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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
09 January 2020 (09.01.2020)

WIPO | PCT

(10) International Publication Number
WO 2020/007291 A1

(51) International Patent Classification:
H04N 19/57 (2014.01)

(21) International Application Number:
PCT/CN2019/094370

(22) International Filing Date:
02 July 2019 (02.07.2019)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
201831024667 02 July 2018 (02.07.2018) IN
201831047744 17 December 2018 (17.12.2018) IN

(71) Applicant: **HUAWEI TECHNOLOGIES CO., LTD.** [CN/CN]; Huawei Administration Building, Bantian, Longgang District, Shenzhen, Guangdong 518129 (CN).

(72) Inventors: **SETHURAMAN, Sriram**; Consulate-1, #1, Richmond Road, Bangalore, Karnataka 560025 (IN). **A, Jeeva Raj**; Consulate-1, #1, Richmond Road, Bangalore, Karnataka 560025 (IN). **KOTECHA, Sagar**; Consulate-1, #1, Richmond Road, Bangalore, Karnataka 560025 (IN).

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

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

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

WO 2020/007291 A1

(54) Title: A VIDEO ENCODER, A VIDEO DECODER AND CORRESPONDING METHODS

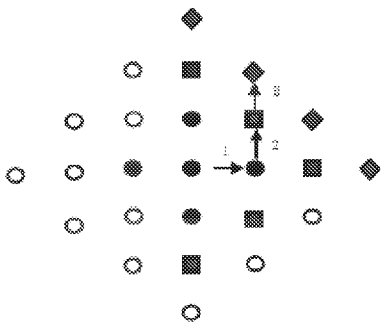

Figure 21

(57) Abstract: A method of decoding is provided, comprising: obtaining an estimate of the motion vector; performing a search within a first search space, wherein searched sample positions are P sample positions adjacent to a position pointed by the estimate of the motion vector in integer-pel distance, and the P sample positions are pointed by a first plurality of candidate motion vectors, wherein P is a positive integer and P is less than 8; performing a search within a n-th search space, wherein searched sample positions are Q sample positions that are adjacent to a position pointed by a (n-1) -th refined motion vector in integer-pel distance and that are not previously searched, and the Q sample positions are pointed by a second plurality of candidate motion vectors, wherein Q is a positive integer and Q is less than 5 or equal to 5; wherein the (n-1) -th refined motion vector is found in the (n-1) -th search space, wherein n is a positive integer which is larger than 1; wherein the motion vector for the image prediction block is found in the n-th search space.

A VIDEO ENCODER, A VIDEO DECODER AND CORRESPONDING METHODS

[0001] The current application claims the priority and benefit of Indian patent application number IN201831024667 filed on 02 July 2018 and Indian patent application number IN201831047744 filed on 17 December 2018.

[0002] Reference is made to documents JVET-K0041-v2, JVET-M0148-v2 and JVET-L0173-v2 of the Joint Video Experts Team (JVET), the contents of which are incorporated by reference, and of which the skilled person will be aware.

TECHNICAL FIELD

[0003] Embodiments of the present application (disclosure) generally relates to the field of video coding, and more particularly to a search method for decoder side motion vector refinement/derivation or a method for determination of a motion vector for a prediction block.

BACKGROUND

[0004] The amount of video data needed to depict even a relatively short video can be substantial, which may result in difficulties when the data is to be streamed or otherwise communicated across a communications network with limited bandwidth capacity. Thus, video data is generally compressed before being communicated across modern day telecommunications networks. The size of a video could also be an issue when the video is stored on a storage device because memory resources may be limited. Video compression devices often use software and/or hardware at the source to code the video data prior to transmission or storage, thereby decreasing the quantity of data needed to represent digital video images. The compressed data is then received at the destination by a video decompression

device that decodes the video data. With limited network resources and ever increasing demands of higher video quality, improved compression and decompression techniques that improve compression ratio with little to no sacrifice in image quality are desirable.

SUMMARY

[0005] Embodiments of the present application (or the present disclosure) provide apparatuses and methods for encoding and decoding.

[0006] Particular embodiments are outlined in the attached independent claims, with other embodiments in the dependent claims.

