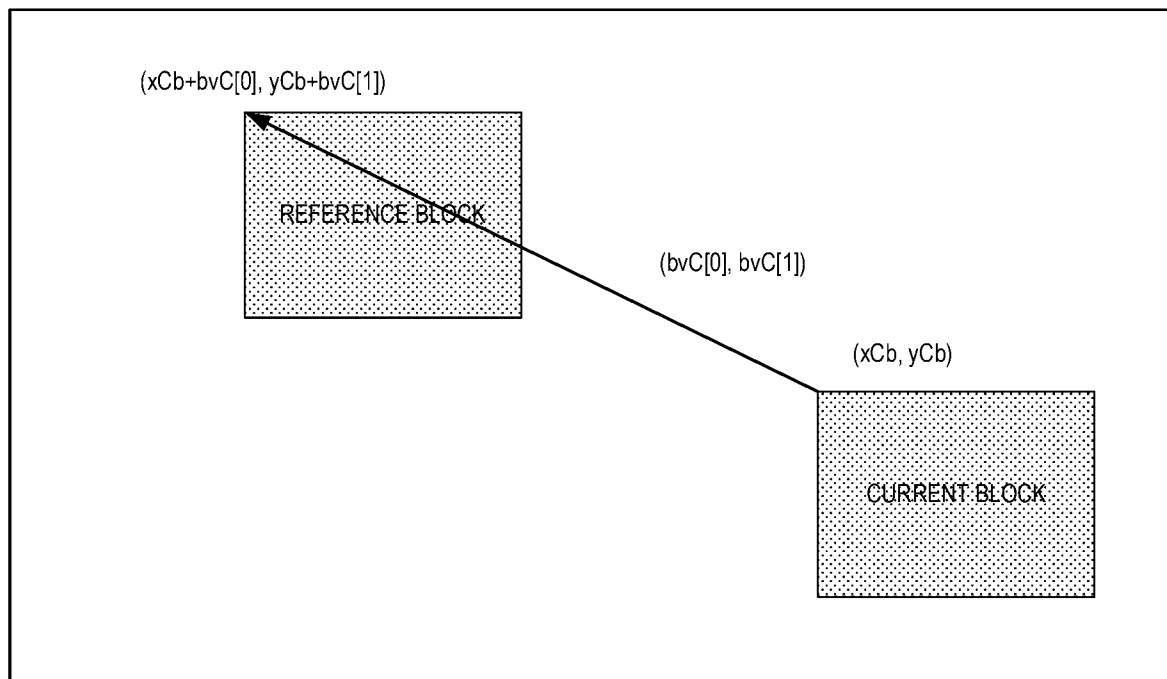


Fig. 52

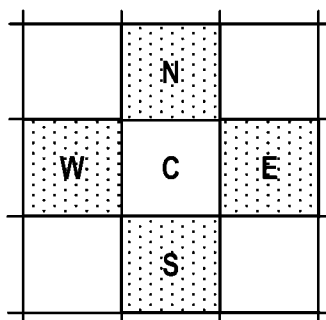


Fig. 53

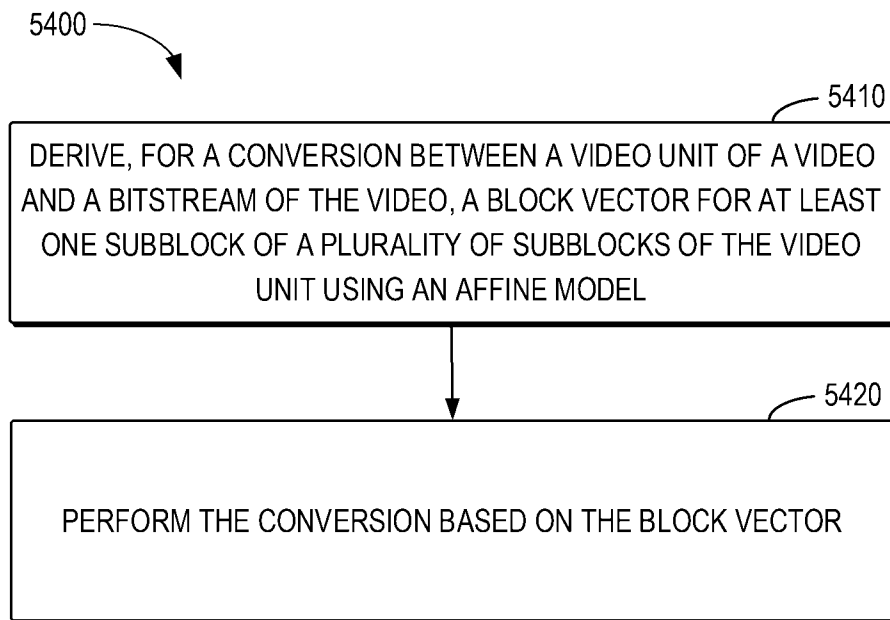


Fig. 54

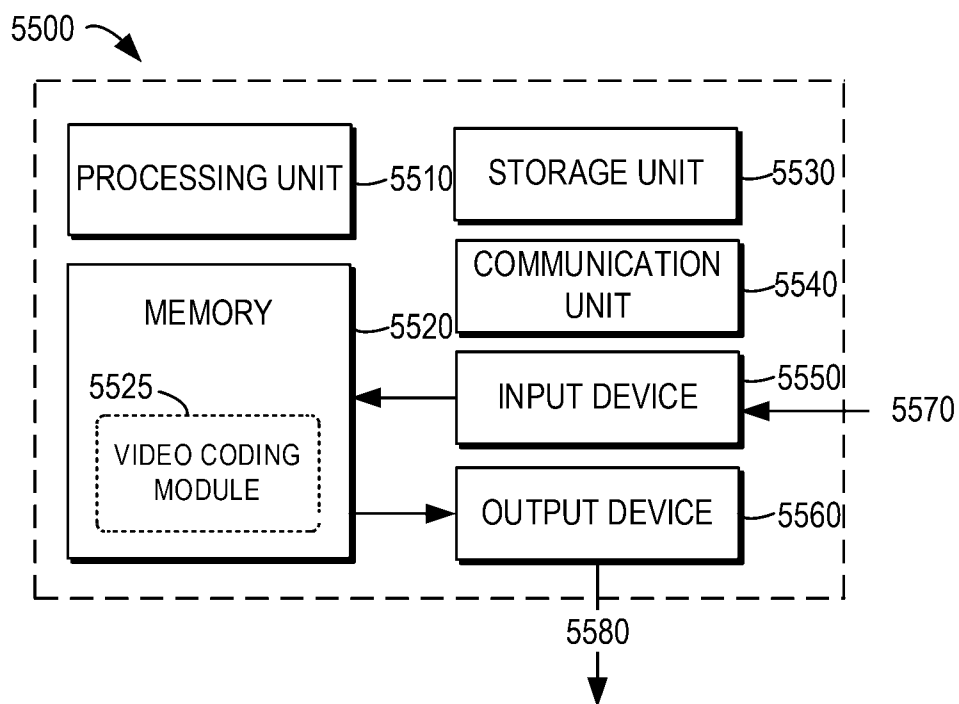


Fig. 55

INTERNATIONAL SEARCH REPORT

International application No.

PCT/CN2024/097635

A. CLASSIFICATION OF SUBJECT MATTER H04N19/149(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) IPC: 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) CNABS, CNTXT, CNKI, VEN, EPTXT, WOTXT, USTXT, JVT: video, encode, decode, block vector, BV, IBC, CPBV, affine, mode, parameter, position, following, adjacent, neighbor, unit, block, apply, allow		
C. DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	CN 111327901 A (BEIJING DAJIA INTERCONNECTION INFORMATION TECHNOLOGY CO., LTD.) 23 June 2020 (2020-06-23) description paragraphs [0097]-[0175], [0199]-[0208], figures 4-10	1-19, 27-57
A	CN 114245136 A (BEIJING DAJIA INTERCONNECTION INFORMATION TECHNOLOGY CO., LTD.) 25 March 2022 (2022-03-25) the whole document	1-57
A	US 2019246110 A1 (TENCENT AMERICA LLC) 08 August 2019 (2019-08-08) the whole document	1-57
A	US 2023011286 A1 (BEIJING DAJIA INTERNET INFORMATION TECH CO LTD) 12 January 2023 (2023-01-12) the whole document	1-57
☐ 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 “D” document cited by the applicant in the international application “E” earlier application or patent but published on or after the international filing date “L” document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified) “O” document referring to an oral disclosure, use, exhibition or other means “P” document published prior to the international filing date but later than the priority date claimed “T” later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention “X” document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone “Y” document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art “&” document member of the same patent family		
Date of the actual completion of the international search 28 September 2024		Date of mailing of the international search report 29 September 2024
Name and mailing address of the ISA/CN CHINA NATIONAL INTELLECTUAL PROPERTY ADMINISTRATION 6, Xitucheng Rd., Jimen Bridge, Haidian District, Beijing 100088, China		Authorized officer ZHANG, Wei Telephone No. (+86) 010-62411445

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2024/097635

Patent document cited in search report			Publication date (day/month/year)	Patent family member(s)			Publication date (day/month/year)
CN	111327901	A	23 June 2020	None			
CN	114245136	A	25 March 2022	None			
US	2019246110	A1	08 August 2019	US	11202079	B2	14 December 2021
				US	2022014757	A1	13 January 2022
				US	11516479	B2	29 November 2022
				WO	2019150352	A1	08 August 2019
				EP	3750314	A1	16 December 2020
US	2023011286	A1	12 January 2023	EP	4122203	A1	25 January 2023
				WO	2021188876	A1	23 September 2021