



(51) International Patent Classification:
H04N 19/52 (2014.01)

(21) International Application Number:
PCT/CN2024/097636

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

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
PCT/CN2023/098962
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,

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

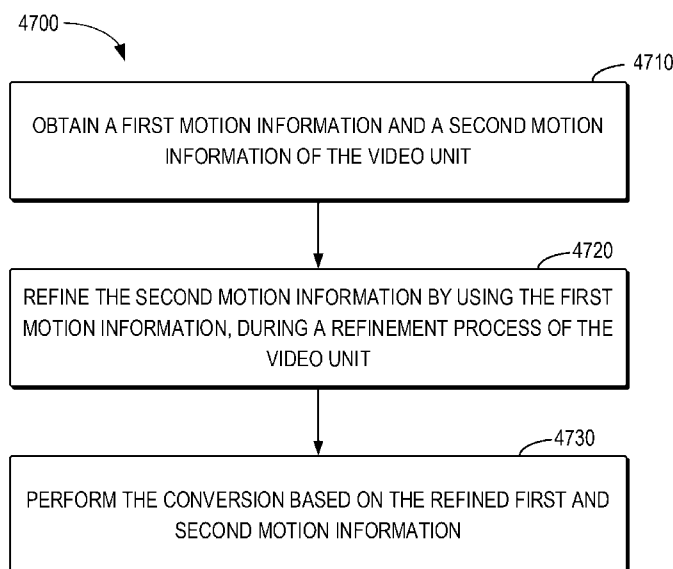


Fig. 47

(57) Abstract: Embodiments of the present disclosure provide a solution for video processing. A method for video processing is proposed. The method comprises: obtaining, for a conversion between a video unit of a video and a bitstream of the video unit, a first motion information and a second motion information of the video unit; refining the second motion information by using the first motion information, during a refinement process of the video unit, wherein the refinement process is a refinement or an iterative refinement; and performing the conversion based on the refined first and second motion information.



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, 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))*

METHOD, APPARATUS, AND MEDIUM FOR VIDEO PROCESSING

FIELDS

[0001] Embodiments of the present disclosure relates generally to video processing
5 techniques, and more particularly, to improvement on template matching.

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),
10 ITU-TH.265 high efficiency video coding (HEVC) standard, versatile video coding (VVC)
standard, have been proposed for video encoding/decoding. However, 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.

15 [0004] In a first aspect, a method for video processing is proposed. The method
comprises: obtaining, for a conversion between a video unit of a video and a bitstream of
the video unit, a first motion information and a second motion information of the video
unit; refining the second motion information by using the first motion information, during
a refinement process of the video unit, wherein the refinement process is a refinement or
20 an iterative refinement; and performing the conversion based on the refined first and
second motion information. In this way, it can improve coding performances.

[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
25 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: obtaining a first motion information and a second motion information of a video unit of the video; refining the second motion information by using the first motion information, during a refinement process of the video unit, wherein the refinement process is a refinement or an iterative refinement; and generating the bitstream based on the refined first and second motion information.

[0008] In a fifth aspect, a method for storing a bitstream of a video is proposed. The method comprises: obtaining a first motion information and a second motion information of a video unit of the video; refining the second motion information by using the first motion information, during a refinement process of the video unit, wherein the refinement process is a refinement or an iterative refinement; generating the bitstream based on the refined first and second motion information; 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 illustrates an example of encoder block diagram of VVC;
- [0015] Fig. 5 illustrates 67 intra prediction modes;
- [0016] Fig. 6A and Fig. 6B illustrate reference samples for wide-angular intra prediction;
- 5 [0017] Fig. 7 illustrates Problem of discontinuity in case of directions beyond 45°;
- [0018] Fig. 8A and Fig. 8B illustrate MMVD Search Point;
- [0019] Fig. 9 illustrates Illustration for symmetrical MVD mode;
- [0020] Fig. 10 illustrates extended CU region used in BDOF;
- [0021] Fig. 11 illustrates top and left neighboring blocks used in CIIP weight derivation;
- 10 [0022] Fig. 12A and Fig. 12B illustrate control point based affine motion model;
- [0023] Fig. 13 illustrates Affine MVF per subblock;
- [0024] Fig. 14 illustrates locations of inherited affine motion predictors;
- [0025] Fig. 15 illustrates control point motion vector inheritance;
- [0026] Fig. 16 illustrates locations of Candidates position for constructed affine merge
15 mode;
- [0027] Fig. 17 illustrates illustration of motion vector usage for proposed combined method;
- [0028] Fig. 18 illustrates Subblock MV VSB and pixel $\Delta v(i,j)$ (red arrow);
- [0029] Fig. 19A and Fig. 19B illustrate the SbTMVP process in VVC;
- 20 [0030] Fig. 20 illustrates local illumination compensation;
- [0031] Fig. 21 illustrates no subsampling for the short side;
- [0032] Fig. 22 illustrates decoding side motion vector refinement;
- [0033] Fig. 23 illustrates diamond regions in the search area;
- [0034] Fig. 24 illustrates positions of spatial merge candidate;
- 25 [0035] Fig. 25 illustrates candidate pairs considered for redundancy check of spatial merge candidates;

- [0036] Fig. 26 illustrates illustration of motion vector scaling for temporal merge candidate;
- [0037] Fig. 27 illustrates Candidate positions for temporal merge candidate, C0 and C1;
- [0038] Fig. 28 illustrates VVC spatial neighboring blocks of the current block;
- 5 [0039] Fig. 29 illustrates illustration of virtual block in the i-th search round;
- [0040] Fig. 30 illustrates examples of the GPM splits grouped by identical angles;
- [0041] Fig. 31 illustrates uni-prediction MV selection for geometric partitioning mode;
- [0042] Fig. 32 illustrates exemplified generation of a bending weight w_0 using geometric partitioning mode;
- 10 [0043] Fig. 33 illustrates spatial neighboring blocks used to derive the spatial merge candidates;
- [0044] Fig. 34 illustrates template matching performs on a search area around initial MV;
- [0045] Fig. 35 illustrates illustration of sub-blocks where OBMC applies;
- 15 [0046] Fig. 36 illustrates SBT position, type and transform type;
- [0047] Fig. 37 illustrates neighbouring samples used for calculating SAD;
- [0048] Fig. 38 illustrates neighbouring samples used for calculating SAD for sub-CU level motion information;
- [0049] Fig. 39 illustrates the sorting process;
- 20 [0050] Fig. 40 illustrates reorder process in encoder;
- [0051] Fig. 41 illustrates reorder process in decoder;
- [0052] Fig. 42 illustrates first HPT and the second HPT;
- [0053] Fig. 43A and Fig. 43B illustrate spatial neighbors for deriving affine merge/AMVP candidates;
- 25 [0054] Fig. 44 illustrates a diagram from non-adjacent neighbors to the first type of constructed affine merge/AMVP candidates;

[0055] Fig. 45 illustrates different numbers of searching points in a same search pattern;

[0056] Fig. 46 illustrates different numbers of searching points in different search patterns;

[0057] Fig. 47 illustrates a flowchart of a method for video processing in accordance
5 with embodiments of the present disclosure; and

[0058] Fig. 48 illustrates a block diagram of a computing device in which various embodiments of the present disclosure can be implemented.

[0059] Throughout the drawings, the same or similar reference numerals usually refer to the same or similar elements.

10 DETAILED DESCRIPTION

[0060] 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.

15 The disclosure described herein can be implemented in various manners other than the ones described below.

[0061] 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.

20 [0062] 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
25 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.

[0063] It shall be understood that although the terms “first” and “second” etc. may be
30 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
5 listed terms.

[0064] 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
10 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

[0065] 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
15 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.
20

[0066] 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.
25

[0067] 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
30 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.

5 The encoded video data may also be stored onto a storage medium/server 130B for access by destination device 120.

[0068] 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
10 110 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.

15 [0069] 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.

[0070] 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
20 accordance with some embodiments of the present disclosure.

[0071] 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
25 processor may be configured to perform any or all of the techniques described in this disclosure.

[0072] 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
30 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.

[0073] 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.

5 [0074] 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.

[0075] 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.

10 [0076] 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)
15 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.

[0077] To perform inter prediction on a current video block, the motion estimation unit
20 204 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.

25 [0078] 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”
30 and “B-slices” may refer to portions of a picture composed of macroblocks that are not dependent on macroblocks in the same picture.

[0079] 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.

[0080] 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.

[0081] 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.

[0082] 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.

[0083] 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.

[0084] 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.

[0085] 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.

[0086] 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.

[0087] 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.

[0088] 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.

[0089] 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.

[0090] 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
5 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.

[0091] After the reconstruction unit 212 reconstructs the video block, loop filtering operation may be performed to reduce video blocking artifacts in the video block.

10 [0092] 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.

15 [0093] 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.

[0094] 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
20 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.

[0095] 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
25 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.

[0096] 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
30 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.

[0097] 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.

[0098] 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.

[0099] 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.

[0100] 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.

[0101] 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.

[0102] 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

[0103] This disclosure is related to video coding technologies. Specifically, it is related to template matching for inter prediction, and how to apply, 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

[0104] Video coding standards have evolved primarily through the development of the well-known ITU-T and ISO/IEC standards. The ITU-T produced H.261 and H.263,

ISO/IEC produced MPEG-1 and MPEG-4 Visual, and the two organizations jointly produced the H.262/MPEG-2 Video and H.264/MPEG-4 Advanced Video Coding (AVC) and H.265/HEVC standards. Since H.262, the video coding standards are based on the hybrid video coding structure wherein temporal prediction plus transform coding are utilized. To explore the future video coding technologies beyond HEVC, Joint Video Exploration Team (JVET) was founded by VCEG and MPEG jointly in 2015. Since then, many new methods have been adopted by JVET and put into the reference software named Joint Exploration Model (JEM). In April 2018, the Joint Video Expert Team (JVET) between VCEG (Q6/16) and ISO/IEC JTC1 SC29/WG11 (MPEG) was created to work on the VVC standard targeting at 50% bitrate reduction compared to HEVC.

2.1. Coding flow of a typical video codec

[0105] 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

[0106] 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.

[0107] 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

[0108] 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.

[0109] 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. 6A and Fig. 6B.

10 [0110] 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 1.

Table 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

[0111] 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 $\Delta p\alpha$. 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.

[0112] In VVC, 4:2:2 and 4:4:4 chroma formats are supported as well as 4:2:0. Chroma
5 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
10 by replacing some values of the entries of the mapping table to convert prediction angle
more precisely for chroma blocks.

2.3. Inter prediction

[0113] For each inter-predicted CU, motion parameters consisting of motion vectors,
reference picture indices and reference picture list usage index, and additional information
15 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
20 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
25 are signalled explicitly per each CU.

2.4. Intra block copy (IBC)

[0114] 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
30 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.

5 [0115] 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.

10 [0116] 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
15 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.

[0117] In block matching search, the search range is set to cover both the previous and current CTUs.

20 [0118] 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.

25 – 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.

30 2.5. Merge mode with MVD (MMVD)

[0119] 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.

[0120] 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.

[0121] Distance index specifies motion magnitude information and indicate the pre-defined offset from the starting point. As shown in Fig. 8A and Fig. 8B, 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.

Table 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

[0122] 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 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 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 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 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.

[0123] 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 Fig. 26. 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 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	+	–

5 2.6. Symmetric MVD coding

[0124] 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
10 derived.

[0125] 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
20 is explicitly signaled if the CU is bi-prediction coded and BiDirPredFlag is equal to 1.

[0126] 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.

$$25 \quad \begin{cases} (mvx_0, mvy_0) = (mvp_x_0 + mvd_x_0, mvp_y_0 + mvd_y_0) \\ (mvx_1, mvy_1) = (mvp_x_1 - mvd_x_0, mvp_y_1 - mvd_y_0) \end{cases} \quad (2-1)$$

Fig. 9 illustrates an illustration for symmetrical MVD mode.

[0127] In the encoder, symmetric MVD motion estimation starts with initial MV
30 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.

2.7. Bi-directional optical flow (BDOF)

5 [0128] 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.

[0129] BDOF is used to refine the bi-prediction signal of a CU at the 4×4 subblock level.

10 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
15 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.
- 20 – BCW weight index indicates equal weight.
- WP is not enabled for the current CU.
- CIIP mode is not used for the current CU.

[0130] 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
25 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.

[0131] 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
30 the two prediction signals are computed by directly calculating the difference between two neighboring samples, i.e.,

$$\frac{\partial I^{(k)}}{\partial x}(i, j) = ((I^{(k)}(i + 1, j) \gg \text{shift1}) - (I^{(k)}(i - 1, j) \gg \text{shift1})) \quad (2-2)$$

$$\frac{\partial I^{(k)}}{\partial y}(i, j) = ((I^{(k)}(i, j + 1) \gg \text{shift1}) - (I^{(k)}(i, j - 1) \gg \text{shift1}))$$

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 $\text{shift1} = \max(6, \text{bitDepth} - 6)$.

Then, the auto- and cross-correlation of the gradients, S_1, S_2, S_3, S_5 and S_6 , are calculated as

$$\begin{aligned} S_1 &= \sum_{(i,j) \in \Omega} \text{Abs}(\psi_x(i, j)), \quad S_3 = \sum_{(i,j) \in \Omega} \theta(i, j) \cdot \text{Sign}(\psi_x(i, j)) \\ S_2 &= \sum_{(i,j) \in \Omega} \psi_x(i, j) \cdot \text{Sign}(\psi_y(i, j)) \\ S_5 &= \sum_{(i,j) \in \Omega} \text{Abs}(\psi_y(i, j)), \quad S_6 = \sum_{(i,j) \in \Omega} \theta(i, j) \cdot \text{Sign}(\psi_y(i, j)) \end{aligned} \quad (2-3)$$

5 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.

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

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

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

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

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

Finally, the BDOF samples of the CU are calculated by adjusting the bi-prediction samples as follows:

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

[0132] 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.

[0133] 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. 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.

[0134] 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 $(8 * W * (H >> 1))$, 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.

[0135] 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.

2.8. Combined inter and intra prediction (CIIP)

[0136] In VVC, when a CU is coded in merge mode, if the CU contains at least 64 luma samples (that is, CU width times CU height is equal to or larger than 64), and if both CU width and CU height are less than 128 luma samples, an additional flag is signalled to indicate if the combined inter/intra prediction (CIIP) mode is applied to the current CU. As its name indicates, the CIIP prediction combines an inter prediction signal with an intra prediction signal. The inter prediction signal in the CIIP mode P_{inter} is derived using the same inter prediction process applied to regular merge mode; and the intra prediction signal P_{intra} is derived following the regular intra prediction process with the planar mode.

Then, the intra and inter prediction signals are combined using weighted averaging, where the weight value is calculated depending on the coding modes of the top and left neighbouring blocks (depicted in Fig. 11) as follows:

- 5 – If the top neighbor is available and intra coded, then set isIntraTop to 1, otherwise set isIntraTop to 0;
- If the left neighbor is available and intra coded, then set isIntraLeft to 1, otherwise set isIntraLeft to 0;
- If (isIntraLeft + isIntraTop) is equal to 2, then wt is set to 3;
- 10 – Otherwise, if (isIntraLeft + isIntraTop) is equal to 1, then wt is set to 2;
- Otherwise, set wt to 1.

The CIIP prediction is formed as follows:

$$P_{CIIP} = ((4 - wt) * P_{inter} + wt * P_{intra} + 2) \gg 2 \quad (2-8).$$

2.9. Affine motion compensated prediction

[0137] 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
15 in/out, rotation, perspective motions and the other irregular motions. In VVC, a block-based affine transform motion compensation prediction is applied. As shown in Fig. 12A which illustrates 4-parameter affine model and Fig. 12B which illustrates 6-parameter affine model, 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).

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

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

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

$$\begin{cases} mv_x = \frac{mv_{1x} - mv_{0x}}{W}x + \frac{mv_{2x} - mv_{0x}}{H}y + mv_{0x} \\ mv_y = \frac{mv_{1y} - mv_{0y}}{W}x + \frac{mv_{2y} - mv_{0y}}{H}y + mv_{0y} \end{cases} \quad (2-10)$$

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.

[0138] In order to simplify the motion compensation prediction, block based affine
30 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. 13, 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.

- 5 [0139] As done for translational motion inter prediction, there are also two affine motion inter prediction modes: affine merge mode and affine AMVP mode.

2.9.1. Affine merge prediction

[0140] 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.
- 15 – Constructed affine merge candidates CPMVPs that are derived using the translational MVs of the neighbour CUs.
- Zero MVs.

[0141] 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. The candidate blocks are shown in Fig. 14. 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 .

[0142] Fig. 15 illustrates control point motion vector inheritance.

[0143] Constructed affine candidate means the candidate is constructed by combining the neighbour translational motion information of each control point. The motion

information for the control points is derived from the specified spatial neighbours and temporal neighbour shown in Fig. 16. $CPMV_k$ ($k=1, 2, 3, 4$) 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.

[0144] 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:

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

[0145] 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.

[0146] 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.

2.9.2. Affine AMVP prediction

[0147] 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.

[0148] 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.

[0149] Constructed AMVP candidate is derived from the specified spatial neighbours shown in Fig. 16. The same checking order is used as done in affine merge candidate construction. In addition, reference picture index of the neighbouring block is also
 5 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
 10 affine AMVP list. Otherwise, constructed AMVP candidate is set as unavailable.

[0150] 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
 15 not full.

2.9.3. Affine motion information storage

[0151] 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
 20 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.

[0152] 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
 25 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. As shown in Fig. 17, along the top CTU
 30 boundary, the bottom-left and bottom right subblock motion vectors of a CU are used for affine inheritance of the CUs in bottom CTUs.

2.9.4. Prediction refinement with optical flow for affine mode

[0153] 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.

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

$$g_y(i, j) = (I(i, j + 1) \gg \text{shift1}) - (I(i, j - 1) \gg \text{shift1}) \quad (2-12)$$

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.

$$\Delta I(i, j) = g_x(i, j) * \Delta v_x(i, j) + g_y(i, j) * \Delta v_y(i, j) \quad (2-13)$$

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. 18. The $\Delta v(i, j)$ is quantized in the unit of 1/32 luma sample precision.

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,

$$\begin{cases} dx(i, j) = i - x_{SB} \\ dy(i, j) = j - y_{SB} \end{cases} \quad (2-14)$$

$$\begin{cases} \Delta v_x(i, j) = C * dx(i, j) + D * dy(i, j) \\ \Delta v_y(i, j) = E * dx(i, j) + F * dy(i, j) \end{cases} \quad (2-15)$$

In order to keep accuracy, the center 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.

For 4-parameter affine model,

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

For 6-parameter affine model,

$$\begin{cases} C = \frac{v_{1x}-v_{0x}}{w} \\ D = \frac{v_{2x}-v_{0x}}{h} \\ E = \frac{v_{1y}-v_{0y}}{w} \\ F = \frac{v_{2y}-v_{0y}}{h} \end{cases} \quad (2-17)$$

where (v_{0x}, v_{0y}) , (v_{1x}, v_{1y}) , (v_{2x}, v_{2y}) are the top-left, top-right and bottom-left control point

5 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.

$$I'(i, j) = I(i, j) + \Delta I(i, j) \quad (2-18)$$

[0154] PROF is not be applied in two cases for an affine coded CU: 1) all control point

10 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.

[0155] 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

15 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

20 PROF is small for this case. In this way, the affine motion estimation with PROF can be accelerated.

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

[0156] VVC supports the subblock-based temporal motion vector prediction (SbTMVP) method. Similar to the temporal motion vector prediction (TMVP) in HEVC, SbTMVP

25 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 SbTVMP. 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
- 30 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.

[0157] The SbTMVP process is illustrated in Fig. 19A and Fig. 19B. 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 Fig. 19A 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).

[0158] 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 Fig. 19B. The example in Fig. 19B 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.

[0159] Fig. 19 B illustrates deriving sub-CU motion field by applying a motion shift from spatial neighbor and scaling the motion information from the corresponding collocated sub-CUs.

[0160] 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.

[0161] 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.

[0162] 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.

2.11. Adaptive motion vector resolution (AMVR)

[0163] 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 AVMP 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.

[0164] 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.

[0165] 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).

[0166] 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.

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

[0167] 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.

$$P_{\text{bi-pred}} = ((8 - w) * P_0 + w * P_1 + 4) \gg 3 \quad (2-19)$$

[0168] 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.

[0169] 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.

[0170] 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.

[0171] 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.

2.13. Local illumination compensation (LIC)

[0172] 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. 20 illustrates the LIC process. In Fig. 20, 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. 20) and their corresponding reference samples in the temporal reference pictures (i.e., either $T0$ or $T1$ in Fig. 20). 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. 20 are used to derive α and β .

[0173] To improve the coding performance, no subsampling for the short side is performed as shown in Fig. 21.

2.14. Decoder side motion vector refinement (DMVR)

[0174] 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. 22, 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.

[0175] 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.
- 5 – WP is not enabled for the current block.
- CIIP mode is not used for the current block.

[0176] 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
10 motion vector prediction for future CU coding.

[0177] The additional features of DMVR are mentioned in the following sub-clauses.

2.14.1. Searching scheme

[0178] 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
15 DMVR, denoted by candidate MV pair (MV0, MV1) obey the following two equations:

$$MV0' = MV0 + MV_offset \quad (2-20)$$

$$MV1' = MV1 - MV_offset \quad (2-21)$$

[0179] 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
20 luma samples from the initial MV. The searching includes the integer sample offset search stage and fractional sample refinement stage.

[0180] 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
25 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.

30 [0181] 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.

- 5 [0182] 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

$$E(x, y) = A(x - x_{min})^2 + B(y - y_{min})^2 + C \quad (2-22)$$

- 10 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:

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

$$y_{min} = (E(0,-1) - E(0,1)) / (2(E(0,-1) + E(0,1) - 2E(0,0))) \quad (2-24).$$

- 15 [0183] 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.

2.14.2. Bilinear-interpolation and sample padding

- 20 [0184] 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
 25 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 DVMR 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
 30 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.