[0007] According to another aspect the invention relates to an apparatus for determination of a motion vector for a prediction block including a processing circuitry configured to:

obtain an estimate of the motion vector(initial motion vector);

perform a search within a first search space of a first iteration, wherein the first search space is located on a position pointed (given) by the estimate of the motion vector, and wherein searched sample positions are P additional sample positions adjacent to a position (e.g. an initial center position or starting position) corresponding to (e.g. pointed by) the estimate of the motion vector in integer-pel distance(in integer sample distance or integer pixel distance or in integer sample step), and the P additional sample positions are horizontal-only and vertical-only x-pixel distance neighbor positions and the P sample positions are pointed by P candidate motion vectors, wherein P is a positive integer and P is less than 5;

perform a search within a n-th search space of the n-th iteration, wherein the n-the search space is located on a position pointed (given) by a previously-found motion vector, and wherein searched sample positions are Q additional sample positions that are not previously searched, and the Q sample positions includes (consist of) a first set of Q_3 sample positions and a second set of Q_2 sample positions, wherein the Q_3 sample positions are horizontal-only and vertical-only x-pixel distance neighbor positions that are adjacent to a position pointed by a (n-1)-the refined motion vector,

wherein the Q_2 sample positions are diagonal x-pixel neighbor positions to the positions in the first set of sample positions, and the Q sample positions are pointed by Q candidate motion vectors, wherein Q, Q_2, Q_3 is a positive integer and Q is less than 5 or equal to 5(1<Q<=5 (Q=5)), Q_2 is less than 2 or equal to 2, Q_3 is less than 3 or equal to 3; wherein the (n-1)-the refined motion vector (the previously-found motion vector) is found in the (n-1)-the search space, wherein n is a positive integer which is larger than 1;

wherein the motion vector for the image prediction block is found in the n-th search space.

[0008] In one embodiment, wherein the processing circuitry is further configured to: obtain a template for the prediction block and determine a refinement of the estimate of the motion vector (initial motion vector) by template matching with the template in one or more search spaces, wherein Q sample positions are accessible for the template matching within the n-the search space.

[0009] In one embodiment, the processing circuitry is configured to:
calculate the template matching cost for the estimate of the motion vector (initial motion vector) pointing to the starting sample position (e.g. an initial center position or a position pointed by the estimate of the motion vector);
determining the first search space consisting of (P+1) or P sample positions,
calculate template matching cost for the respective candidate motion vector according to a cost function;
comparing template matching cost of the estimate of the motion vector (initial motion vector) with the template matching cost of the respective candidate motion vector, and
performing a search within the second search space if the calculated matching cost of one of the P candidate motion vectors is the lowest cost among the (P+1) or P sample positions.

[0010] In one embodiment, within the first search space, the processing circuitry is further configured to:

calculate the template matching cost for the estimate of the motion vector (initial motion vector) pointing to the starting sample position (e.g. an initial center position or a position pointed by the estimate of the motion vector);

calculate the template matching cost for the respective candidate motion vector according to a cost function;

comparing the template matching cost of the starting sample position with the template matching cost of the respective candidate motion vector, and

determining the motion vector for the prediction block if the template matching cost of the estimate of the motion vector (initial motion vector) is lower than the calculated template matching cost of each of a plurality of the candidate motion vectors.

[0011] In one embodiment, wherein the matching cost comprises bilateral matching cost or template matching cost.

[0012] In one embodiment, wherein the processing circuitry is specifically configured for:

determining the first search space consisting of the P or (P+1) sample positions;

performing template matching in the first search space to obtain a best matching sample position,

determining the n-the search space based on the best matching sample position of the (n-1)-the iteration, wherein n=2,3,…,K, and K is a predefined threshold; and performing template matching in the n-the search space to obtain a best matching position.

[0013] In one exemplary implementation, wherein the processing circuitry is specifically configured to:

determine the first search space consisting of the P or (P+1) sample positions;

perform a search within the first search space to obtain a best matching sample position according to a cost function,

determine then-the search space based on the best matching sample position, wherein n=2,3,…,K, and K is a predefined threshold; and

performing template matching in the n-the search space to obtain a best matching position.

[0014] In one exemplary implementation, wherein if n is equal to a predefined threshold K, the motion vector for the prediction block is found in the n-the search space according to a cost function.