2.14.3. Maximum DMVR processing unit

[0185] 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.

5 2.15. Multi-pass decoder-side motion vector refinement

[0186] 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.

2.15.1. First pass – Block based bilateral matching MV refinement

[0187] 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
15 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 initiate MVs based on the minimum bilateral matching cost between the two reference blocks in L0 and L1.

[0188] BM performs local search to derive integer sample precision intDeltaMV and half-pel sample precision halfDeltaMv. The local search applies a 3x3 square search
20 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.

[0189] 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
25 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.

30 [0190] The existing fractional sample refinement is further applied to derive the final deltaMV. The refined MVs after the first pass are then derived as:

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

2.15.2. Second pass – Subblock based bilateral matching MV refinement

[0191] 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.

[0192] 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.

[0193] 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. 23. 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.

[0194] 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.

[0195] The existing VVC DMVR fractional sample refinement is further applied to derive the final $\Delta MV(sbIdx2)$. The refined MVs at second pass is then derived as:

- $MV0_pass2(sbIdx2) = MV0_pass1 + \Delta MV(sbIdx2)$,
- $MV1_pass2(sbIdx2) = MV1_pass1 - \Delta MV(sbIdx2)$.

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

[0196] 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 Vx and Vy without clipping starting from the refined MV of the parent subblock of the second pass. The derived bioMv(Vx, Vy) is rounded to 1/16 sample precision and clipped between -32 and 32.

[0197] The refined MVs (MV0_pass3(sbIdx3) and MV1_pass3(sbIdx3)) at third pass are derived as:

- MV0_pass3(sbIdx3) = MV0_pass2(sbIdx2) + bioMv,
- MV1_pass3(sbIdx3) = MV0_pass2(sbIdx2) – bioMv.

2.16. Sample-based BDOF

[0198] In the sample-based BDOF, instead of deriving motion refinement (Vx, Vy) on a block basis, it is performed per sample.

- [0199] 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 Vx and Vy. The derived motion refinement (Vx, Vy) is applied to adjust the bi-predicted sample value for the center sample of the window.

2.17. Extended merge prediction

[0200] 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.

[0201] 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.

[0202] 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.

2.17.1. Spatial candidates derivation

[0203] The derivation of spatial merge candidates in VVC is same to that in HEVC
5 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. 24. 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
10 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. 25 are considered and a candidate is only added to the list if the
15 corresponding candidate used for redundancy check has not the same motion information.

2.17.2. Temporal candidates derivation

[0204] 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
20 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. 26, 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
25 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.

[0205] The position for the temporal candidate is selected between candidates C0 and C1, as depicted in Fig. 27. If CU at position C0 is not available, is intra coded, or is outside
30 of the current row of CTUs, position C1 is used. Otherwise, position C0 is used in the derivation of the temporal merge candidate.

2.17.3. History-based merge candidates derivation

[0206] 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.

[0207] 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.

[0208] 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.

[0209] To reduce the number of redundancy check operations, the following simplifications are introduced:

[0210] 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.

[0211] 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.

2.17.4. Pair-wise average merge candidates derivation

[0212] 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
5 invalid.

[0213] 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.

2.17.5. Merge estimation region

10 [0214] 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
15 $(x_{Cb} + cbWidth) \gg \text{Log2ParMrgLevel}$ is greater than $x_{Cb} \gg \text{Log2ParMrgLevel}$ and $(y_{Cb} + cbHeight) \gg \text{Log2ParMrgLevel}$ is great 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.

20 2.18. New merge candidates

2.18.1. Non-adjacent merge candidates derivation

[0215] In VVC, five spatially neighboring blocks shown in Fig. 28 as well as one temporal neighbor are used to derive merge candidates.

[0216] 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
25 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
30 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.

5 [0217] Fig. 29 illustrates the relationship between the virtual block and the current block. 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
10 neighboring blocks that are used in VVC merge mode.

[0218] 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.

[0219] Non-adjacent spatial merge candidates are inserted into the merge list after the
15 temporal merge candidate in the order of $B_1 \rightarrow A_1 \rightarrow C_1 \rightarrow D_1 \rightarrow E_1$.

2.18.2. STMVP

[0220] It is proposed to derive an averaging candidate as STMVP candidate using three spatial merge candidates and one temporal merge candidate.

[0221] STMVP is inserted before the above-left spatial merge candidate.

20 [0222] The STMVP candidate is pruned with all the previous merge candidates in the merge list.

[0223] For the spatial candidates, the first three candidates in the current merge candidate list are used.

[0224] For the temporal candidate, the same position as VTM / HEVC collocated
25 position is used.

[0225] 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.

[0226] The temporal candidate with the same position as VTM / HEVC collocated position used in TMVP is denoted as Col.

30 [0227] The motion vector of the STMVP candidate in prediction direction X (denoted

as mvLX) 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),

$$\text{mvLX} = (\text{mvLX_F} + \text{mvLX_S} + \text{mvLX_T} + \text{mvLX_Col}) \gg 2$$

- 5 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),

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

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

$$\text{mvLX} = (\text{mvLX_S} \times 3 + \text{mvLX_T} \times 3 + \text{mvLX_Col} \times 2) \gg 3.$$

- 10 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),

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

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

$$\text{mvLX} = (\text{mvLX_T} + \text{mvLX_Col}) \gg 1.$$

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

2.18.3. Merge list size

[0228] 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.

20 2.19. Geometric partitioning mode (GPM)

[0229] 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.

[0230] When this mode is used, a CU is split into two parts by a geometrically located straight line (Fig. 30). 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

30 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 2.19.1.

[0231] 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 2.19.2. 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 2.19.3.

2.19.1. Uni-prediction candidate list construction

[0232] The uni-prediction candidate list is derived directly from the merge candidate list constructed according to the extended merge prediction process in 2.17. 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. 31. 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.

2.19.2. Blending along the geometric partitioning edge

[0233] 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.

[0234] The distance for a position (x, y) to the partition edge are derived as:

$$d(x, y) = (2x + 1 - w) \cos(\varphi_i) + (2y + 1 - h) \sin(\varphi_i) - \rho_j \quad (2-25)$$

$$\rho_j = \rho_{x,j} \cos(\varphi_i) + \rho_{y,j} \sin(\varphi_i) \quad (2-26)$$

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

$$\rho_{y,j} = \begin{cases} \pm(j \times h) \gg 2 & i \% 16 = 8 \text{ or } (i \% 16 \neq 0 \text{ and } h \geq w) \\ 0 & \text{otherwise} \end{cases} \quad (2-28)$$

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 .

[0235] The weights for each part of a geometric partition are derived as following:

$$wIdxL(x, y) = partIdx ? 32 + d(x, y) : 32 - d(x, y) \quad (2-29)$$

$$w_0(x, y) = \frac{Clip3(0, 8, (wIdxL(x, y) + 4) \gg 3)}{8} \quad (2-30)$$

$$w_1(x, y) = 1 - w_0(x, y) \quad (2-31)$$

[0236] The partIdx depends on the angle index i . One example of weigh w_0 is illustrated in Fig. 32.

2.19.3. Motion field storage for geometric partitioning mode

[0237] 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.

[0238] The stored motion vector type for each individual position in the motion filed are determined as:

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

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 .

[0239] 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.

2.20. Multi-hypothesis prediction

[0240] 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.

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

[0241] The weighting factor α is specified according to the following Table 4:

Table 4 – weighting factor for MHP

add_hyp_weight_idx	α
0	1/4
1	-1/8

[0242] For inter AMVP mode, MHP is only applied if non-equal weight in BCW is selected in bi-prediction mode.

[0243] 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.

2.21. Non-adjacent spatial candidate

[0244] 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. 33. 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.

2.22. Template matching (TM)

[0245] 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. 34, 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 in this contribution is changed with two modifications: search step size is determined based on AMVR mode and TM can be cascaded with bilateral matching process in merge modes.

[0246] 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 5. This search process ensures that the MVP candidate still keeps the same MV

precision as indicated by AMVR mode after TM process.

Table 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	

[0247] In merge mode, similar search method is applied to the merge candidate indicated by the merge index. As Table 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.

[0248] In the template matching merge mode, the encoder can choose among uni-prediction from list0, uni-prediction from list1 or bi-prediction for a CU. The selection is based on a template matching cost as follows:

If $\text{costBi} \leq \text{factor} * \min(\text{cost0}, \text{cost1})$

bi-prediction is used;

Otherwise, if $\text{cost0} \leq \text{cost1}$

uni-prediction from list0 is used;

Otherwise,

uni-prediction from list1 is used;

where cost0 is the SAD of list0 template matching, cost1 is the SAD of list1 template matching and costBi is the SAD of bi-prediction template matching. The value of factor is equal to 1.125, which means that the selection process is biased toward bi-prediction.

2.23. Overlapped block motion compensation (OBMC)

[0249] 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. 35.

[0250] 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.

[0251] 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.

[0252] 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.

2.24. Multiple transform selection (MTS) for core transform

[0253] 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 6 shows the basis functions of the selected DST/DCT.

Table 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	$T_i(j) = \sqrt{\frac{4}{2N + 1}} \cdot \sin\left(\frac{\pi \cdot (2i + 1) \cdot (j + 1)}{2N + 1}\right)$

[0254] 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.

[0255] 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.

[0256] If MTS CU flag is equal to zero, then DCT2 is applied in both directions.

- 5 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 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
- 10 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
- 15 DST-7 and DCT-8, use 8-bit primary transform cores.

Table 7 – Transform and signalling mapping table

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

[0257] 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.

- 20 [0258] 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.

2.25. Subblock transform (SBT)

[0259] 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.

[0260] 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. 36. 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.

[0261] 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. 36. 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.

[0262] The SBT is not applied to the CU coded with combined inter-intra mode.

2.26. Template matching based adaptive merge candidate reorder

[0263] 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.

[0264] 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. 37. 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. 38.

[0265] The sorting process is operated in the form of sub-group, as illustrated in Fig. 39. The first three merge candidates are sorted together. The following three merge candidates are sorted together.

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

2.27. Adaptive Merge Candidate List

[0266] 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).

[0267] 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. 40.

[0268] 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.

[0269] 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. 41. In Fig. 41, the subgroup the selected (signaled) merge candidate located in is called the selected subgroup.

[0270] 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:

[0271] 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.

- 5 [0272] 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.

[0273] 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.

- 10 [0274] 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.

$$RT = ((8 - w) * RT_0 + w * RT_1 + 4) \gg 3 \quad (2-33)$$

- 15 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.

- [0275] If the Local Illumination Compensation (LIC) flag of the merge candidate is true,
20 the reference samples of the template are derived with LIC method.

[0276] The template matching cost is calculated based on the sum of absolute differences (SAD) of T and RT.

[0277] The template size is 1. That means the width of the left template and/or the height of the above template is 1.

- 25 [0278] If the coding mode is MMVD, the merge candidates to derive the base merge candidates are not reordered.

[0279] If the coding mode is GPM, the merge candidates to derive the uni-prediction candidate list are not reordered.

2.28. Geometric prediction mode with Motion Vector Difference

- 30 [0280] In Geometric prediction mode with Motion Vector Difference (GMVD), each

geometric partition in GPM can decide to use GMVD or not. If GMVD is chosen for a geometric region, the MV of the region is calculated as a sum of the MV of a merge candidate and an MVD. All other processing is kept the same as in GPM.

[0281] With GMVD, an MVD is signaled as a pair of direction and distance. There are
 5 nine candidate distances ($\frac{1}{4}$ -pel, $\frac{1}{2}$ -pel, 1-pel, 2-pel, 3-pel, 4-pel, 6-pel, 8-pel, 16-pel), and eight candidate directions (four horizontal/vertical directions and four diagonal directions) involved. In addition, when `pic_fpel_mmvd_enabled_flag` is equal to 1, the MVD in GMVD is also left shifted by 2 as in MMVD.

2.29. History-parameter-based affine model inheritance and non-adjacent affine mode

[0282] History-parameter-based affine model inheritance (HAMI) allows the affine model to be inherited from a previously affine-coded block which may not be neighboring to the current block. Similar to the enhanced regular merge mode, non-adjacent affine mode (NA-AFF) is introduced.

[0283] A first history-parameter table (HPT) is established. An entry of the first HPT stores a set of affine parameters: a , b , c and d , each of which is represented by a 16-bit signed integer. Entries in HPT is categorized by reference list and reference index. Five reference indices are supported for each reference list in HPT. In a formular way, the category of HPT (denoted as $HPTCat$) is calculated as

$$HPTCat(RefList, RefIdx) = 5 \times RefList + \min(RefIdx, 4),$$

wherein $RefList$ and $RefIdx$ represents a reference picture list (0 or 1) and a reference index, respectively. For each category, at most seven entries can be stored, resulting in 70 entries totally in HPT. At the beginning of each CTU row, the number of entries for each category is initialized as zero. After decoding an affine-coded CU with reference list
 25 $RefList_{cur}$ and $RefIdx_{cur}$, the affine parameters are utilized to update entries in the category $HPTCat(RefList_{cur}, RefIdx_{cur})$ in a way similar to HMVP table updating.

[0284] A history-affine-parameter-based candidate (HAPC) is derived from one of the seven neighbouring 4×4 blocks denoted as $A0$, $A1$, $A2$, $B0$, $B1$, $B2$ or $B3$ in Fig. 41 and a set of affine parameters stored in a corresponding entry in the first HPT. The MV of a
 30 neighbouring 4×4 block served as the base MV. In a formulating way, the MV of the current block at position (x, y) is calculated as:

$$\begin{cases} mv^h(x, y) = a(x - x_{base}) + c(y - y_{base}) + mv_{base}^h, \\ mv^v(x, y) = b(x - x_{base}) + d(y - y_{base}) + mv_{base}^v, \end{cases}$$

where $(mv_{base}^h, mv_{base}^v)$ represents the MV of the neighbouring 4×4 block, (x_{base}, y_{base}) represents the center position of the neighbouring 4×4 block. (x, y) can be the top-left, top-right and bottom-left corner of the current block to obtain the corner-position MVs (CPMVs) for the current block, or it can be the center of the current block to obtain a regular MV for the current block.

[0285] A second history-parameter table (HPT) with base MV information is also appended. There are nine entries in the second HPT, wherein an entry comprises a base MV, a reference index and four affine parameters for each reference list, and a base position. An additional merge HAPC can be generated from the second HPT with the base MV information the corresponding affine models stored in an entry. The difference between the first HPT and the second HPT is illustrated in Fig. 42.

[0286] Moreover, pair-wised affine merge candidates are generated by two affine merge candidates which are history-derived or not history-derived. A pair-wised affine merge candidates is generated by averaging the CPMVs of existing affine merge candidates in the list.

[0287] As a response to new HAPCs being introduced, the size of sub-block-based merge candidate list is increased from five to fifteen, which are all involved in the ARMC process.

[0288] In NA-AFF, the pattern of obtaining non-adjacent spatial neighbors is shown in Fig. 43A and Fig. 43B. Fig. 43A shows deriving inherited candidates and Fig. 43B shows deriving the first type of constructed candidates. Same as the existing non-adjacent regular merge candidates, the distances between non-adjacent spatial neighbors and current coding block in the NA-AFF are also defined based on the width and height of current CU.

[0289] The motion information of the non-adjacent spatial neighbors in Fig. 43A and Fig. 43B is utilized to generate additional inherited and constructed affine merge/AMVP candidates. Specifically, for inherited candidates, the same derivation process of the inherited affine merge/AMVP candidates in the VVC is kept unchanged except that the CPMVs are inherited from non-adjacent spatial neighbors. The non-adjacent spatial

neighbors are checked based on their distances to the current block, i.e., from near to far. At a specific distance, only the first available neighbor (that is coded with the affine mode) from each side (e.g., the left and above) of the current block is included for inherited candidate derivation. As indicated by the red dash arrows in Fig. 43A, the checking orders of the neighbors on the left and above sides are bottom-to-up and right-to-left, respectively.

[0290] For the first type of constructed candidates, as shown in Fig. 43B, the positions of one left and above non-adjacent spatial neighbors are firstly determined independently; After that, the location of the top-left neighbor can be determined accordingly which can enclose a rectangular virtual block together with the left and above non-adjacent neighbors. Then, as shown in the Fig. 44, the motion information of the three non-adjacent neighbors is used to form the CPMVs at the top-left (A), top-right (B) and bottom-left (C) of the virtual block, which is finally projected to the current CU to generate the corresponding constructed candidates.

[0291] The NA-AFF candidates are inserted into the existing affine merge candidate list and affine AMVP candidate list according to the following orders:

Affine merge mode:

1. SbTMVP candidate, if available.
2. Inherited from adjacent neighbors.
3. **Inherited from non-adjacent neighbors.**
4. Constructed from adjacent neighbors.
5. **The first type of constructed affine candidates from non-adjacent neighbors.**
6. Zero MVs.

Affine AMVP mode:

1. Inherited from adjacent neighbors.
2. Constructed from adjacent neighbors.
3. Translational MVs from adjacent neighbors.
4. Translational MVs from temporal neighbors.
5. **Inherited from non-adjacent neighbors.**
6. **The first type of constructed affine candidates from non-adjacent neighbors.**
7. Zero MVs.

[0292] Due to the inclusion of the additional candidates generated by NA-AFF, the size of the affine merge candidate list is increased from 5 to 15. The subgroup size of ARMC for the affine merge mode is increased from 3 to 15.

[0293] Fig. 43A and Fig. 43B illustrate spatial neighbors for deriving affine merge/AMVP candidates: Fig. 43A for deriving inherited candidates and Fig. 43B for deriving the first type of constructed candidates.

In NA-AFF:

1. The area from where the non-adjacent neighbors come is restricted to be within the current CTU (i.e., no additional storage requirements for line buffer).
2. The storage granularity for affine motion information, including CPMVs and reference indexes, is reduced from 8x8 to 16x16 (i.e., only the affine motion from the top-left 8x8 block is saved). Additionally, the saved CPMVs are projected to each 16x16 block before storage, such that the position and size information are not needed.
3. Only the top-left and top-right CPMVs are stored (i.e., always using 4-parameter affine model for NA-AFF).

2.30. Affine MMVD

[0294] In affine MMVD, an affine merge candidate (which is called, base affine merge candidate) is selected, the MVs of the control points are further refined by the signalled MVD information.

[0295] The MVD information for the MVs of all the control points are the same in one prediction direction.

[0296] 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), the MV offset added to the list0 MV component of starting MV and the MV offset for the list1 MV has opposite value; otherwise, when the starting MVs is 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 MV offset added to the list0 MV component of starting MV and the MV offset for the list1 MV are the same.

2.31. Adaptive decoder side motion vector refinement (ADMVR)

[0297] In ECM-2.0, a multi-pass decoder-side motion vector refinement (DMVR) method is applied in regular merge mode if the selected merge candidate meets the DMVR conditions. In the first pass, bilateral matching (BM) is applied to the 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).

[0298] Adaptive decoder side motion vector refinement method consists of the two new merge modes introduced to refine MV only in one direction, either L0 or L1, of the bi prediction for the merge candidates that meet the DMVR conditions. The multi-pass DMVR process is applied for the selected merge candidate to refine the motion vectors, however either MVD0 or MVD1 is set to zero in the 1st pass (i.e., PU level) DMVR.

[0299] Like the regular merge mode, merge candidates for the proposed merge modes are derived from the spatial neighboring coded blocks, TMVPs, non-adjacent blocks, HMVPs, and pair-wise candidate. The difference is that only those meet DMVR conditions are added into the candidate list. The same merge candidate list (i.e., ADMVR merge list) is used by the two proposed merge modes and merge index is coded as in regular merge mode.

3. Problems

[0300] In current design of template matching, the factor (i.e., 1.125) for the determination of whether uni-prediction or bi-prediction is used after motion vector refinement is constant, which may limit the coding performance.

[0301] A refined motion vector with the lower cost (e.g., $MV'0$) is used to further refine another refined motion vector with the larger cost (e.g., $MV'1$) to get a further refined motion vector ($MV''1$). The refinement could be performed in an iterative way.

4. Detailed description

[0302] The detailed solutions below should be considered as examples to explain general concepts. These solutions should not be interpreted in a narrow way. Furthermore, these solutions can be combined in any manner.

Iterative refinement for TM

1. It is proposed that a first motion information (MI_A) is used to refine a second MI (MI_B) during the motion refinement process.
 - a. In one example, the motion refinement process may refer to bilateral matching.
 - b. In one example, the motion refinement process may refer to template matching.
 - c. In one example, MI_A or MI_B may be pre-defined.
 - d. In one example, MI_A or MI_B may be in the same reference list.
 - i. Alternatively, MI_A or MI_B may be in the different reference lists.
 - e. In one example, MI_A or MI_B may be in the same direction, such as the picture order count (POC) of both MI_A and MI_B is less than or larger than that of the current picture.
 - i. Alternatively, MI_A or MI_B may be in the different directions, such as POC of MI_A is less than that of the current picture while POC of MI_B is larger than the current picture.
 - ii. In another example, POC of MI_A is larger than that of the current picture while POC of MI_B is less than the current picture.

- f. In one example, MI_A may be used to determine the refinement process of MI_B such as:
- i. The search range
 - ii. The starting searching point
 - iii. The pattern shape
- g. In one example, MI_A may be refined before being used to refine MI_B .
- h. In one example, MI_A or MI_B may be refined more than once, and denote the i -th refined MI of MI_A and MI_B as $MI_A^{(ri)}$ and $MI_B^{(ri)}$.
- i. In one example, iterative refinement may be used.
- i. In one example, the i -th refined MI of MI_A ($MI_A^{(ri)}$) may be used to obtain the j -th refined MI of MI_B ($MI_B^{(rj)}$).
 - 1) In one example, i may be less than, or equal to, or larger than j .
 - a) In one example, $i = 1$ and $j = 1$, or $i = 2$ and $j = 1$, or $i = 1$ and $j = 2$, or $i = 2$ and $j = 2$, or $i = 0$ and $j = 0$, $i = 1$ and $j = 0$, or $i = 0$ and $j = 1$.
 - 2) In one example, the j -th refined MI of MI_B ($MI_B^{(rj)}$) may be used to obtain the $(i+1)$ -th refined MI of MI_A ($MI_A^{(ri+1)}$) after the i -th refined MI of MI_A ($MI_A^{(ri)}$) is used to obtain the j -th refined MI of MI_B ($MI_B^{(rj)}$).
 - ii. Alternatively, the j -th refined MI of MI_B ($MI_B^{(rj)}$) may be used to obtain the i -th refined MI of MI_A ($MI_A^{(ri)}$).
 - iii. In one example, the template size/shape during the iterative refinement may be different from that without iterative refinement.
 - iv. In one example, the search range and/or the pattern shape (e.g., diamond, square, cross) used for search may be different from that without iterative refinement.
 - v. In one example, the iterative refinement may be applied to a specific coding tool.
 - 1) In one example, the coding tool may refer to adaptive reordering of merge candidates (ARMC), TM merge mode, AMVP-merge mode, or other coding tools in which bilateral matching and/or template matching is used to refine the motion information.
 - 2) In one example, how to use the iterative refinement may be different for different coding tools.
- j. In one example, the determination of whether to and/or how to use the refinement or iterative refinement may depend on coding information.
- i. In one example, the coding information refer to bilateral matching cost, and/or template matching cost, and/or block size/dimensions, and/or template size/dimensions, and/or quantization parameter (QP), and/or POC values of current picture and/or the reference pictures.
 - ii. In one example, the iterative refinement may be used when the template matching cost of bi-prediction ($cost_{Bi}$) is larger than or equal to $S1 * cost_{Uni}$, wherein $cost_{Uni}$ is equal to the template matching cost of a uni-prediction ($cost_0$ or $cost_1$, or $\min(cost_0, cost_1)$, or $\max(cost_0, cost_1)$).
 - iii. In one example, when the POC difference between current picture and its reference pictures is less than or equal to T_P , the refinement may be used.
 - 1) In one example, $T_P = 2$, or 4, or 6, or 8, or 16.

- iv. In one example, when the refinement is used for bi-prediction, the refinement may be used when the POC value of the first reference picture is less than the POC value of current picture and the POC value of the second reference picture is larger than the POC value of current picture.
- 5 v. In one example, when the bilateral matching cost and/or template matching cost is less than or equal to T_C , the refinement may be used.
 - 1) In one example, T_C may depend on block size/dimensions.
- vi. Alternatively, the iterative refinement may be always used.
- 2. It is proposed that the determination of search range and/or pattern shape used for search
 - 10 in template matching may depend on coding information.
 - a. In one example, the coding information may refer to the precision of motion vector (MV) or MV difference (MVD), or a syntax element indicating MV or MVD.
 - b. In one example, the search range for a first MV precision may be larger than that for a second MV precision.
 - 15 i. In one example, the first MV precision is integer and the second MV precision is fractional.
 - ii. In one example, the first MV precision is $1/M$ and the second MV precision is $1/N$, wherein M is less than N .
 - c. In one example, the number of searching points of the pattern shape for a first
 - 20 MV precision may be more than that for a second MV precision.
 - i. In one example, the pattern shapes may be same. An example is shown in Fig. 45.
 - ii. In one example, the patterns shapes may be different. An example is shown in Fig. 46.
 - d. In one example, the coding information may refer to block size/dimensions. Denote the block width and height as W and H respectively.
 - 25 i. In one example, a first search range and/or pattern shape may be used when $W*H \leq T_1$, and a second search range and/or pattern shape may be used when $W*H > T_1$, where T_1 is an integer larger than 0.
 - 30 1) In one example, $T_1 = 64$, or 128, or 256, or 512, or 1024.
 - ii. In one example, the first search pattern shape may refer to 8-point search pattern and the second search pattern shape may refer to 16-point search pattern.
- 3. Instead of a constant factor (S) such as described in section 2.22, it is proposed that at
 - 35 least one adaptive factor may be used for the determination of whether uni-prediction or bi-prediction is used.
 - a. In one example, uni-prediction may be used when the template matching cost of bi-prediction ($cost_{Bi}$) is less than or equal to $S * cost_{Uni}$, wherein $cost_{Uni}$ is equal to the template matching cost of a uni-prediction ($cost_0$ or $cost_1$, or $\min(cost_0, cost_1)$, or $\max(cost_0, cost_1)$).
 - 40
 - b. In one example, the determination may be used for the template matching.
 - c. In one example, the determination may be used for the current video unit.
 - d. In one example, more than one factors may be pre-defined, or signalled, or derived.
 - 45 e. In one example, the determination of using which factor may depend on coding information.

- i. In one example, the coding information may refer to POC or POC difference. Denote the POC of the current picture, the two reference pictures as poc0, poc1, poc2, respectively.
 - 1) In one example, a first factor (S1) may be used when POC difference (d) is less than or equal to T, and a second factor (S2) may be used when POC difference (d) is larger than T, wherein T is an integer larger than 1.
 - a) In one example, $d = \text{abs}(\text{poc0} - \text{poc1}) + \text{abs}(\text{poc0} - \text{poc2})$.
 - b) In one example, $T = 2$, or 3, or 4, or 5, or 6, or 7, or 8, or 10, or 16.
 - c) In one example, S1 is less than S and S2 is equal to or larger than S, such as $S = 1.125$.
 - i. $S1 = 1.115$ and $S2 = 1.125$.
 - d) In one example, S1 is equal to or less than S and S2 is larger than S, such as $S = 1.125$.
 - i. $S1 = 1.125$ and $S2 = 1.135$.
 - ii. In one example, the coding information may refer to whether the two reference pictures are in the same direction.
 - 1) In one example, a first factor (S3) may be used when one reference picture is in the forward and the other reference picture is in the backward, and a second factor (S4) may be used when the two reference pictures are both in the forward or both in the backward, wherein the reference picture in the forward means that its POC is less than the POC of current picture and the reference picture in the backward means that its POC is larger than the POC of current picture and.
 - iii. In one example, the coding information may be:
 - 1) whether a specific coding tool is allowed.
 - 2) block dimensions and/or block size.
 - 3) the depth of a block.
 - 4) slice/picture type and/or partition tree type (single, or dual tree, or local dual tree).
 - 5) block location.
 - 6) colour component.
4. It is proposed that a first motion refinement is used as a part of a second motion refinement.
 - a. In one example, the first motion refinement may refer to template matching, and the second motion refinement may refer to bilateral matching (e.g., DMVR/multi-pass DMVR/Adaptive DMVR).
 - i. In one example, the first motion refinement may refer to template matching for bi-prediction, and the second motion refinement may refer to bilateral matching (e.g., DMVR/multi-pass DMVR/Adaptive DMVR).
 - 1) In one example, the template matching for bi-prediction is used as part of bilateral matching may be used for a specific coding tool, such as ARMC, or TM merge mode.
 - b. In one example, coding information from the first motion refinement may be used in the second motion refinement.

- i. In one example, the coding information may refer to a cost calculated in the first motion refinement.

General aspects

- 5 5. In above examples, the video unit may refer to the 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.
- 10 6. 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.
- 15 7. Whether to and/or how to apply the disclosed methods above may be signalled at PB/TB/CB/PU/TU/CU/VPDU/CTU/CTU row/slice/tile/sub-picture/other kinds of region contains more than one sample or pixel.
8. Whether to and/or how to apply the disclosed methods above may be dependent on coded information, such as block size, colour format, single/dual tree partitioning, colour component, slice/picture type.

[0303] 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. In the following discussion, IntraTMP may be replaced by other coding tools that rely on coded/decoded/reconstructed information within the same region, e.g., palette, intra block copy (IBC).

[0304] Fig. 47 illustrates a flowchart of a method 4700 for video processing in accordance with embodiments of the present disclosure. The method 4700 is implemented during a conversion between a video unit of a video and a bitstream of the video.

[0305] At block 4710, for a conversion between a video unit of a video and a bitstream of the video unit, a first motion information and a second motion information of the video unit is obtained. In some embodiments, the first motion information and the second motion information are in the same reference list. In some other embodiments, the first motion information and the second motion information are in different reference lists.

[0306] At block 4720, the second motion information is refined by using the first motion information, during a refinement process of the video unit. In some embodiments, the refinement process is bilateral matching. In some other embodiments, the refinement process is template matching. In some embodiments, the refinement process is a

refinement. Alternatively, the refinement process is an iterative refinement.

[0307] At block 4730, the conversion is performed based on the refined first and second motion information. In some embodiments, the conversion may include encoding the video unit into the bitstream. Alternatively, or in addition, the conversion may include
5 decoding the video unit from the bitstream. In this way, it improves coding efficiency and coding performance.

[0308] In some embodiments, the iterative refinement is applied to a coding tool. For example, the coding tool may include at least one of: adaptive reordering of merge candidates (ARMC), template matching (TM) merge mode, advanced motion vector
10 prediction (AMVP)-merge mode, other coding tool in which at least one of: bilateral matching or template matching is used to refine motion information. In some other embodiments, an approach to use the iterative refinement is different for different coding tools.

[0309] In some embodiments, a determination of whether to and/or how to use the
15 refinement or the iterative refinement depends on coding information. For example, the coding information comprises at least one of: bilateral matching cost, template matching cost, block size, block dimensions, template size, template dimensions, quantization parameter (QP), a picture order count (POC) value of current picture, or POC values of reference pictures. In some other embodiments, the iterative refinement is always used.

20 [0310] In some embodiments, the iterative refinement is used, if a template matching cost of bi-prediction (costBi) is larger than or equal to $S1 * \text{costUni}$. In this case, costUni may be equal to a template matching cost of a uni-prediction, and $S1$ may be a parameter. In some embodiments, the costUni is equal to one of: cost0 , cost1 , a minimum value between cost0 and cost1 , or a maximum value between cost0 and cost1 .

25 [0311] In some embodiments, if a POC difference between the current picture and its reference pictures is less than or equal to a first threshold, the refinement is used. For example, the first threshold is one of: 2, 4, 6, 8, or 16. In some other embodiments, if the refinement is used for bi-prediction and a first POC value of a first reference picture is less than a POC value of the current picture and a second POC value of a second reference
30 picture is larger than the POC value of current picture, the refinement is used.

[0312] In some embodiments, if at least one of: a bilateral matching cost or template

matching cost is less than or equal to a second threshold, the refinement is used. In some embodiments, the second threshold depends on block size or block dimensions.

[0313] In some embodiments, a determination of at least one of: search range or pattern shape used for search in template matching depends on coding information. In some
5 embodiments, the coding information comprises block size or block dimensions.

[0314] In some embodiments, at least one of a first search range or a first pattern shape is used, if $W \times H \leq T1$. In addition, at least one of: a second search range or a second pattern shape is used, if $W \times H > T1$. In this case, W represents a block width, H represents a block height, and T1 is an integer larger than 0. In some embodiments, T1 is equal to
10 64, or 128, or 256, or 512, or 1024. In some embodiments, the first search pattern shape is an 8-point search pattern and the second search pattern shape is a 16-point search pattern.

[0315] In some embodiments, a first motion refinement is used as a part of a second motion refinement, the first motion refinement is a template matching for bi-prediction, and the second motion refinement comprises a bilateral matching. In some embodiments,
15 the template matching for bi-prediction being used as part of bilateral matching is used for a coding tool. In some embodiments, the coding tool is an ARMC or TM merge mode.

[0316] In some embodiments, the video unit comprises at least one of: a color component, a prediction block (PB), a transform block (TB), a coding block (CB), a prediction unit (PU), a transform unit (TU), a coding tree block (CTB), a coding unit (CU),
20 a coding tree unit (CTU), a CTU row, groups of CTU, a slice, a tile, a sub-picture, a block, a sub-block of a block, a sub-region within a block, or a region containing more than one sample or pixel.

[0317] In some embodiments, an indication of whether to and/or how to refine the second motion information by using the first motion information during the refinement
25 process of the video unit is indicated at one of the followings: sequence level, group of pictures level, picture level, slice level, or tile group level.

[0318] In some embodiments, an indication of whether to and/or how to refine the second motion information by using the first motion information during the refinement
30 process of the video unit is indicated in one of the following: 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.

[0319] In some embodiments, an indication of whether to and/or how to refine the second motion information by using the first motion information during the refinement process of the video unit in one of the following: a prediction block (PB), a transform block (TB), a coding block (CB), a prediction unit (PU), a transform unit (TU), a coding unit (CU), a virtual pipeline data unit (VPDU), a coding tree unit (CTU), a CTU row, a slice, a tile, a sub-picture, or a region containing more than one sample or pixel.

[0320] In some embodiments, the method 4700 further comprises: determining, based on coded information of the video unit, whether and/or how to refine the second motion information by using the first motion information during the refinement process of the video unit. The coded information may include at least one of: a block size, a colour format, a single and/or dual tree partitioning, a colour component, a slice type, or a picture type.

[0321] 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: obtaining a first motion information and a second motion information of a video unit of the video; refining the second motion information by using the first motion information, during a refinement process of the video unit, wherein the refinement process is a refinement or an iterative refinement; and generating the bitstream based on the refined first and second motion information.

[0322] According to still further embodiments of the present disclosure, a method for storing bitstream of a video is provided. The method comprises: obtaining a first motion information and a second motion information of a video unit of the video; refining the second motion information by using the first motion information, during a refinement process of the video unit, wherein the refinement process is a refinement or an iterative refinement; generating the bitstream based on the refined first and second motion information; and storing the bitstream in a non-transitory computer-readable recording medium.

[0323] 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.

[0324] Clause 1. A method of video processing, wherein comprising: obtaining, for a conversion between a video unit of a video and a bitstream of the video unit, a first motion information and a second motion information of the video unit; refining the second motion information by using the first motion information, during a refinement process of the video unit, wherein the refinement process is a refinement or an iterative refinement; and performing the conversion based on the refined first and second motion information.

[0325] Clause 2. The method of clause 1, wherein the iterative refinement is applied to a coding tool.

[0326] Clause 3. The method of clause 2, wherein the coding tool comprises at least one of: adaptive reordering of merge candidates (ARMC), template matching (TM) merge mode, advanced motion vector prediction (AMVP)-merge mode, other coding tool in which at least one of: bilateral matching or template matching is used to refine motion information.

[0327] Clause 4. The method of clause 2, wherein an approach to use the iterative refinement is different for different coding tools.

[0328] Clause 5. The method of clause 1, wherein a determination of whether to and/or how to use the refinement or the iterative refinement depends on coding information.

[0329] Clause 6. The method of clause 5, wherein the coding information comprises at least one of: bilateral matching cost, template matching cost, block size, block dimensions, template size, template dimensions, quantization parameter (QP), a picture order count (POC) value of current picture, or POC values of reference pictures.

[0330] Clause 7. The method of clause 5, wherein the iterative refinement is used, if a template matching cost of bi-prediction (costBi) is larger than or equal to $S1 * \text{costUni}$, wherein costUni is equal to a template matching cost of a uni-prediction, and S1 is a parameter.

[0331] Clause 8. The method of clause 7, wherein the costUni is equal to one of: cost0, cost1, a minimum value between cost0 and cost 1, or a maximum value between cost0 and cost1.

[0332] Clause 9. The method of clause 5, wherein if a POC difference between the current picture and its reference pictures is less than or equal to a first threshold, the refinement is used.

[0333] Clause 10. The method of clause 9, wherein the first threshold is one of: 2, 4, 6, 8, or 16.

[0334] Clause 11. The method of clause 5, wherein if the refinement is used for bi-prediction and a first POC value of a first reference picture is less than a POC value of the current picture and a second POC value of a second reference picture is larger than the POC value of current picture, the refinement is used.

[0335] Clause 12. The method of clause 5, wherein if at least one of: a bilateral matching cost or template matching cost is less than or equal to a second threshold, the refinement is used.

10 [0336] Clause 13. The method of clause 12, wherein the second threshold depends on block size or block dimensions.

[0337] Clause 14. The method of clause 1, wherein the iterative refinement is used.

[0338] Clause 15. The method of clause 1, wherein a determination of at least one of: search range or pattern shape used for search in template matching depends on coding information.

[0339] Clause 16. The method of clause 15, wherein the coding information comprises block size or block dimensions.

[0340] Clause 17. The method of clause 16, wherein at least one of a first search range or a first pattern shape is used, if $W \cdot H \leq T1$, and at least one of: a second search range or a second pattern shape is used, if $W \cdot H > T1$, and wherein W represents a block width, H represents a block height, and $T1$ is an integer larger than 0.

[0341] Clause 18. The method of clause 17, wherein $T1$ is equal to 64, or 128, or 256, or 512, or 1024.

25 [0342] Clause 19. The method of clause 17, wherein the first search pattern shape is an 8-point search pattern and the second search pattern shape is a 16-point search pattern.

[0343] Clause 20. The method of clause 1, wherein a first motion refinement is used as a part of a second motion refinement, the first motion refinement is a template matching for bi-prediction, and the second motion refinement comprises a bilateral matching.

30 [0344] Clause 21. The method of clause 20, wherein the template matching for bi-prediction being used as part of bilateral matching is used for a coding tool.

[0345] Clause 22. The method of clause 20, wherein the coding tool is an ARMC or TM merge mode.

[0346] Clause 23. The method of any of clauses 1-22, wherein the video unit comprises at least one of: a color component, a prediction block (PB), a transform block (TB), a coding block (CB), a prediction unit (PU), a transform unit (TU), a coding tree block (CTB), a coding unit (CU), a coding tree unit (CTU), a CTU row, groups of CTU, a slice, a tile, a sub-picture, a block, a sub-block of a block, a sub-region within a block, or a region containing more than one sample or pixel.

[0347] Clause 24. The method of any of clauses 1-22, wherein an indication of whether to and/or how to refine the second motion information by using the first motion information during the refinement process of the video unit is indicated at one of the followings: sequence level, group of pictures level, picture level, slice level, or tile group level.

[0348] Clause 25. The method of any of clauses 1-22, wherein an indication of whether to and/or how to refine the second motion information by using the first motion information during the refinement process of the video unit is indicated in one of the following: 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.

[0349] Clause 26. The method of any of clauses 1-22, wherein an indication of whether to and/or how to refine the second motion information by using the first motion information during the refinement process of the video unit in one of the following: a prediction block (PB), a transform block (TB), a coding block (CB), a prediction unit (PU), a transform unit (TU), a coding unit (CU), a virtual pipeline data unit (VPDU), a coding tree unit (CTU), a CTU row, a slice, a tile, a sub-picture, or a region containing more than one sample or pixel.

[0350] Clause 27. The method of any of clauses 1-22, further comprising: determining, based on coded information of the video unit, whether and/or how to refine the second motion information by using the first motion information during the refinement process of the video unit, the coded information including at least one of: a block size, a colour format, a single and/or dual tree partitioning, a colour component, a slice type, or a picture

type.

[0351] Clause 28. The method of any of clauses 1-27, wherein the conversion includes encoding the video unit into the bitstream.

[0352] Clause 29. The method of any of clauses 1-27, wherein the conversion includes
5 decoding the video unit from the bitstream.

[0353] Clause 30. 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 1-29.

10 [0354] Clause 31. A non-transitory computer-readable storage medium storing instructions that cause a processor to perform a method in accordance with any of clauses 1-29.

[0355] Clause 32. 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
15 processing, wherein the method comprises: obtaining a first motion information and a second motion information of a video unit of the video; refining the second motion information by using the first motion information, during a refinement process of the video unit, wherein the refinement process is a refinement or an iterative refinement; and generating the bitstream based on the refined first and second motion information.

20 [0356] Clause 33. A method for storing a bitstream of a video, comprising: obtaining a first motion information and a second motion information of a video unit of the video; refining the second motion information by using the first motion information, during a refinement process of the video unit, wherein the refinement process is a refinement or an iterative refinement; generating the bitstream based on the refined first and second motion
25 information; and storing the bitstream in a non-transitory computer-readable recording medium.

Example Device

[0357] Fig. 48 illustrates a block diagram of a computing device 4800 in which various embodiments of the present disclosure can be implemented. The computing device 4800
30 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).

[0358] It would be appreciated that the computing device 4800 shown in Fig. 48 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.

5 [0359] As shown in Fig. 48, the computing device 4800 includes a general-purpose computing device 4800. The computing device 4800 may at least comprise one or more processors or processing units 4810, a memory 4820, a storage unit 4830, one or more communication units 4840, one or more input devices 4850, and one or more output devices 4860.

10 [0360] In some embodiments, the computing device 4800 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
15 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
20 combination thereof, including the accessories and peripherals of these devices, or any combination thereof. It would be contemplated that the computing device 4800 can support any type of interface to a user (such as “wearable” circuitry and the like).

[0361] The processing unit 4810 may be a physical or virtual processor and can implement various processes based on programs stored in the memory 4820. In a multi-
25 processor system, multiple processing units execute computer executable instructions in parallel so as to improve the parallel processing capability of the computing device 4800. The processing unit 4810 may also be referred to as a central processing unit (CPU), a microprocessor, a controller or a microcontroller.

[0362] The computing device 4800 typically includes various computer storage medium.
30 Such medium can be any medium accessible by the computing device 4800, including, but not limited to, volatile and non-volatile medium, or detachable and non-detachable medium. The memory 4820 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 4830 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 4800.

[0363] The computing device 4800 may further include additional detachable/non-detachable, volatile/non-volatile memory medium. Although not shown in Fig. 48, 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.

[0364] The communication unit 4840 communicates with a further computing device via the communication medium. In addition, the functions of the components in the computing device 4800 can be implemented by a single computing cluster or multiple computing machines that can communicate via communication connections. Therefore, the computing device 4800 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.

[0365] The input device 4850 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 4860 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 4840, the computing device 4800 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 4800, or any devices (such as a network card, a modem and the like) enabling the computing device 4800 to communicate with one or more other computing devices, if required. Such communication can be performed via input/output (I/O) interfaces (not shown).