[0015] In one exemplary implementation, wherein the cost function is based on a predetermined template and indicates, for the respective candidate motion vector, a level of similarity between the predetermined template and a predictor pointed to by the respective candidate motion vector.

[0016] In one exemplary implementation, wherein in the n-the search space, a motion vector pointing to a sample position where the cost function is minimized is a part or whole of the motion vector for the prediction block.

[0017] In one exemplary implementation, wherein the first search space consists of P or (P+1) sample positions to which P or (P+1) candidate motion vectors point respectively, and the position pointed by the estimate of the motion vector (initial motion vector) is located substantially in the center of the first search space, P=4.

[0018] In one exemplary implementation, wherein the first search space consists of a central position pointed by the estimate of the motion vector (initial motion vector) and surrounding horizontal-only and vertical-only integer-pixel distance neighbor positions; or, wherein the first search space consists of horizontal-only and vertical-only sample positions surrounding a central position pointed by the estimate of the motion vector (initial motion vector).

[0019] In one exemplary implementation, wherein the first search space consists of a central position pointed by the estimate of the motion vector (initial motion vector) and the left, top, right, and bottom integer-pixel distance positions from the center position pointed by t the estimate of the motion vector (initial motion vector).

[0020] In one exemplary implementation, wherein the n-the search space consists of Q or (Q+1) sample positions to which Q or (Q+1) candidate motion vectors point respectively, and the position pointed by a (n-1)-the refined motion vector (the previously-found motion vector) is located substantially in the center of the n-the search space, 1<Q<=5.

[0021] In one exemplary implementation, wherein the first search space, the n-the search space have integer resolution, wherein n=2, 3,...K, K is a predefined threshold.

[0022] In one exemplary implementation, wherein the processing circuitry is specifically configured to determine the estimate of the motion vector from a list of motion vectors including motion vectors of at least one block adjacent to a current block.

[0023] In one exemplary implementation, wherein the first search space comprises a search space in a first reference picture and a search space in a second reference picture; and/or,

[0024] the n-the search space comprises an_1-th search space in a first reference picture and an_2-th search space in a second reference picture.

[0025] In one exemplary implementation, wherein the Q_2 positions are diagonal 1-pixel neighbor positions to the positions in the first set of positions and are within a current iteration count of distance from the position (e.g. an initial center position) pointed by the estimate of the motion vector, wherein the current iteration count is equal to n.

[0026] In one exemplary implementation, wherein x-pixel distance= 1-pixel distance, or x=1.

[0027] According to another aspect the invention relates to an apparatus including one or more processing circuitry configured for:

- evaluating a cost function at the refinement center and surrounding neighbor positions (such as surrounding horizontal-only and vertical-only x-pixel distance neighbor positions);

- determining the lowest cost position among these evaluated positions;

- performing subsequent iterations of refinement centered on the lowest cost position of the preceding iteration comprising the steps of:

- determining whether the maximum number of iterations has been exceeded and exiting the refinement loop when it has exceeded;

- evaluating the cost function at Q additional positions, wherein the Q sample positions include a first set of Q_3 additional positions, wherein the Q_3 positions are the horizontal-only and vertical-only x-pixel distance neighbor positions which are of the current iteration center and at which the cost function has not been evaluated in any of the previous iterations, and the Q sample positions further include a second set of Q_2 additional positions that are diagonal x-pixel neighbor positions to the positions in the first set of additional positions ;

- determining the lowest cost position among the center and these remaining positions at which the cost function has not been evaluated in any of the previous iterations;

- determining whether the lowest cost position is the current iteration's center position and exiting the refinement loop if the lowest cost position is the current iteration's center position;

wherein Q, Q_2, Q_3 is a positive integer and Q is less than or equal to 5, Q_2 is less than or equal to 2, and Q_3 is less than or equal to 3.

[0028] In a possible implementation form of the apparatus according to the aspect as such, wherein the Q sample positions consist of the first set of Q_3 positions and the second set of Q_2 positions, and the second set of Q_2 additional positions are diagonal 1-pixel neighbor positions to the positions in the first set of positions and are within current iteration count of distance from the initial center of refinement, wherein the initial center of refinement(e.g. an initial center position) is the position pointed by the estimate of the motion vector(e.g. the initial motion vector).