[0366] In some embodiments, instead of being integrated in a single device, some or all components of the computing device 4800 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
5 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
10 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
15 location. Alternatively, they may be provided from a conventional server or installed directly or otherwise on a client device.

[0367] The computing device 4800 may be used to implement video encoding/decoding in embodiments of the present disclosure. The memory 4820 may include one or more video coding modules 4825 having one or more program instructions. These modules are
20 accessible and executable by the processing unit 4810 to perform the functionalities of the various embodiments described herein.

[0368] In the example embodiments of performing video encoding, the input device 4850 may receive video data as an input 4870 to be encoded. The video data may be processed, for example, by the video coding module 4825, to generate an encoded
25 bitstream. The encoded bitstream may be provided via the output device 4860 as an output 4880.

[0369] In the example embodiments of performing video decoding, the input device 4850 may receive an encoded bitstream as the input 4870. The encoded bitstream may be processed, for example, by the video coding module 4825, to generate decoded video data.
30 The decoded video data may be provided via the output device 4860 as the output 4880.

[0370] 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 of video processing, wherein comprising:
obtaining, for a conversion between a video unit of a video and a bitstream of the
5 video unit, a first motion information and a second motion information of the video unit;
refining the second motion information by using the first motion information, during a
refinement process of the video unit, wherein the refinement process is a refinement or an
iterative refinement; and
performing the conversion based on the refined first and second motion information.

2. The method of claim 1, wherein the iterative refinement is applied to a coding tool.

3. The method of claim 2, wherein the coding tool comprises at least one of: adaptive
reordering of merge candidates (ARMC), template matching (TM) merge mode, advanced
15 motion vector prediction (AMVP)-merge mode, other coding tool in which at least one of:
bilateral matching or template matching is used to refine motion information.

4. The method of claim 2, wherein an approach to use the iterative refinement is
different for different coding tools.

5. The method of claim 1, wherein a determination of whether to and/or how to use the
refinement or the iterative refinement depends on coding information.

6. The method of claim 5, wherein the coding information comprises at least one of:
25 bilateral matching cost, template matching cost, block size, block dimensions, template size,
template dimensions, quantization parameter (QP), a picture order count (POC) value of
current picture, or POC values of reference pictures.

7. The method of claim 5, wherein the iterative refinement is used, if a template
30 matching cost of bi-prediction (costBi) is larger than or equal to $S1 * \text{costUni}$, wherein
costUni is equal to a template matching cost of a uni-prediction, and S1 is a parameter.

8. The method of claim 7, wherein the costUni is equal to one of: cost0, cost1, a minimum value between cost0 and cost 1, or a maximum value between cost0 and cost1.

9. The method of claim 5, wherein if a POC difference between the current picture and its reference pictures is less than or equal to a first threshold, the refinement is used.

10. The method of claim 9, wherein the first threshold is one of: 2, 4, 6, 8, or 16.

11. The method of claim 5, wherein if the refinement is used for bi-prediction and a first POC value of a first reference picture is less than a POC value of the current picture and a second POC value of a second reference picture is larger than the POC value of current picture, the refinement is used.

12. The method of claim 5, wherein if at least one of: a bilateral matching cost or template matching cost is less than or equal to a second threshold, the refinement is used.

13. The method of claim 12, wherein the second threshold depends on block size or block dimensions.

14. The method of claim 1, wherein the iterative refinement is used.

15. The method of claim 1, wherein a determination of at least one of: search range or pattern shape used for search in template matching depends on coding information.

16. The method of claim 15, wherein the coding information comprises block size or block dimensions.

17. The method of claim 16, wherein at least one of a first search range or a first pattern shape is used, if $W \cdot H \leq T1$, and
at least one of: a second search range or a second pattern shape is used, if $W \cdot H > T1$,
and
wherein W represents a block width, H represents a block height, and T1 is an integer larger than 0.

18. The method of claim 17, wherein T1 is equal to 64, or 128, or 256, or 512, or 1024.

19. The method of claim 17, wherein the first search pattern shape is an 8-point search pattern and the second search pattern shape is a 16-point search pattern.

20. The method of claim 1, wherein a first motion refinement is used as a part of a second motion refinement, the first motion refinement is a template matching for bi-prediction, and the second motion refinement comprises a bilateral matching.

21. The method of claim 20, wherein the template matching for bi-prediction being used as part of bilateral matching is used for a coding tool.

22. The method of claim 20, wherein the coding tool is an ARMC or TM merge mode.

23. The method of any of claims 1-22, wherein the video unit comprises at least one of:

a color component,
a prediction block (PB),
a transform block (TB),
a coding block (CB),
a prediction unit (PU),
a transform unit (TU),
a coding tree block (CTB),
a coding unit (CU),
a coding tree unit (CTU),
a CTU row,
groups of CTU,
a slice,
a tile,
a sub-picture,
a block,
a sub-block of a block,
a sub-region within a block, or

a region containing more than one sample or pixel.

24. The method of any of claims 1-22, wherein an indication of whether to and/or how to refine the second motion information by using the first motion information during the

refinement process of the video unit is indicated at one of the followings:

sequence level,

group of pictures level,

picture level,

slice level, or

tile group level.

25. The method of any of claims 1-22, wherein an indication of whether to and/or how to refine the second motion information by using the first motion information during the

refinement process of the video unit is indicated in one of the following:

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.

26. The method of any of claims 1-22, wherein an indication of whether to and/or how to refine the second motion information by using the first motion information during the

refinement process of the video unit in one of the following:

a prediction block (PB),

a transform block (TB),

a coding block (CB),

a prediction unit (PU),

a transform unit (TU),

a coding unit (CU),

a virtual pipeline data unit (VPDU),
a coding tree unit (CTU),
a CTU row,
a slice,
5 a tile,
a sub-picture, or
a region containing more than one sample or pixel.

27. The method of any of claims 1-22, further comprising:
10 determining, based on coded information of the video unit, whether and/or how to
refine the second motion information by using the first motion information during the
refinement process of the video unit, the coded information including at least one of:

a block size,
a colour format,
15 a single and/or dual tree partitioning,
a colour component,
a slice type, or
a picture type.

20 28. The method of any of claims 1-27, wherein the conversion includes encoding the
video unit into the bitstream.

29. The method of any of claims 1-27, wherein the conversion includes decoding the
video unit from the bitstream.

25 30. 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 1-29.

30 31. A non-transitory computer-readable storage medium storing instructions that cause
a processor to perform a method in accordance with any of claims 1-29.

32. 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:

5 obtaining a first motion information and a second motion information of a video unit of the video;

refining the second motion information by using the first motion information, during a refinement process of the video unit, wherein the refinement process is a refinement or an iterative refinement; and

10 generating the bitstream based on the refined first and second motion information.

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

obtaining a first motion information and a second motion information of a video unit of the video;

15 refining the second motion information by using the first motion information, during a refinement process of the video unit, wherein the refinement process is a refinement or an iterative refinement;

generating the bitstream based on the refined first and second motion information; and storing the bitstream in a non-transitory computer-readable recording medium.

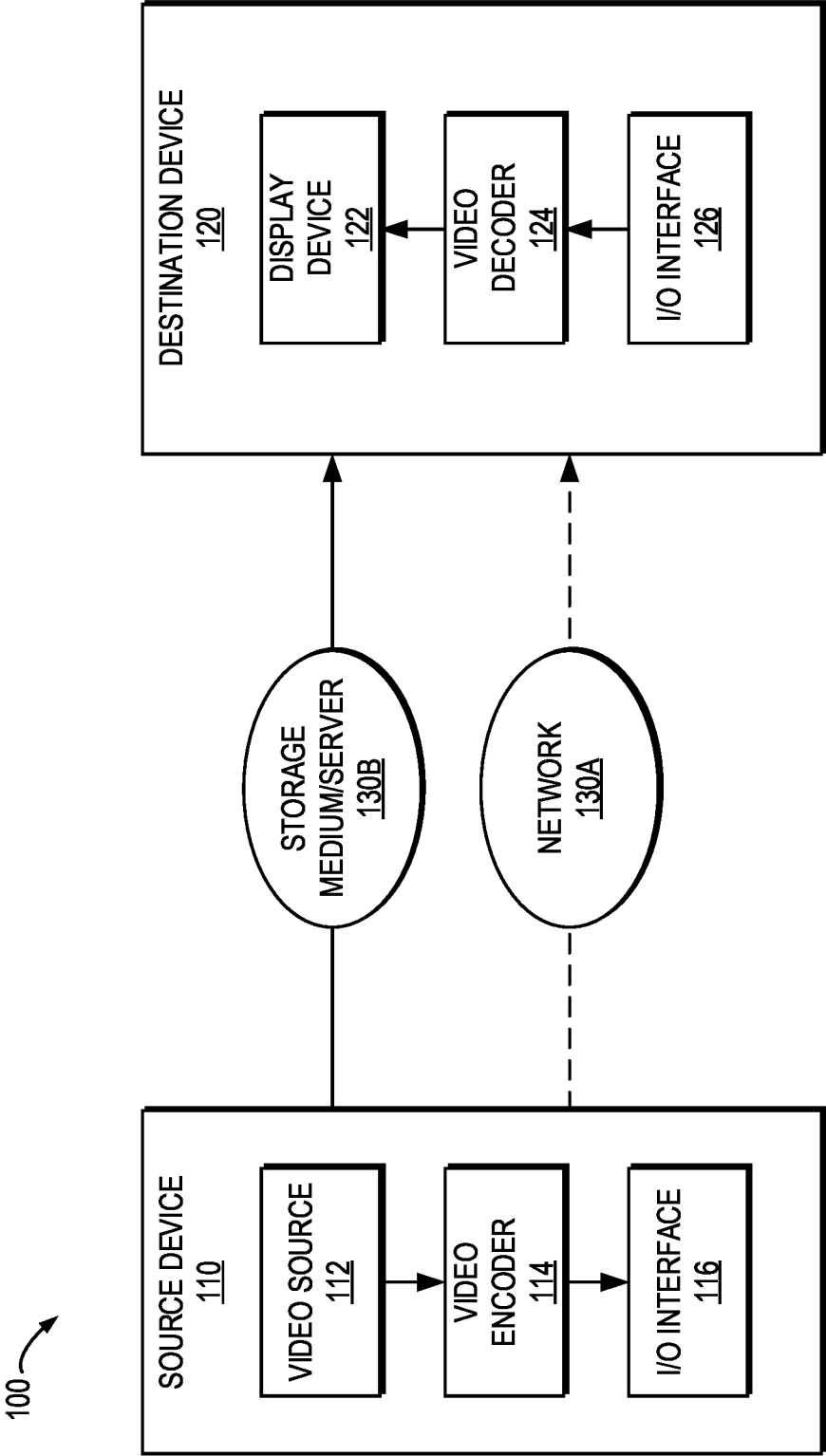


Fig. 1

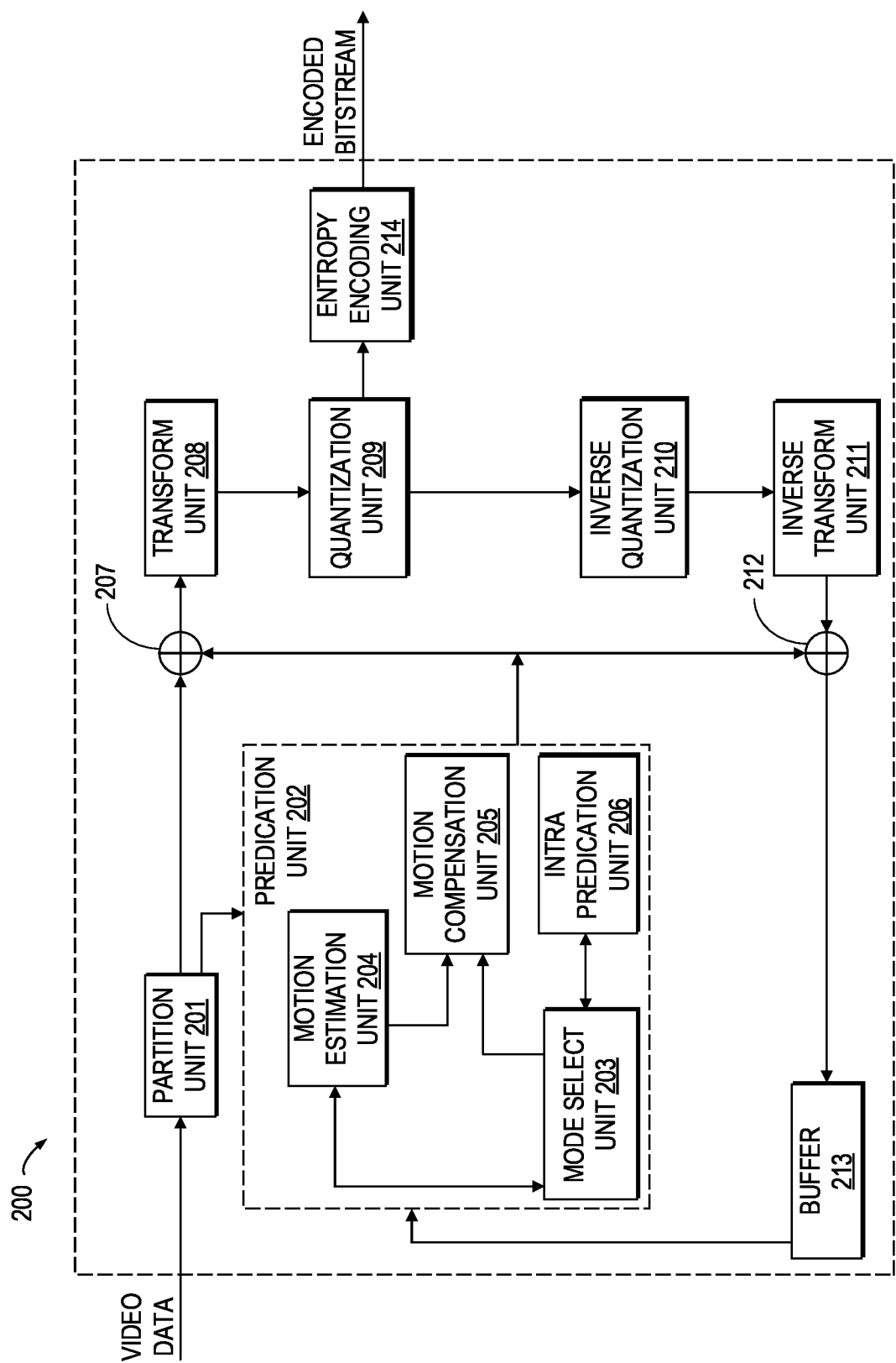


Fig. 2

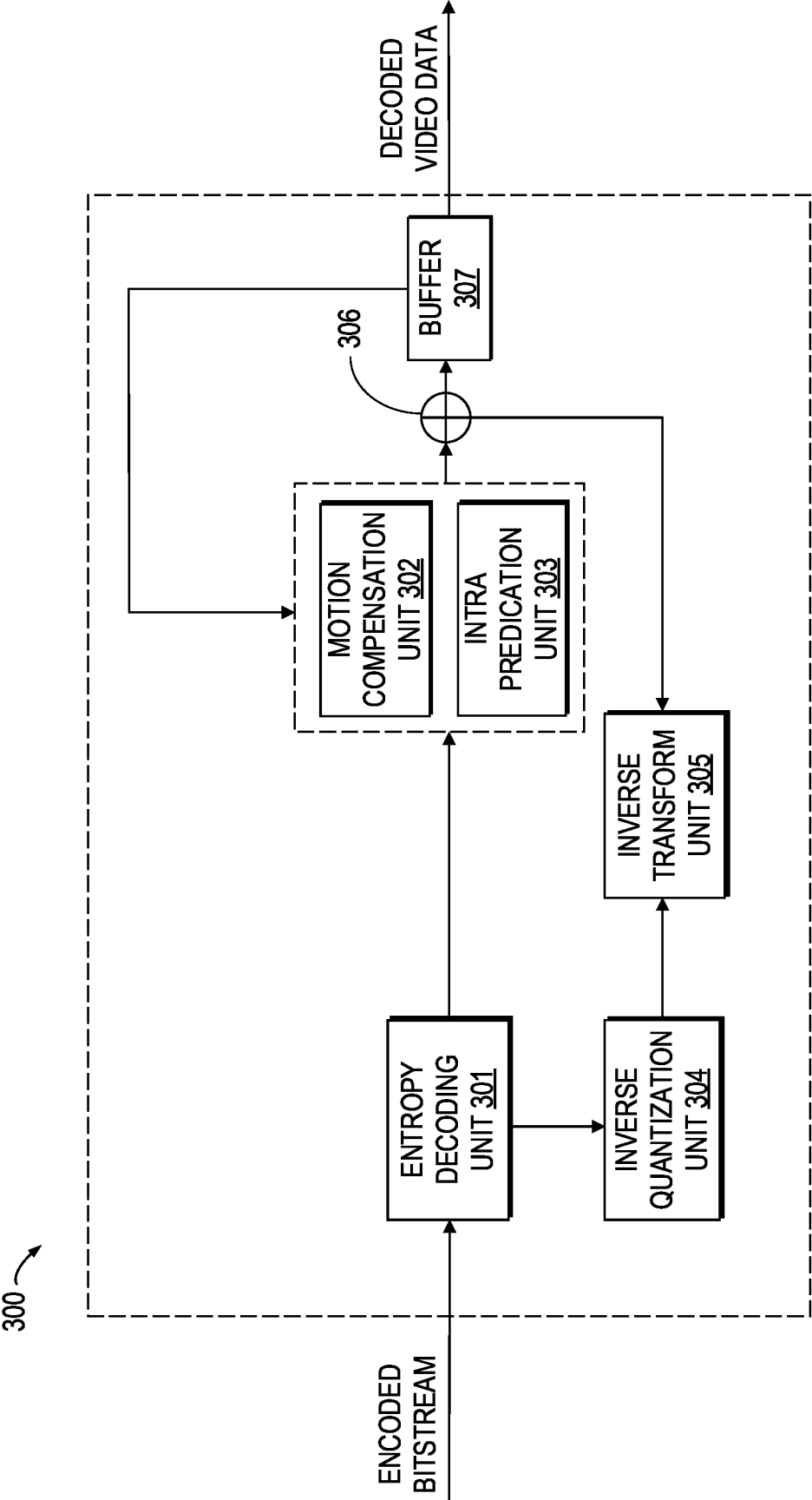


Fig. 3

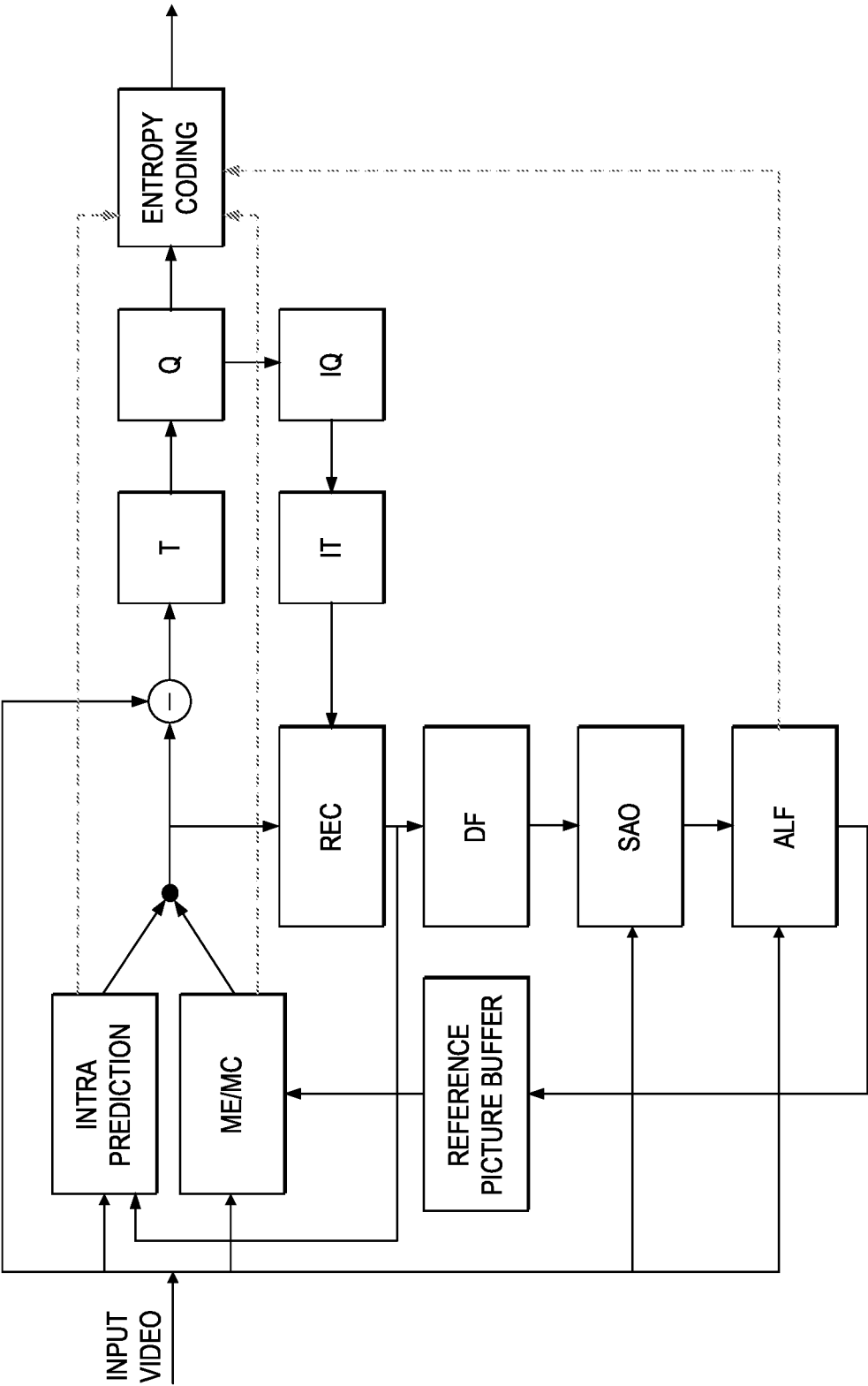


Fig. 4

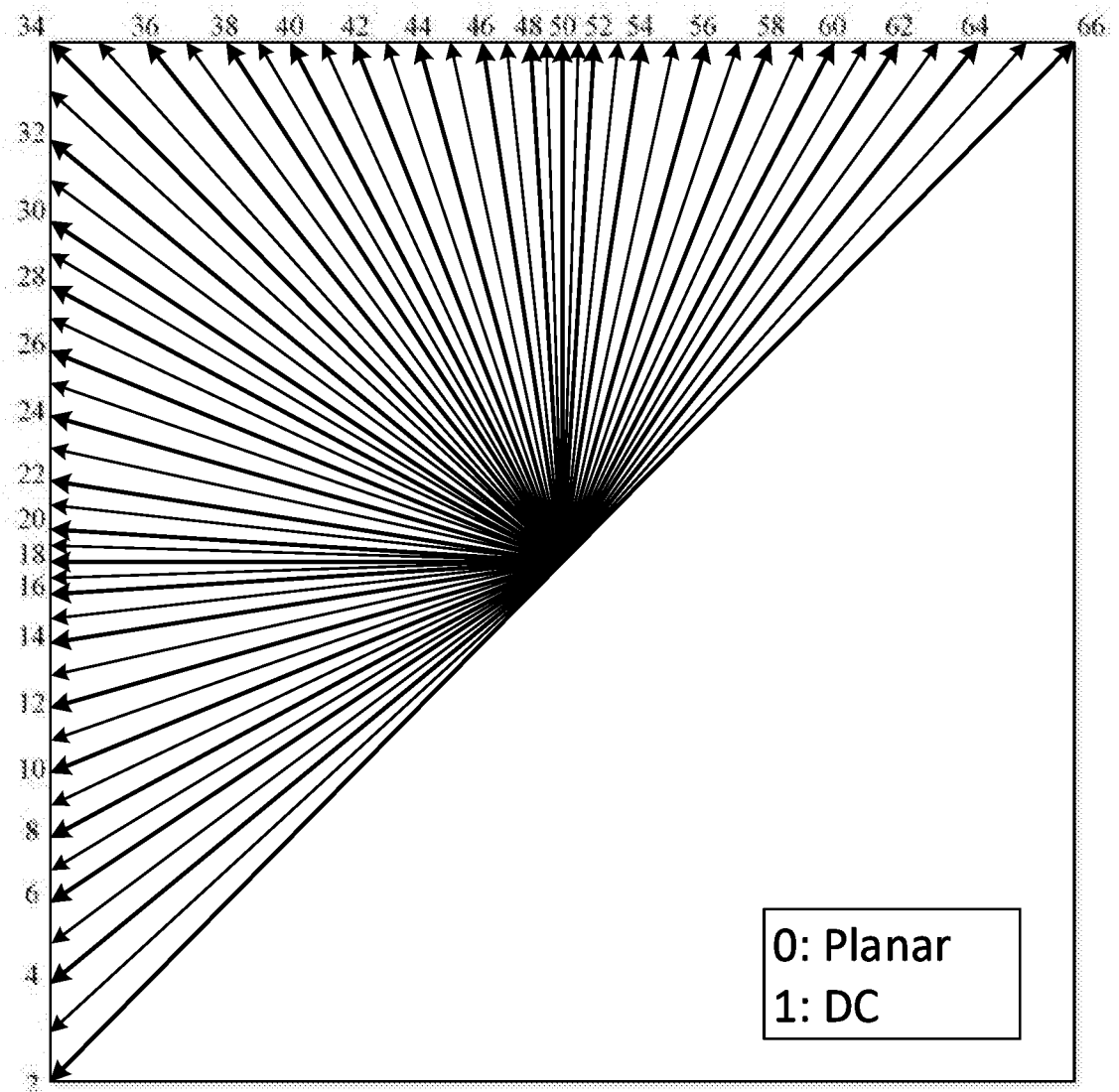


Fig. 5

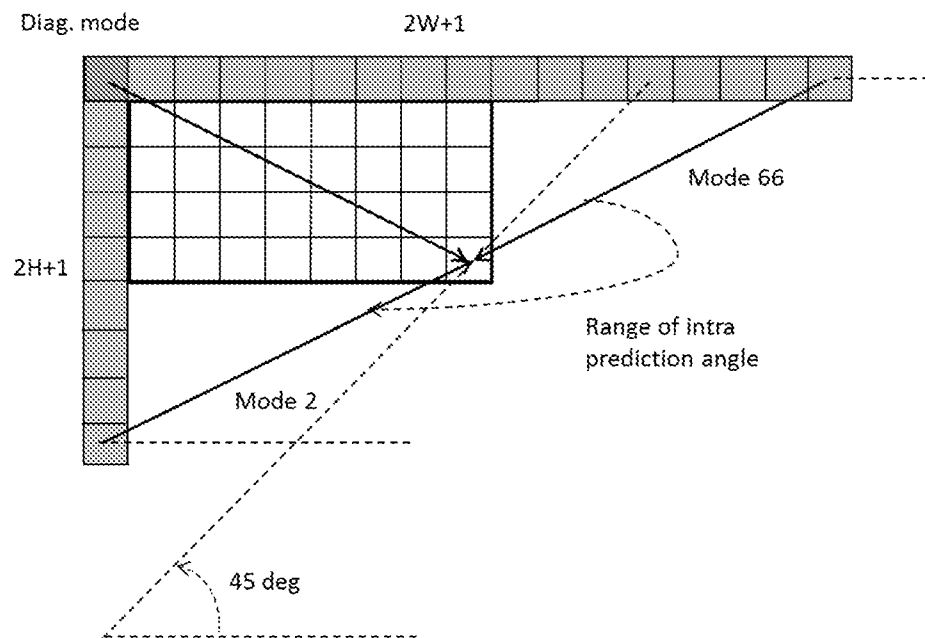


Fig. 6A

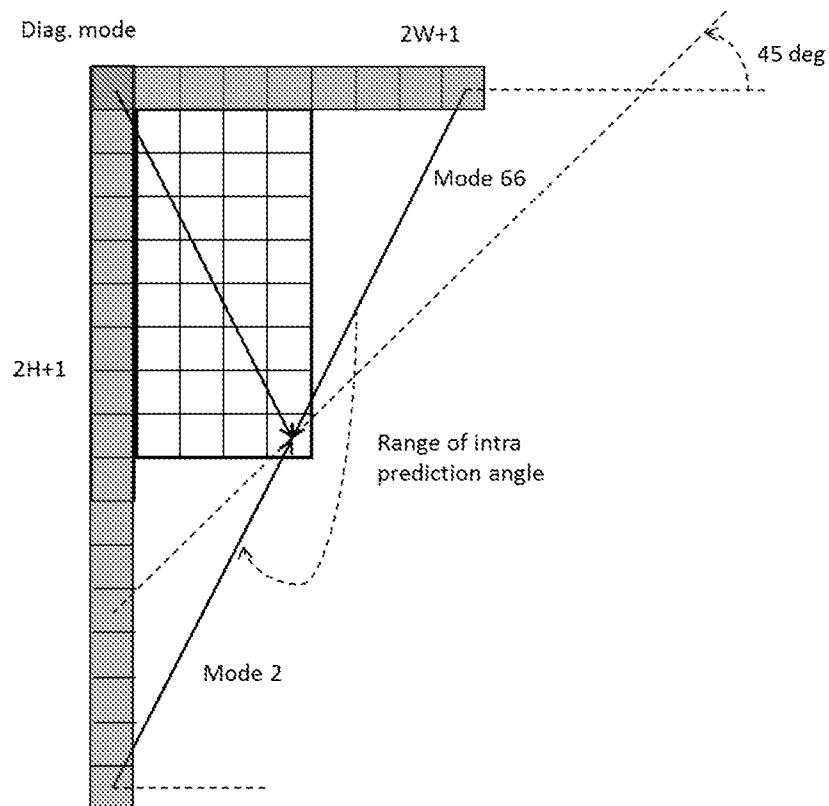


Fig. 6B

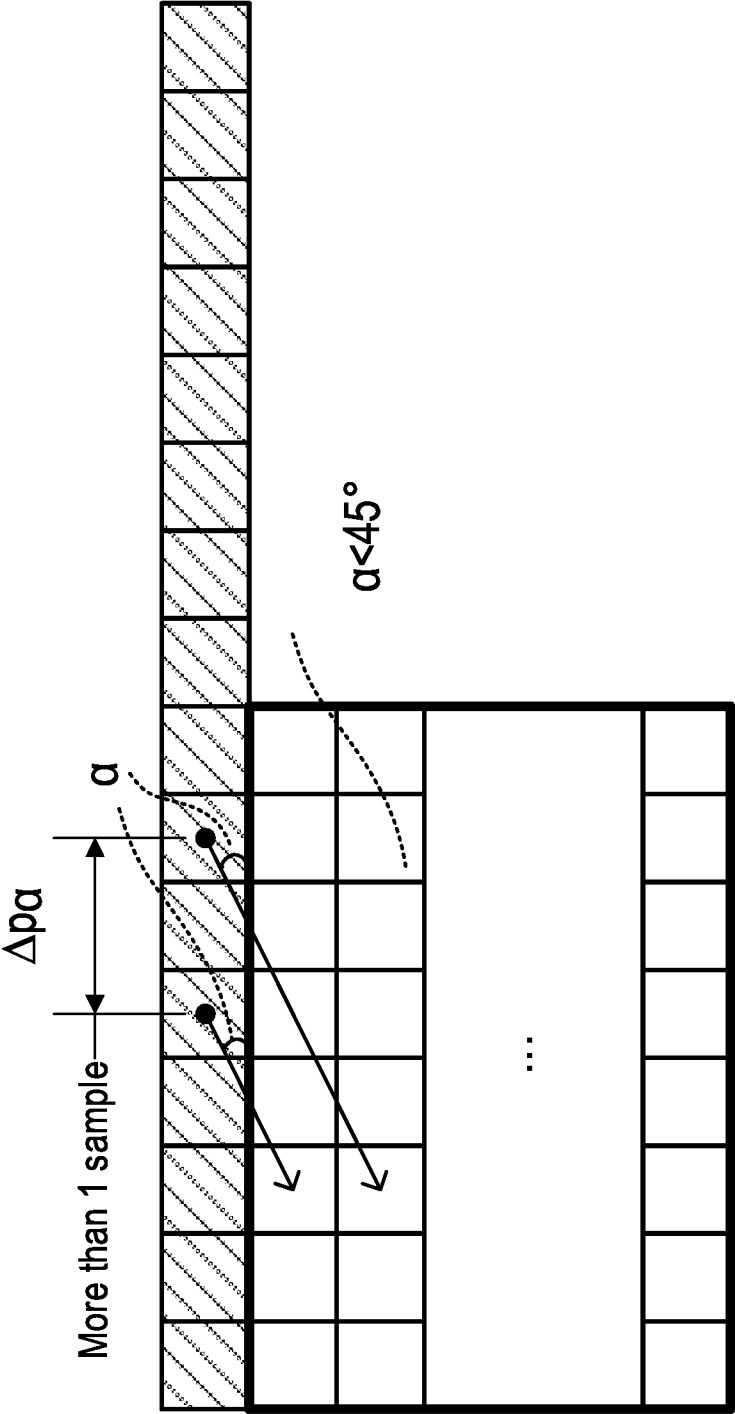


Fig. 7

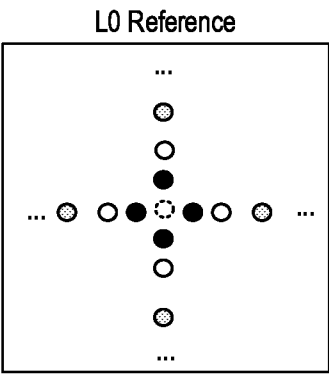


Fig. 8A

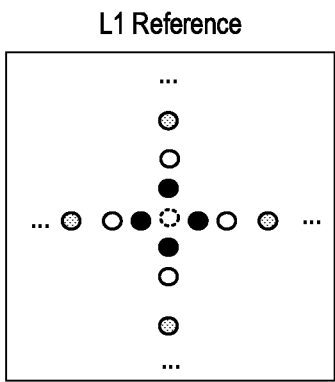


Fig. 8B

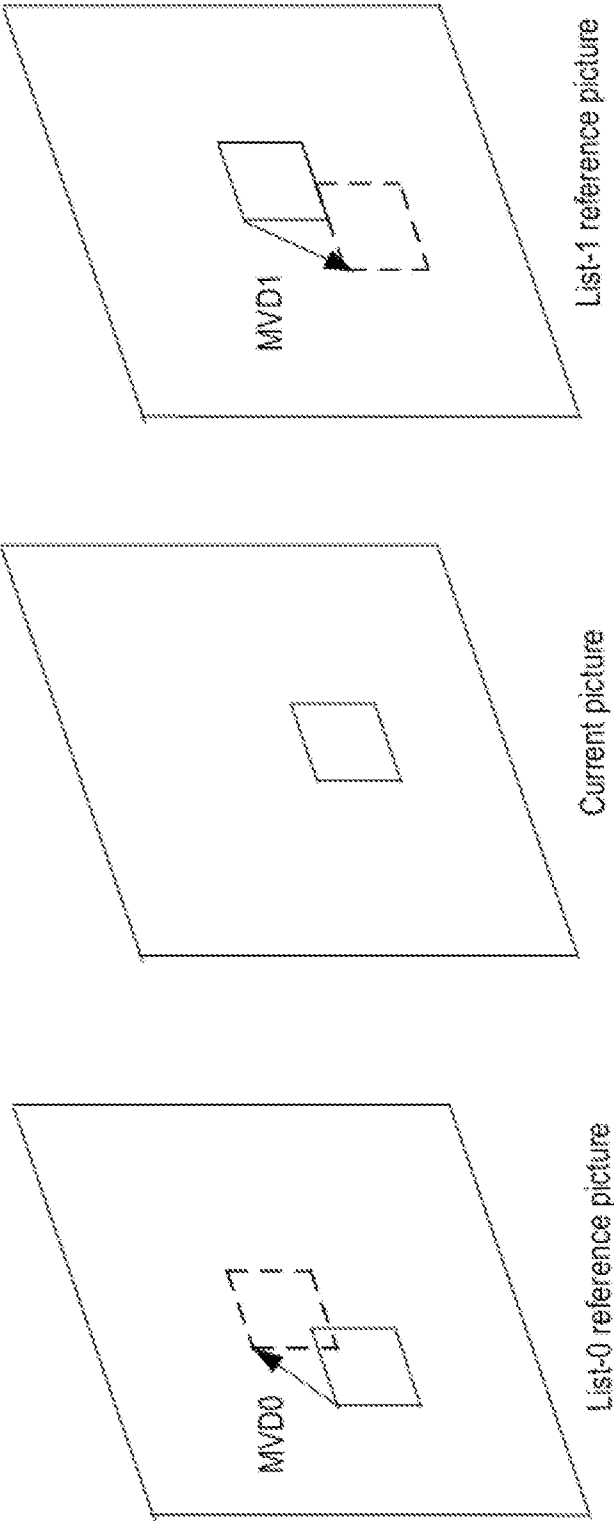


Fig. 9

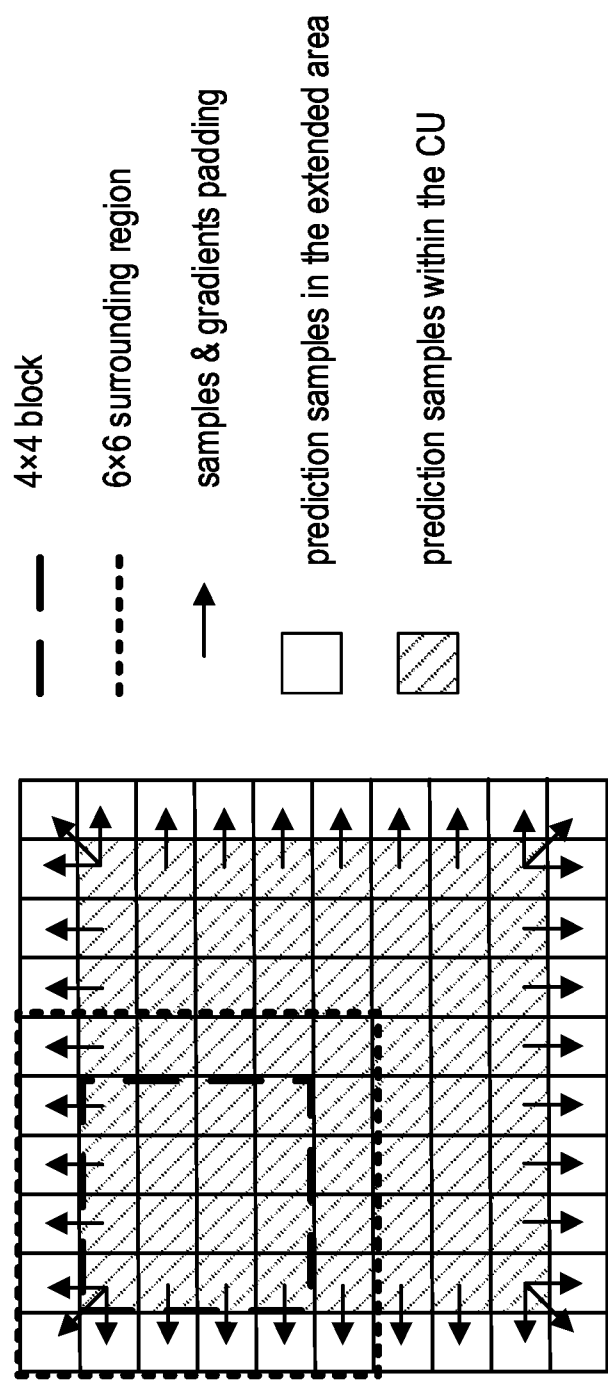


Fig. 10

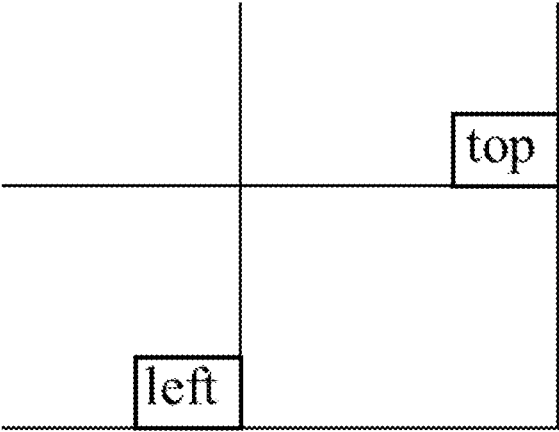


Fig. 11

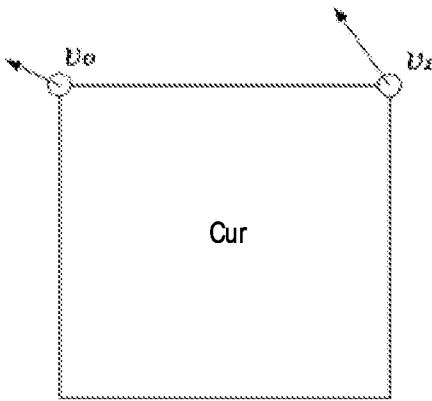


Fig. 12A

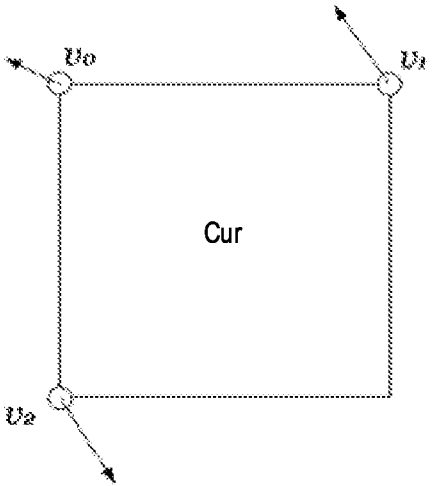


Fig. 12B

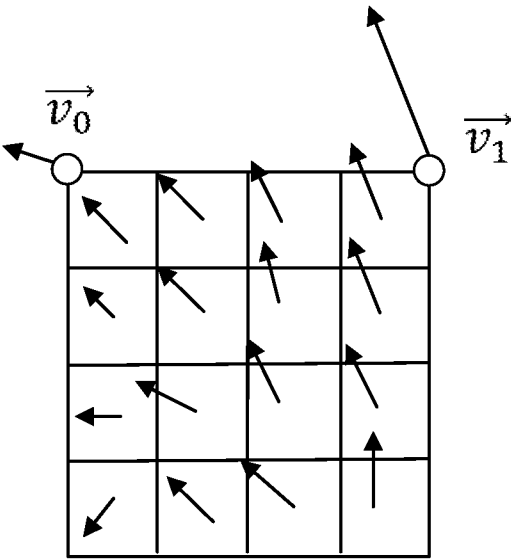


Fig. 13

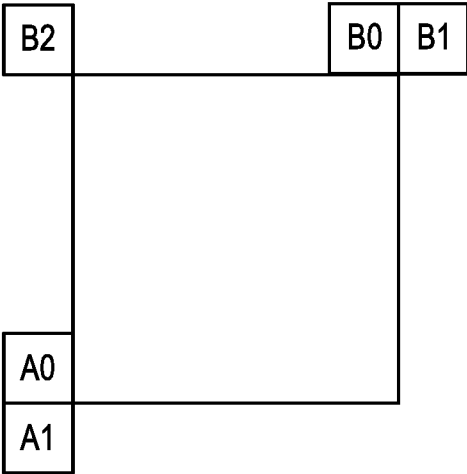


Fig. 14

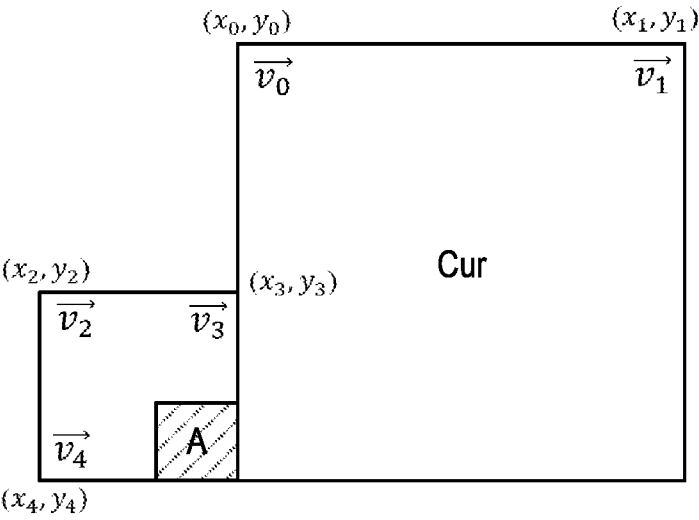


Fig. 15

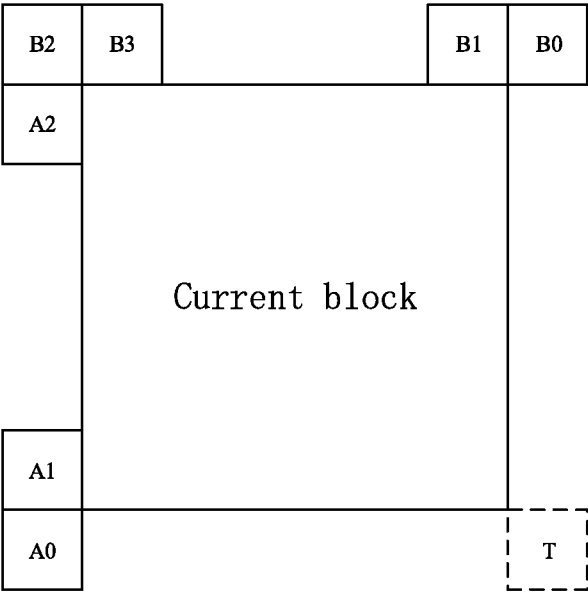


Fig. 16

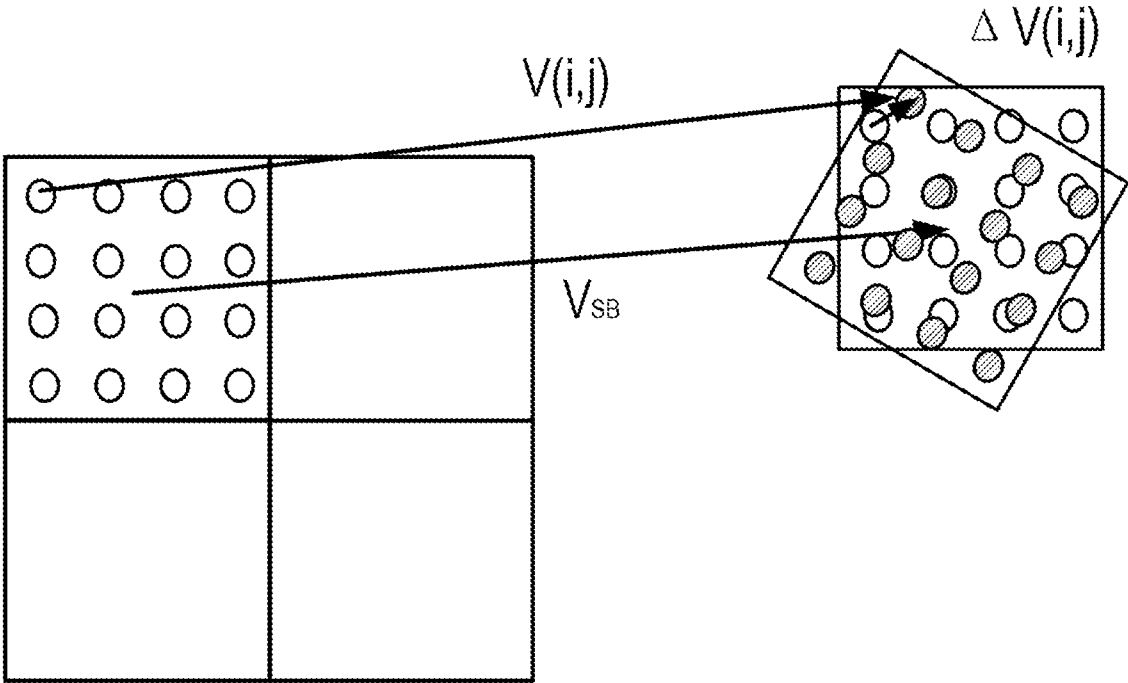


Fig. 18

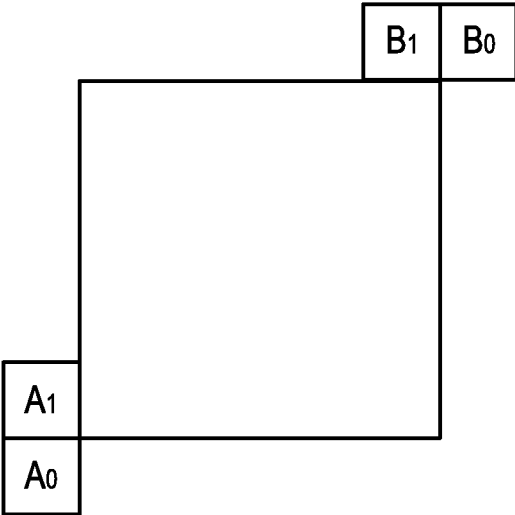


Fig. 19A

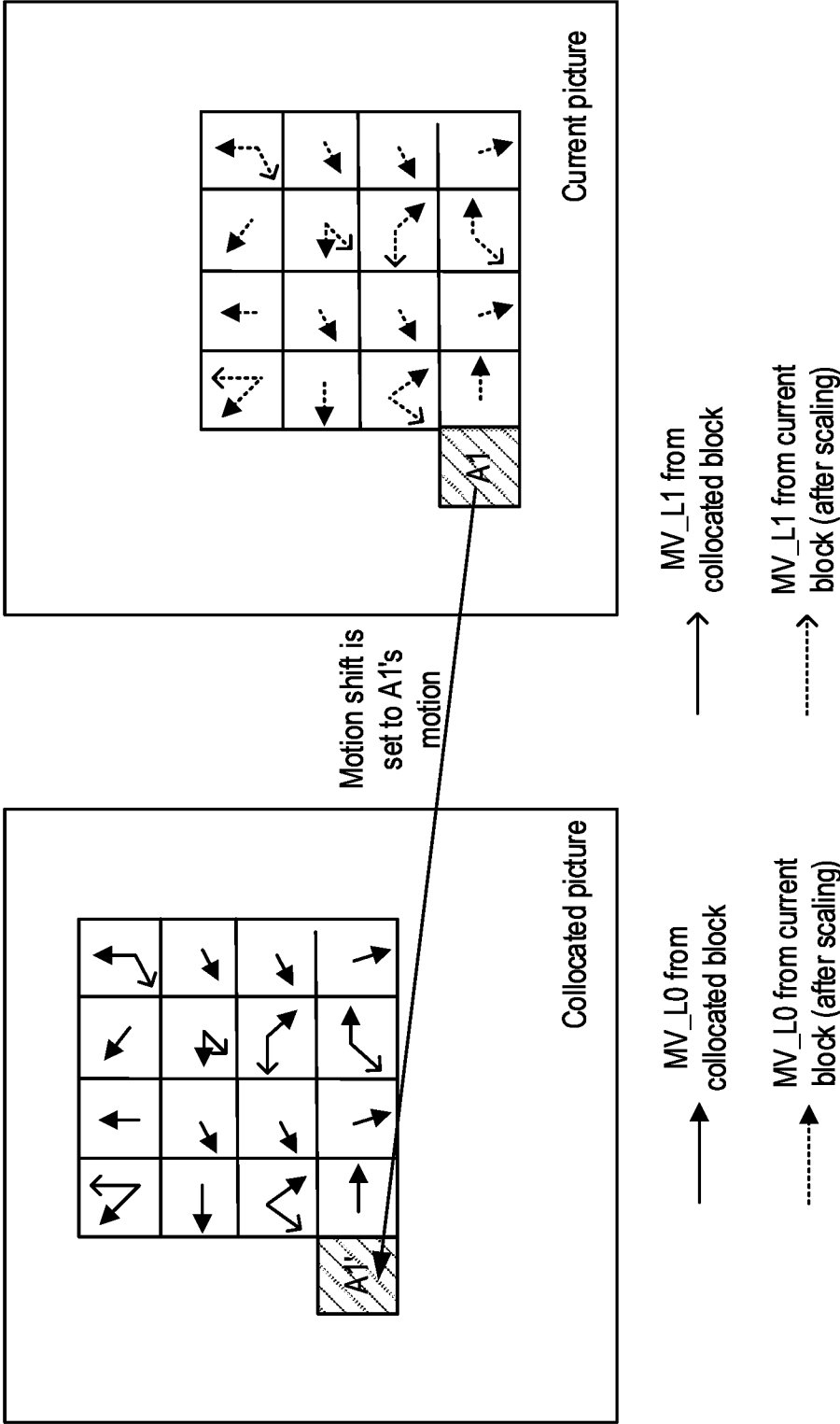


Fig. 19B

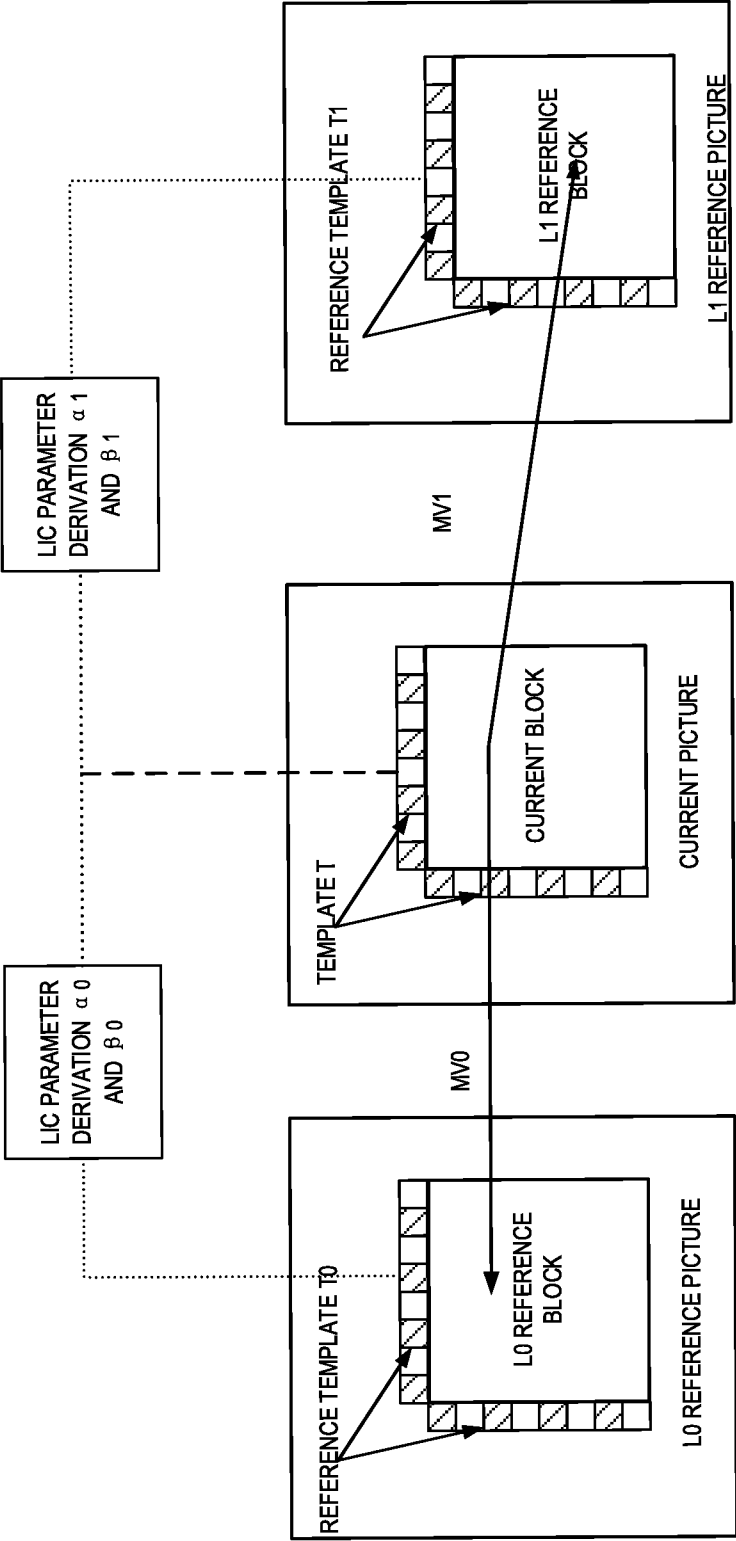


Fig. 20

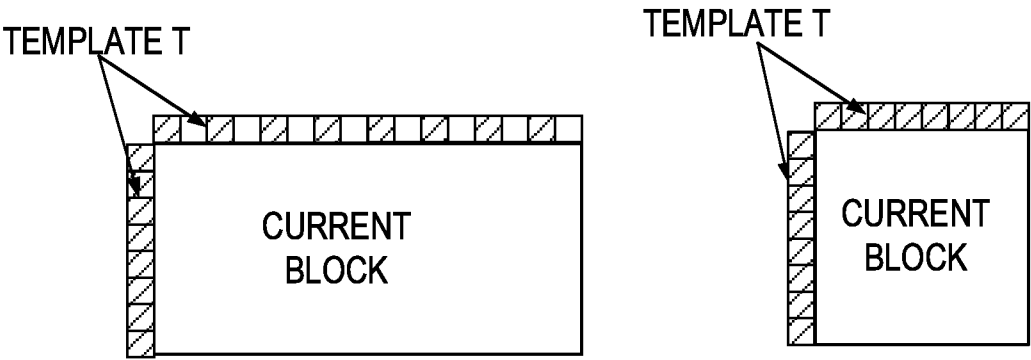


Fig. 21

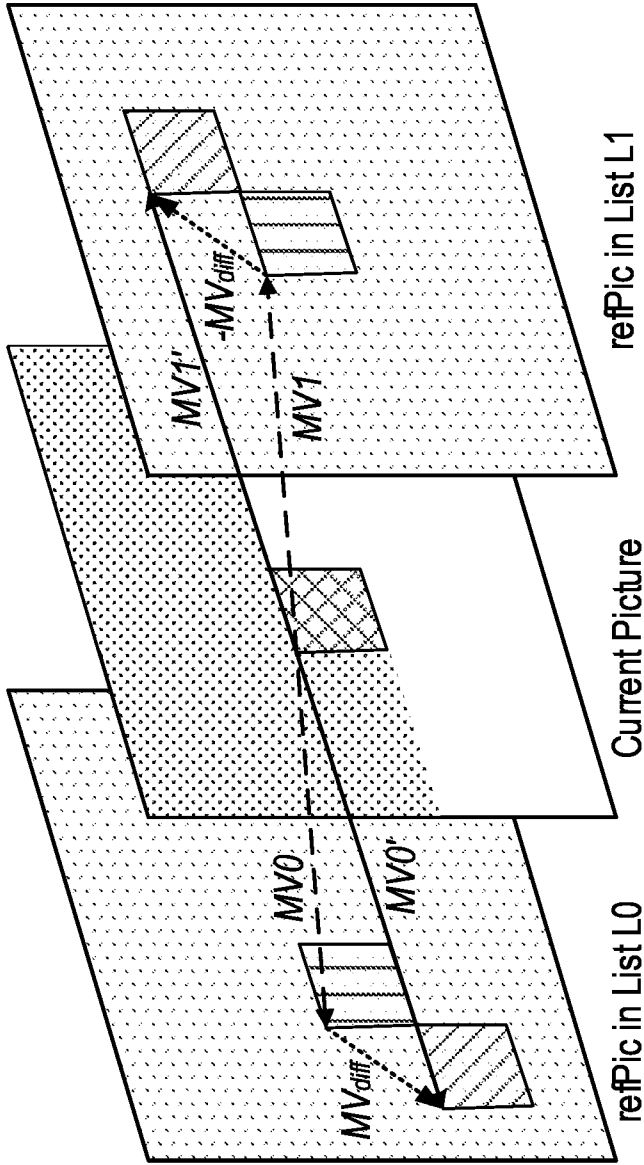


Fig. 22

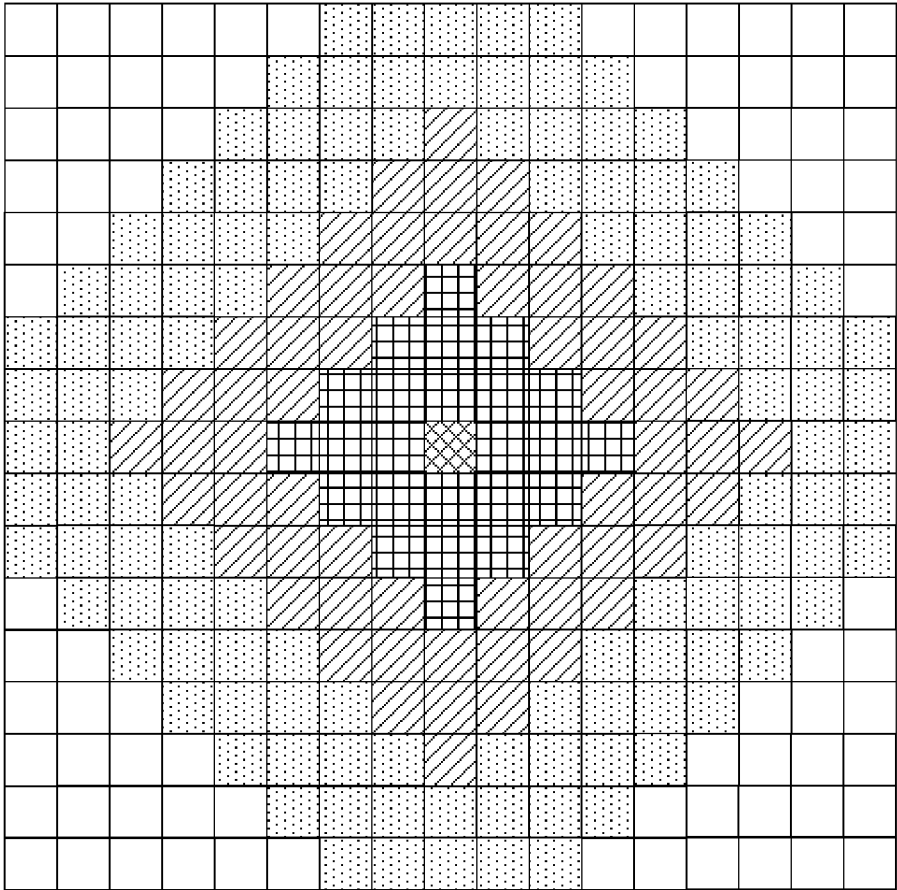


Fig. 23

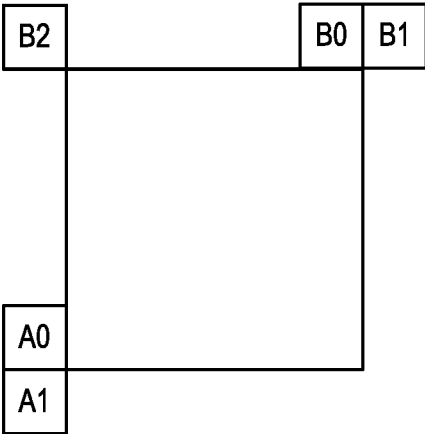


Fig. 24

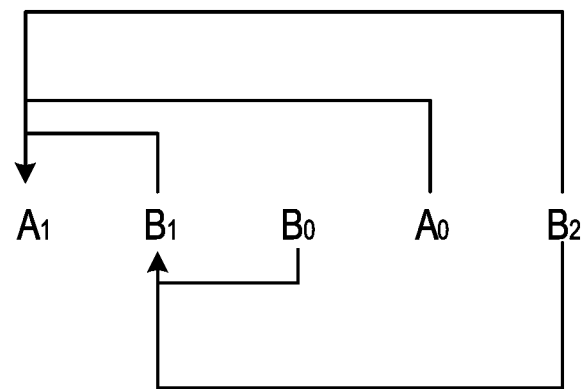


Fig. 25