[0029] In a possible implementation form of the apparatus according to the aspect as such, wherein x-pixel distance= 1-pixel distance, or x=1.

[0030] According to another aspect the invention relates to an apparatus including a processing circuitry configured for:

- evaluating a cost function at the refinement center and surrounding horizontal-only and vertical-only 1-pixel distance neighbor positions
- determining the lowest cost position among these evaluated positions
- performing subsequent iterations of refinement centered on the lowest cost position of the preceding iteration comprising the steps of:
    - determining whether the prescribed maximum number of iterations has been exceeded and exiting the refinement loop when it has exceeded
    - evaluating the cost function at up to 5 additional positions selected to include a first set of up to 3 additional positions that include the horizontal-only and vertical-only 1-pixel distance neighbor positions of the current iteration center at which the cost function has not been evaluated in any of the previous iterations, and then include a second set of up to 2 additional positions that are diagonal 1-pixel neighbor positions to the positions in the first set of additional positions and those that are within current iteration count of distance from the initial center of refinement.
    - determining the lowest cost position among the center and these remaining positions at which the cost function has not been evaluated in any of the previous iterations;
- determining whether the lowest cost position is the current iteration's center position and exiting the refinement loop if the lowest cost position is the current iteration's center position.

[0031] An encoding apparatus for encoding video images split to prediction blocks into a bitstream, the encoding apparatus comprising:
the apparatus for determination of a motion vector for a prediction block;
an encoding circuitry for encoding difference between the prediction block and the predictor given by a prediction block in a position based on the determined motion

vector and for generating bitstream including the encoded difference and the estimate of the motion vector (initial motion vector).

[0032] A decoding apparatus for decoding from a bitstream video images split to prediction blocks, the decoding apparatus comprising:

a parsing unit for parsing from the bitstream an estimate of the motion vector (initial motion vector) and an encoded difference between a prediction block and a predictor given by a prediction block in a position specified by a refined motion vector;

the apparatus for determination of the refined motion vector for the prediction block;

decoding circuitry for reconstructing the prediction block as a sum of the parsed difference and the predictor given by the prediction block in the position specified by the refined motion vector.

[0033] A method for determining a motion vector for an image prediction block, the method comprising:

obtaining an estimate of the motion vector (initial motion vector);

performing a search within a first search space of a first iteration, wherein the first search space is located on a position pointed (given) by the estimate of the motion vector, and wherein searched sample positions are P additional sample positions adjacent to a position (e.g. an initial center position) pointed by the estimate of the motion vector in integer-pel distance(in integer sample distance or integer pixel distance or in integer sample step), and the P additional sample positions are horizontal-only and vertical-only x-pixel distance neighbor positions and the P sample positions are pointed by P candidate motion vectors, wherein P is a positive integer and P is less than 5;

performing a search within a n-th search space, wherein searched sample positions are Q additional sample positions that are not previously searched, and the Q sample positions includes(e.g. consist of) a first set of Q_3 sample positions and a second set of Q_2 sample positions, wherein the Q_3 sample positions are horizontal-only and vertical-only x-pixel distance neighbor positions that are adjacent to a position pointed by a (n-1)-th refined motion vector, wherein the Q_2 sample

positions are diagonal x-pixel neighbor positions to the positions in the first set of sample positions, and the Q sample positions are pointed by Q candidate motion vectors, wherein Q, Q_2, Q_3 is a positive integer and Q is less than or equal to 5, Q_2 is less than or equal to 2, Q_3 is less than or equal to 3; wherein the (n-1)-th refined motion vector (the previously-found motion vector) is found in the (n-1)-th search space, wherein n is a positive integer which is larger than 1;

wherein the motion vector for the image prediction block is found in the n-th search space.

[0034] In one exemplary implementation, wherein the obtaining the estimate of the motion vector (initial motion vector), comprises: obtaining an initial motion vector and a template for the prediction block.

[0035] The method according to claim 1, the performing a search within a first search space, comprises:

calculate the template matching cost for the estimate of the motion vector (initial motion vector) pointing to the starting sample position;

determining the first search space consisting of (P+1) or P sample positions,

calculate template matching cost for the respective candidate motion vector according to a cost function;

comparing template matching cost of the estimate of the motion vector (initial motion vector) with the template matching cost of the respective candidate motion vector, and

performing a search within the second search space if the calculated matching cost of one of the P candidate motion vectors is the lowest cost among the (P+1) or P sample positions.