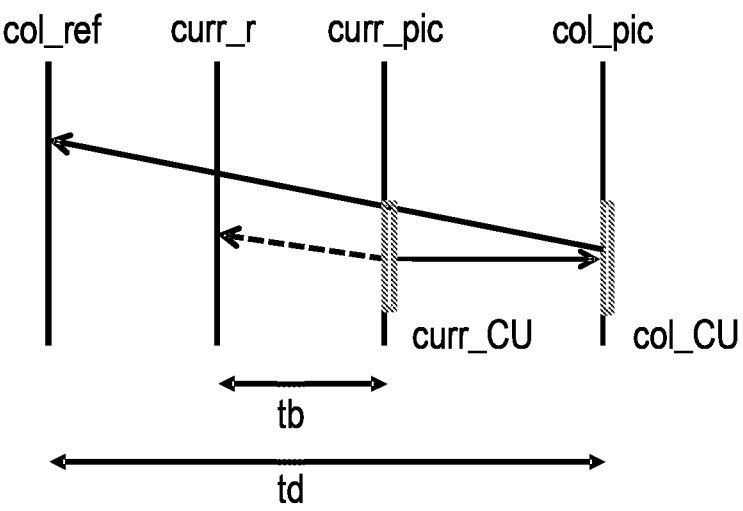


Fig. 26

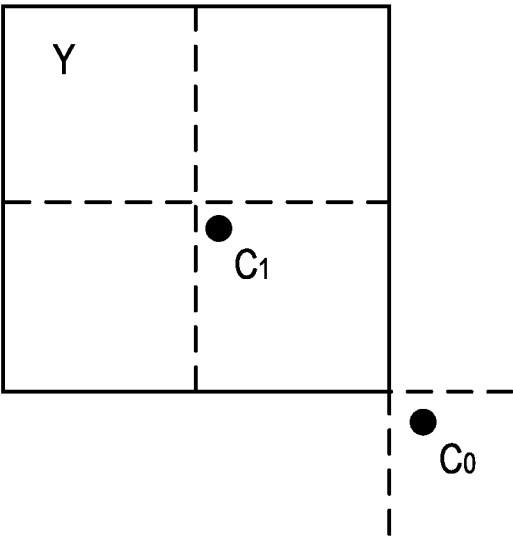


Fig. 27

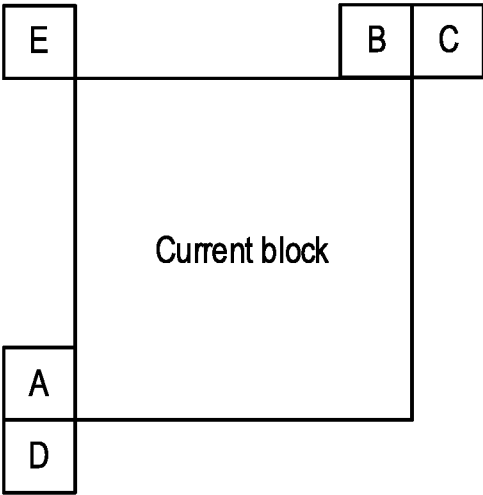


Fig. 28

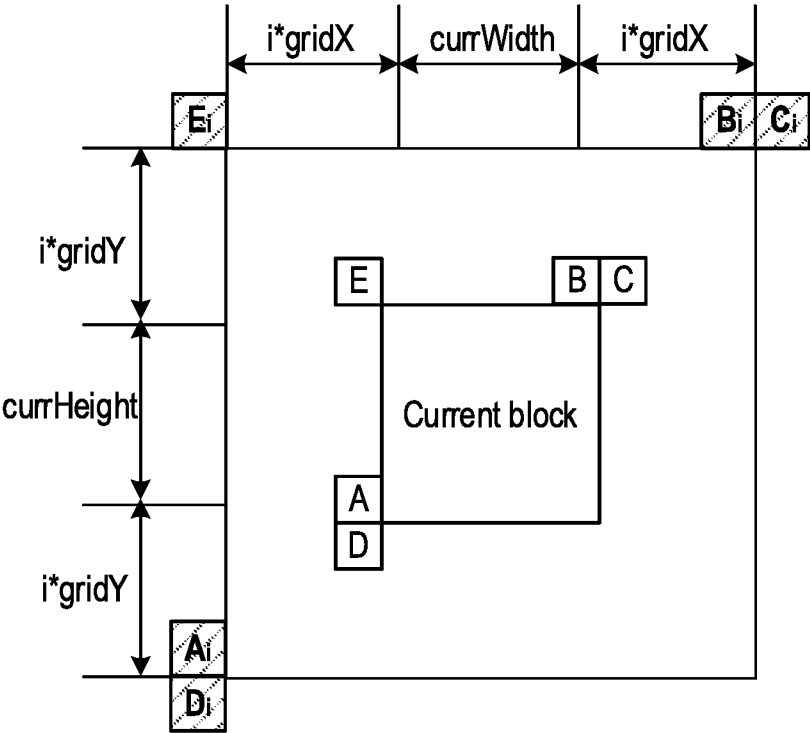


Fig. 29

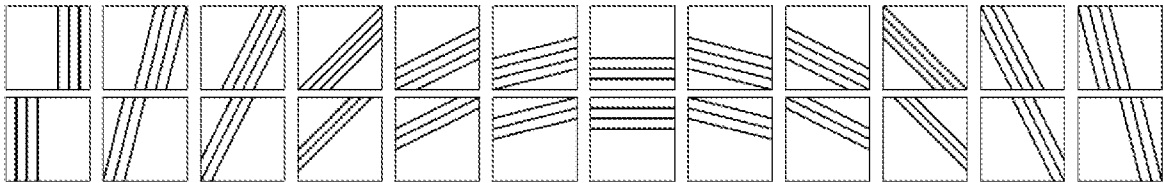


Fig. 30

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

Fig. 31

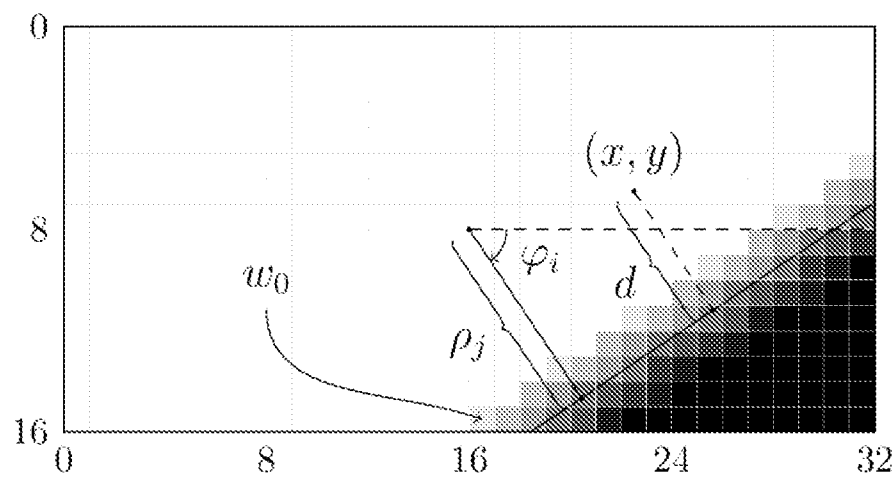


Fig. 32

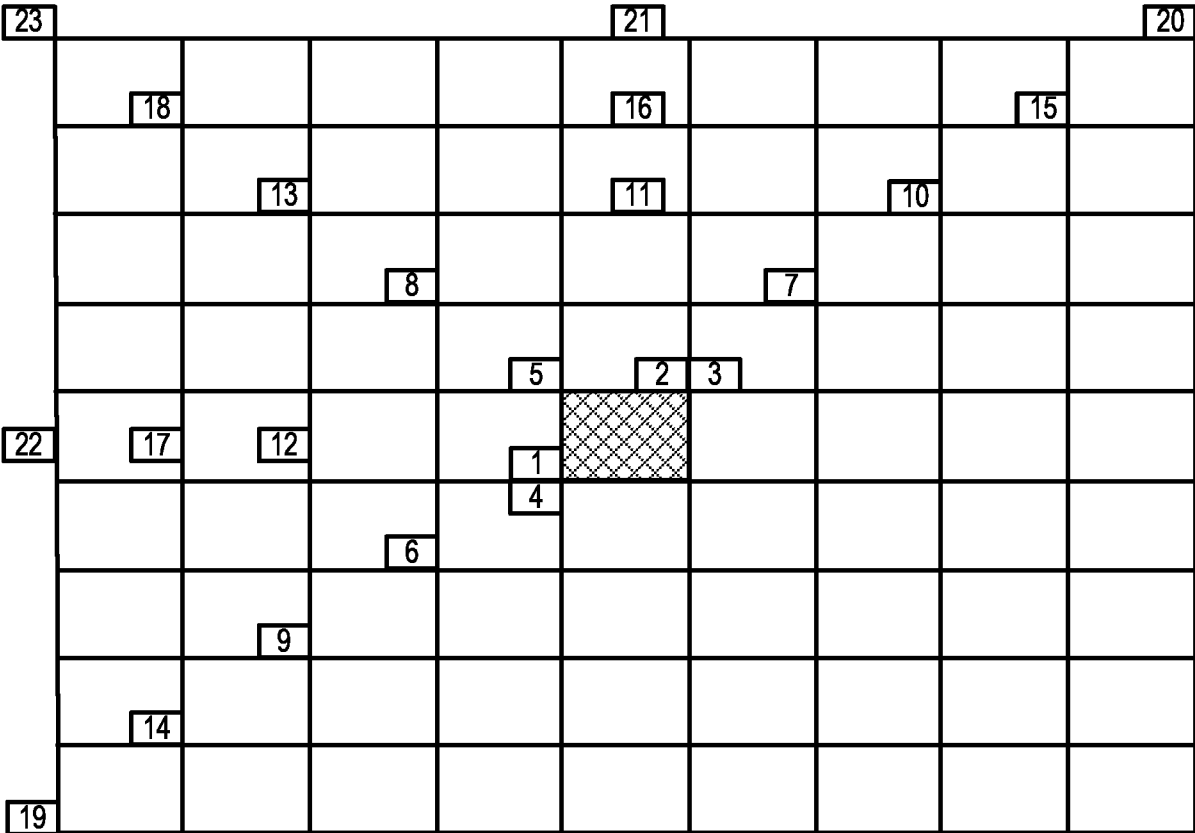


Fig. 33

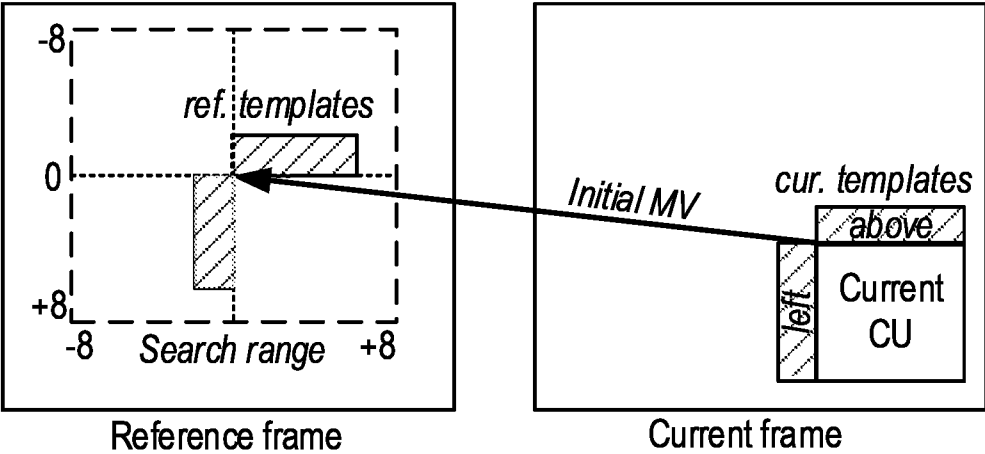


Fig. 34

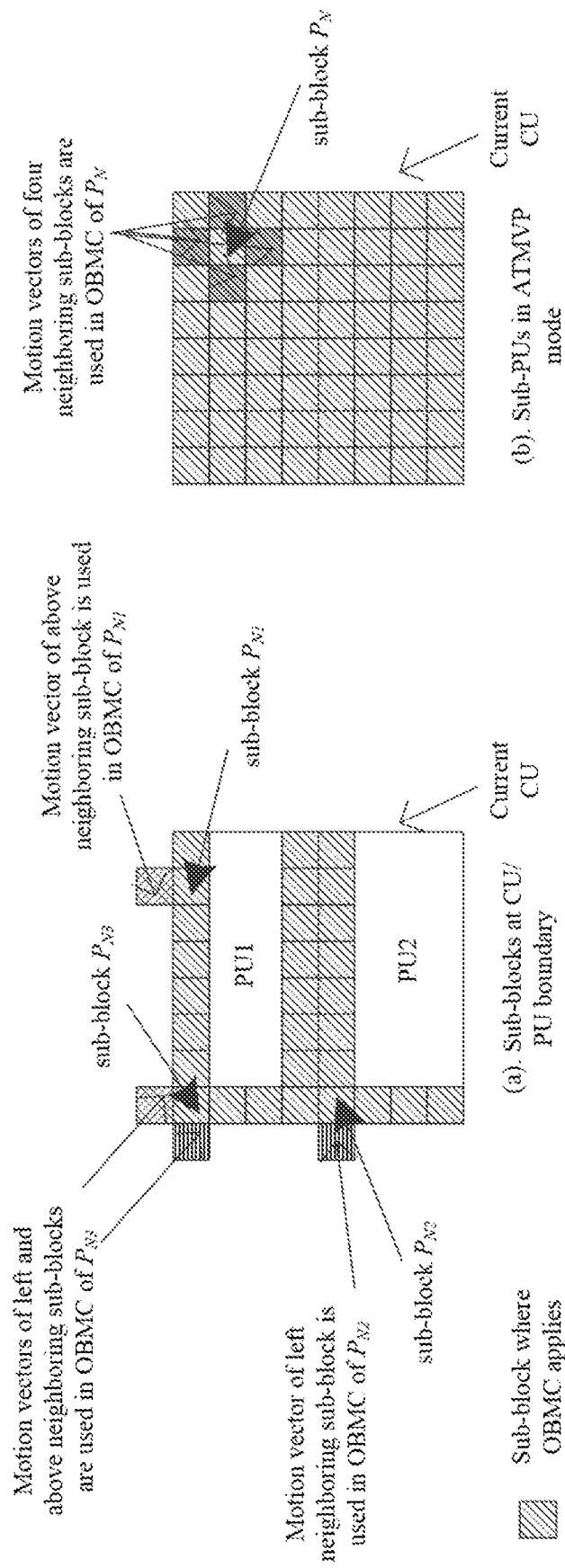


Fig. 35

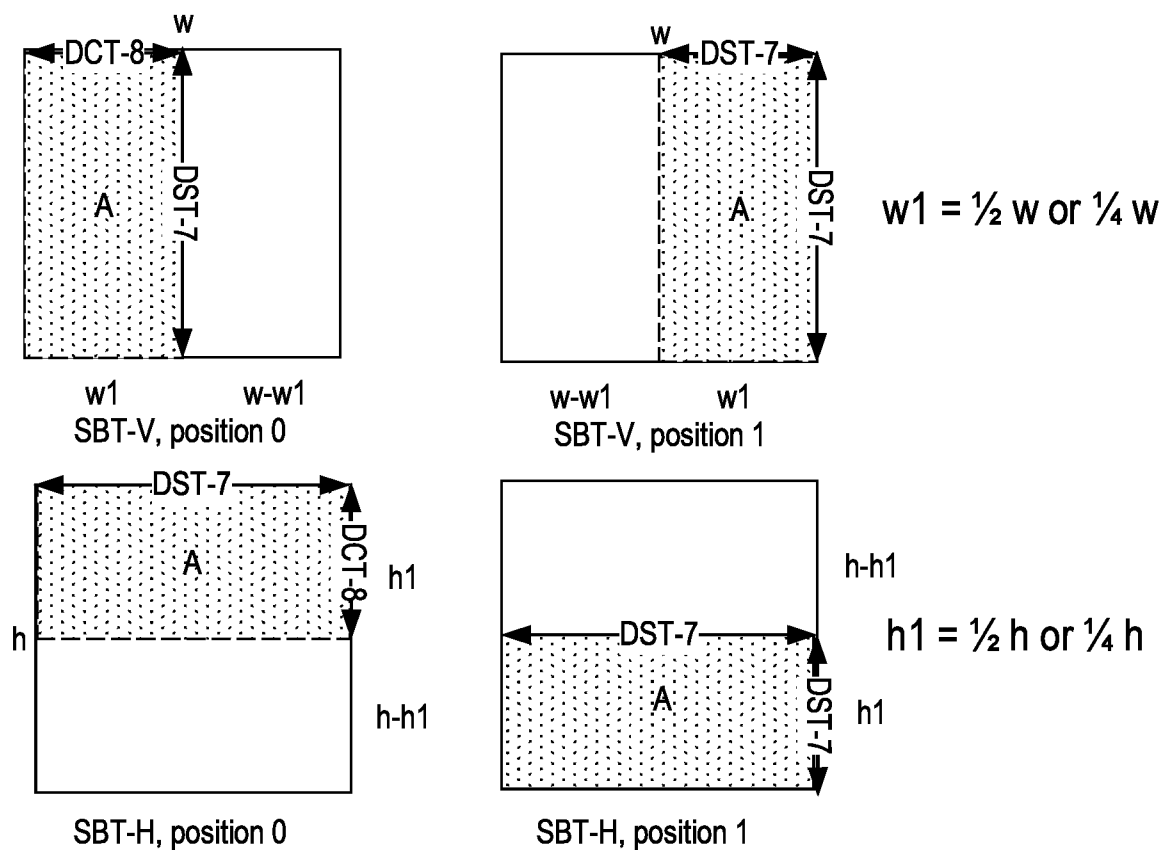


Fig. 36

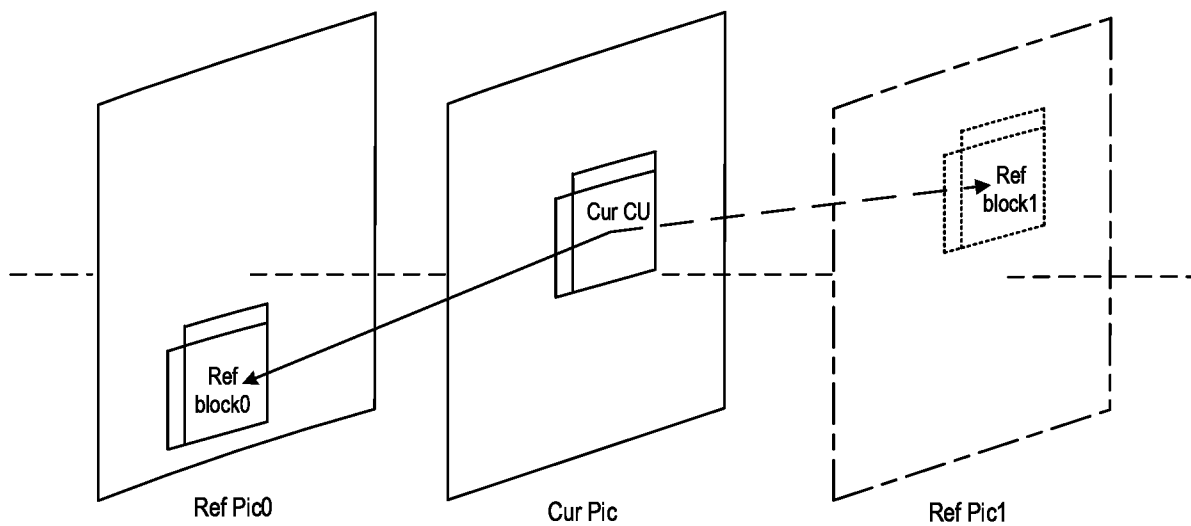


Fig. 37

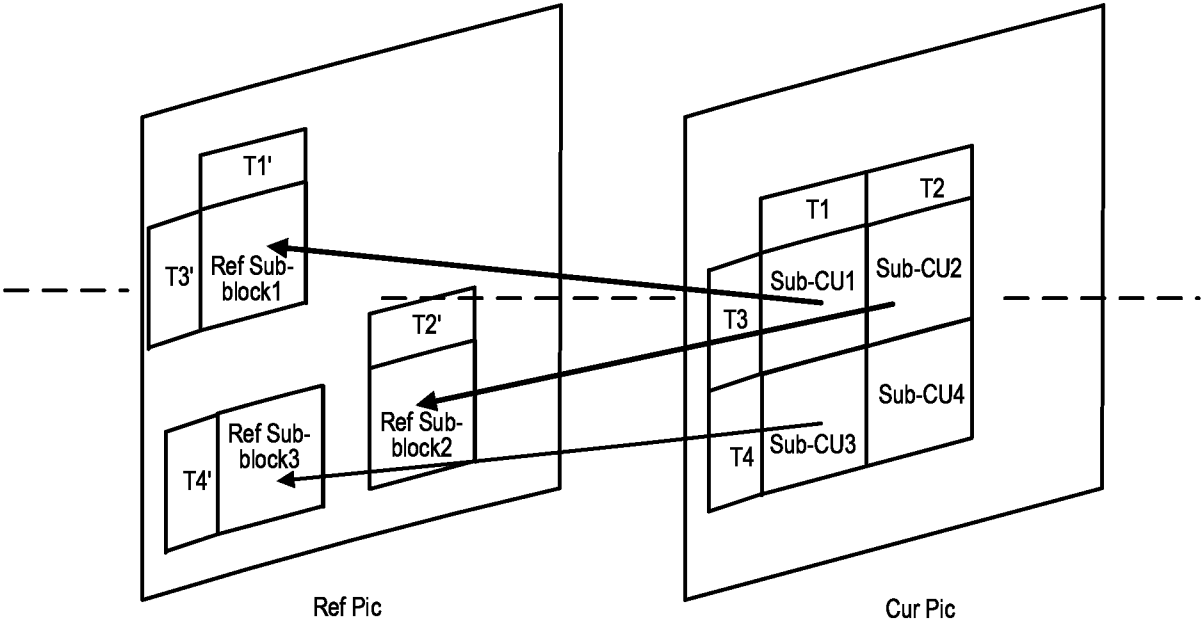


Fig. 38

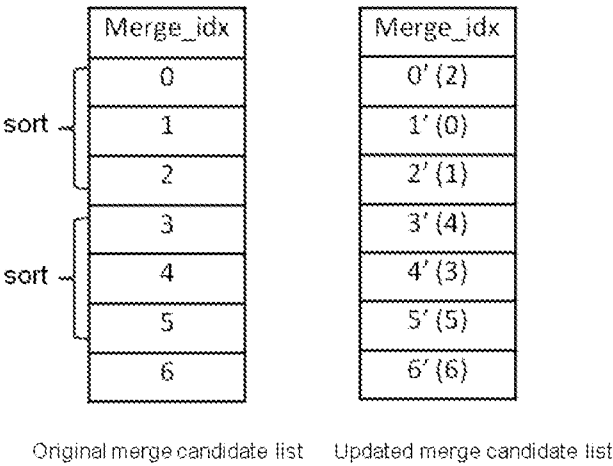
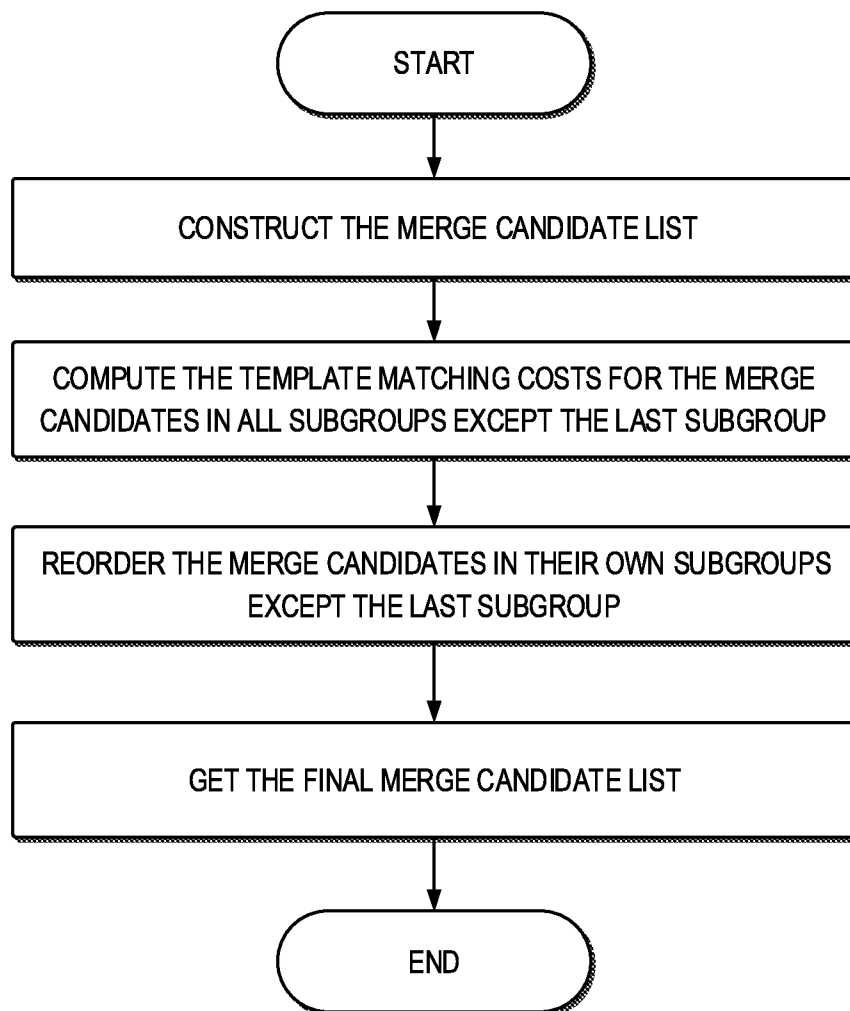
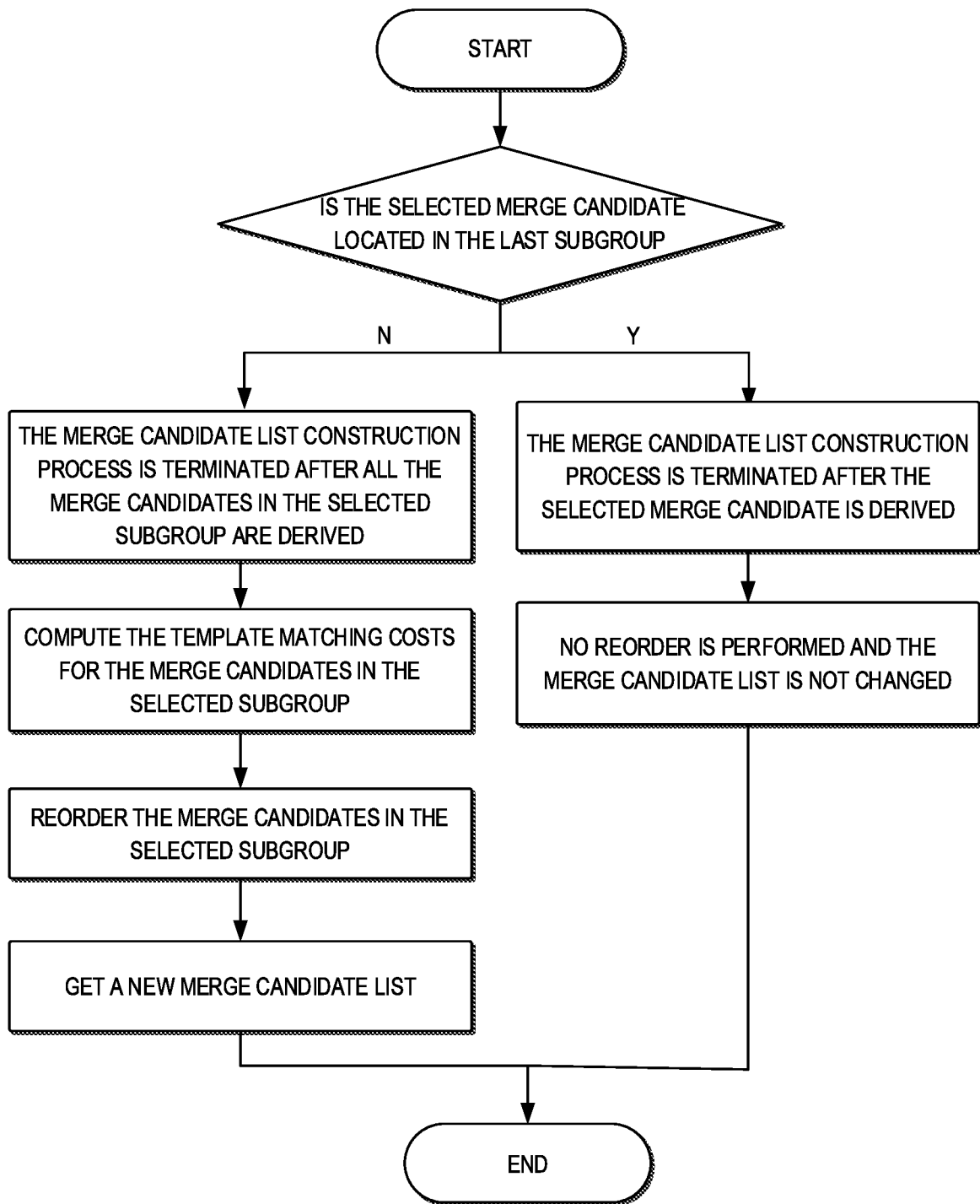


Fig. 39

**Fig. 40**

**Fig. 41**

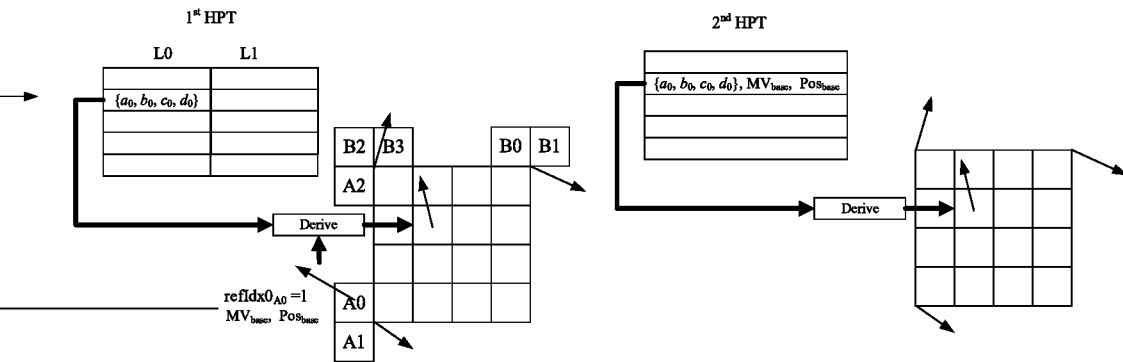


Fig. 42

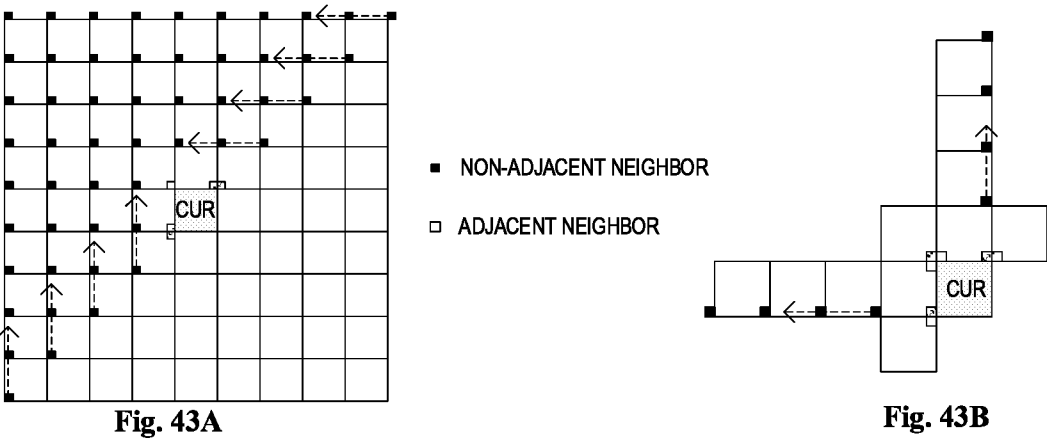


Fig. 43A

Fig. 43B

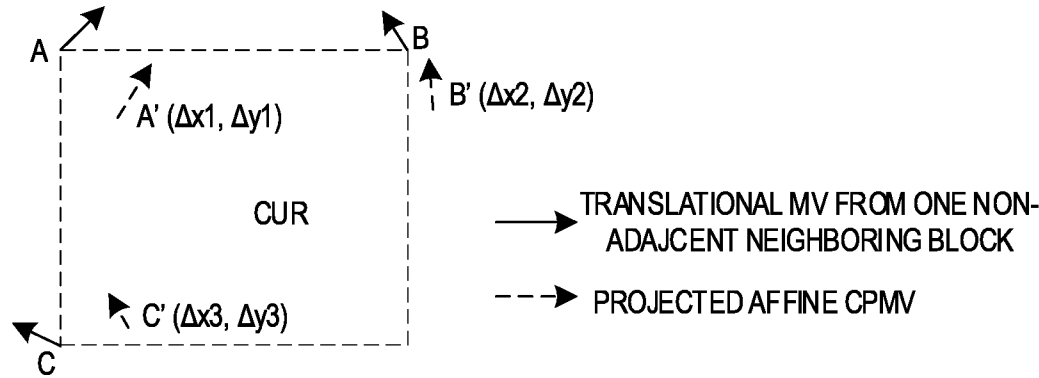


Fig. 44

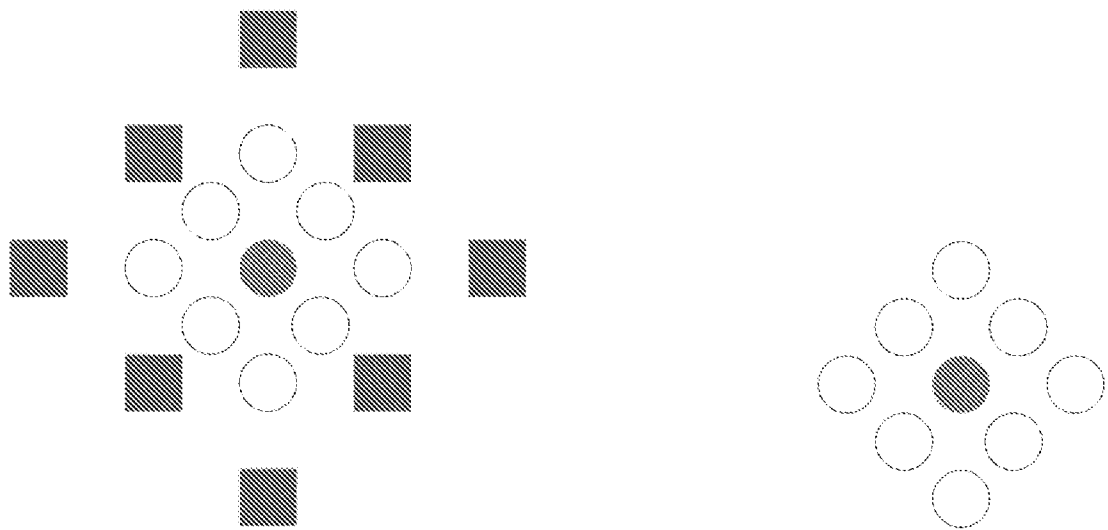
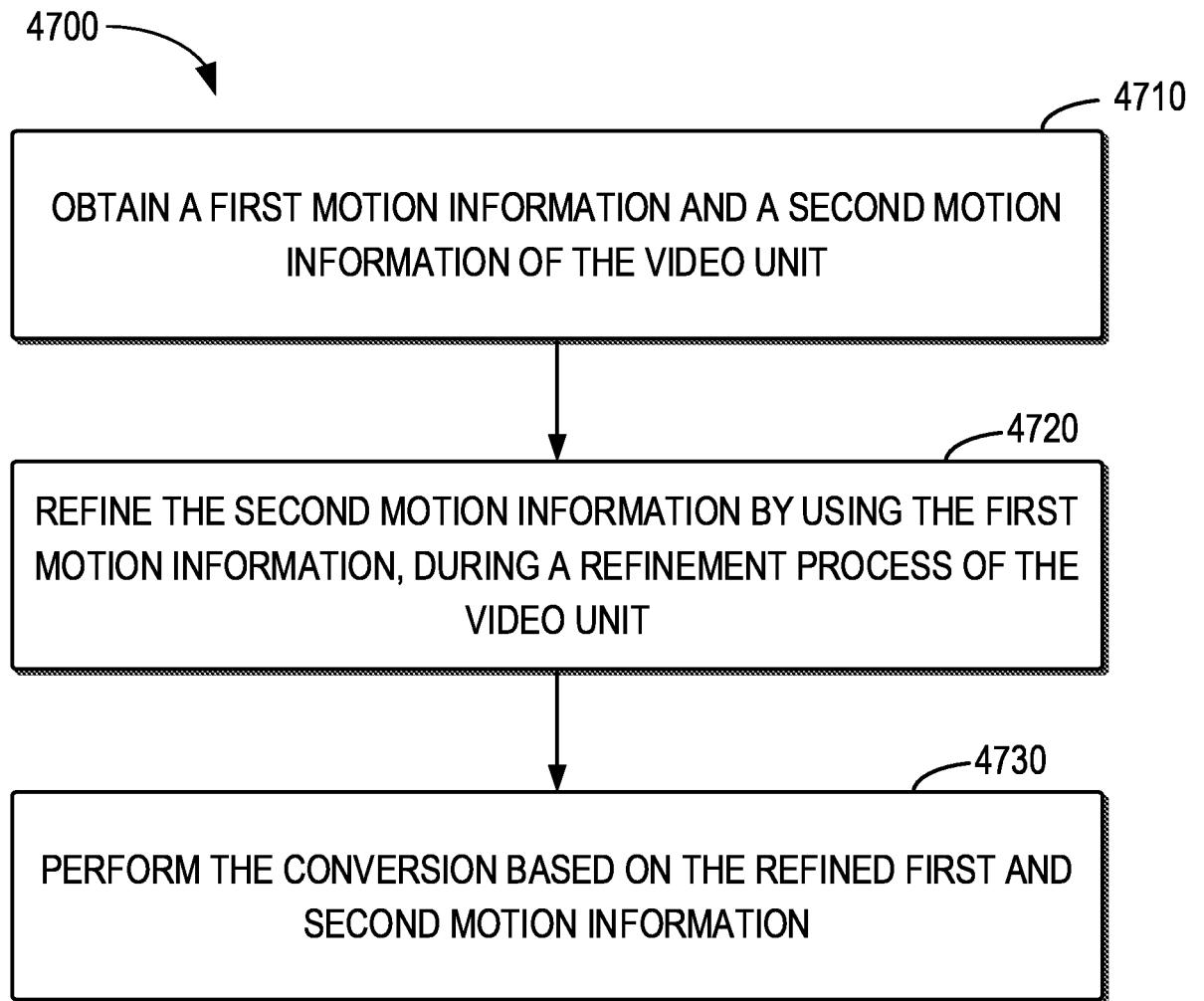
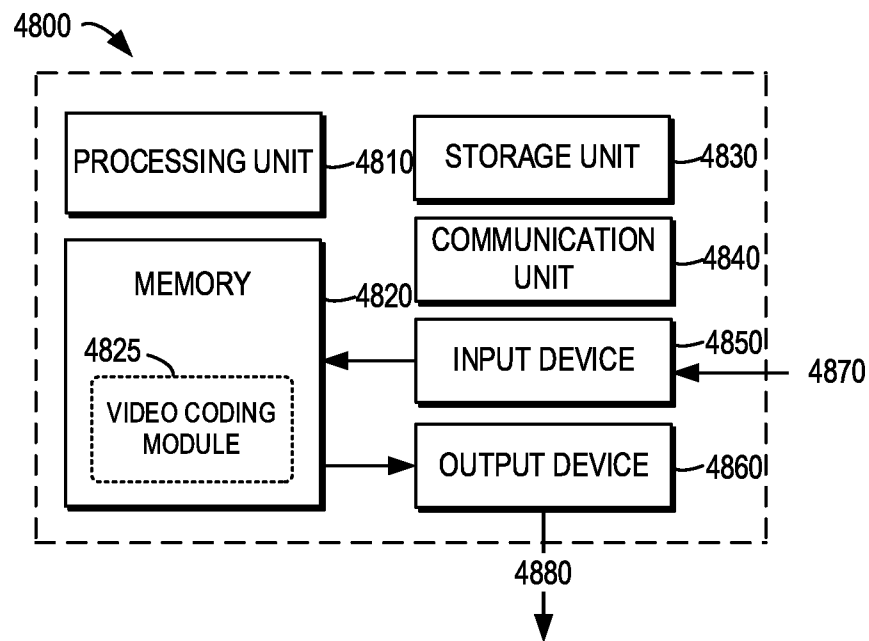


Fig. 45



Fig. 46

**Fig. 47**

**Fig. 48**

INTERNATIONAL SEARCH REPORT

International application No.

PCT/CN2024/097636

A. CLASSIFICATION OF SUBJECT MATTER H04N19/52(2014.01)i According to International Patent Classification (IPC) or to both national classification and IPC		
B. FIELDS SEARCHED Minimum documentation searched (classification system followed by classification symbols) 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,ENTXT,VEN,ENTXTC,JVET: motion,vector,information,MV, refine,DMVR,BDOF,iterative,multi-pass,MP ,bilateral,template,use,enable,POC,QP		
C. DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2020128258 A1 (MEDIATEK INC) 23 April 2020 (2020-04-23) the description paragraphs 69-127, figs.1-20	1-8,12-23,27-33
Y	US 2020128258 A1 (MEDIATEK INC) 23 April 2020 (2020-04-23) the description paragraphs 69-127, figs.1-20	9-11,24-26
Y	US 2019089961 A1 (INTELLECTUAL DISCOVERY CO LTD) 21 March 2019 (2019-03-21) the description paragraphs 45-200, figs.7-11	9-11,24-26
X	WO 2022222988 A1 (BEIJING BYTEDANCE NETWORK TECH CO LTD et al.) 27 October 2022 (2022-10-27) page 22 line 7 to page 45 line 11	1-8,12-23,27-33
Y	WO 2022222988 A1 (BEIJING BYTEDANCE NETWORK TECH CO LTD et al.) 27 October 2022 (2022-10-27) page 22 line 7 to page 45 line 11	9-11,24-26
A	WO 2019072373 A1 (HUAWEI TECH CO LTD et al.) 18 April 2019 (2019-04-18) the whole document	1-33
☒ 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 11 September 2024		Date of mailing of the international search report 19 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 SHANG,Qin Telephone No. (+86) 010-62089385

INTERNATIONAL SEARCH REPORT

International application No.

PCT/CN2024/097636

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2022210438 A1 (QUALCOMM INC) 30 June 2022 (2022-06-30) the whole document	1-33
A	US 2021227246 A1 (BEIJING BYTEDANCE NETWORK TECH CO LTD et al.) 22 July 2021 (2021-07-22) the whole document	1-33

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.

PCT/CN2024/097636

Patent document cited in search report			Publication date (day/month/year)	Patent family member(s)			Publication date (day/month/year)
US	2020128258	A1	23 April 2020	WO	2018121506	A1	05 July 2018
				EP	3560202	A1	30 October 2019
				EP	3560202	A4	29 July 2020
				US	10911761	B2	02 February 2021
				TW	201832560	A	01 September 2018
				TWI	677237	B	11 November 2019
US	2019089961	A1	21 March 2019	EP	4072141	A1	12 October 2022
				US	10778987	B2	15 September 2020
				US	2022303552	A1	22 September 2022
				US	11973960	B2	30 April 2024
				EP	3435673	A2	30 January 2019
				EP	3435673	A4	25 December 2019
				US	2022303553	A1	22 September 2022
				US	11770539	B2	26 September 2023
				US	2020374529	A1	26 November 2020
				US	11388420	B2	12 July 2022
				WO	2017164645	A2	28 September 2017
				WO	2017164645	A3	16 November 2017
				US	2023353757	A1	02 November 2023
WO	2022222988	A1	27 October 2022	US	2024196001	A1	13 June 2024
				WO	2022222989	A1	27 October 2022
				WO	2022222990	A1	27 October 2022
				US	2024196002	A1	13 June 2024
				US	2024187633	A1	06 June 2024
WO	2019072373	A1	18 April 2019	WO	2019072422	A1	18 April 2019
				EP	3685583	A1	29 July 2020
US	2022210438	A1	30 June 2022	US	11758151	B2	12 September 2023
US	2021227246	A1	22 July 2021	None			