[0036] In one exemplary implementation, within the first search space, the performing a search within a first search space, further comprises:

calculate the template matching cost for the estimate of the motion vector (initial motion vector) pointing to the starting sample position;

calculate the template matching cost for the respective candidate motion vector according to a cost function;

comparing the template matching cost of the starting sample position with the template matching cost of the respective candidate motion vector, and

determining the motion vector for the prediction block if the template matching cost of the estimate of the motion vector (initial motion vector) is lower than the calculated template matching cost of each of a plurality of the candidate motion vectors.

[0037] In one exemplary implementation, wherein the matching cost comprises bilateral matching cost or template matching cost.

[0038] In one exemplary implementation, wherein the performing a search within a first search space, comprises:

determining the first search space consisting of the P or (P+1) sample positions;

performing template matching in the first search space to obtain a best matching sample position,

the performing a search within a n-th search space, comprises:

determining the n-th search space based on the best matching sample position of the (n-1)-th iteration, wherein n=2,3,…,K, and K is a predefined threshold; and

performing template matching in the n-th search space to obtain a best matching position.

[0039] In one exemplary implementation, wherein if n is equal to a predefined threshold K, the motion vector for the prediction block is found in the n-th search space according to a cost function.

[0040] In one exemplary implementation, wherein the cost function is based on a predetermined template and indicates, for the respective candidate motion vector, a level of similarity between the predetermined template and a predictor pointed to by the respective candidate motion vector.

[0041] In one exemplary implementation, wherein in the n-th search space, a motion vector pointing to a sample position where the cost function is minimized is a part or whole of the motion vector for the prediction block.

[0042] In one exemplary implementation, wherein the first search space consists of P or (P+1) sample positions to which P or (P+1) candidate motion vectors point

respectively, and the position pointed by the estimate of the motion vector (initial motion vector) is located substantially in the center of the first search space, P=4.

[0043] In one exemplary implementation, wherein the first search space consists of a central position pointed by the estimate of the motion vector (initial motion vector) and surrounding horizontal-only and vertical-only integer-pixel distance neighbor positions; or, wherein the first search space consists of horizontal-only and vertical-only sample positions surrounding a central position pointed by the estimate of the motion vector (initial motion vector).

[0044] In one exemplary implementation, wherein the first search space consists of a central position pointed by the estimate of the motion vector (initial motion vector) and the left, top, right, and bottom integer-pixel distance positions from the center position pointed by t the estimate of the motion vector (initial motion vector).

[0045] In one exemplary implementation, wherein the n-th search space consists of Q or (Q+1) sample positions to which Q or (Q+1) candidate motion vectors point respectively, and the position pointed by a (n-1)-th refined motion vector (the previously-found motion vector) is located substantially in the center of the n-th search space, $1<Q<=5$.

[0046] In one exemplary implementation, wherein the first search space, the n-th search space have integer resolution, wherein n=2, 3,...K, K is a predefined threshold.

[0047] In one exemplary implementation, wherein the obtaining an estimate of the motion vector (initial motion vector) comprises: determining the estimate of the motion vector from a list of motion vectors including motion vectors of at least one block adjacent to a current block.

In one exemplary implementation, wherein the first search space comprises a search space in a first reference picture and a search space in a second reference picture; and/or,

the n-th search space comprises an_1-th search space in a first reference picture and an_2-th search space in a second reference picture.

[0048] In one exemplary implementation, wherein the method is used for Decoder Side Motion Vector Refinement (DMVR) or Decoder Side Motion Vector derivation (DMVD).

[0049] In one exemplary implementation, wherein the Q_2 positions are diagonal 1-pixel neighbor positions to the positions in the first set of positions and are within a current iteration count of distance from the position (e.g. an initial center position) pointed by the estimate of the motion vector, wherein the current iteration count is equal to n.

[0050] In one exemplary implementation, wherein x-pixel distance= 1-pixel distance, or x=1.

[0051] According to another aspect the invention relates to an integer distance search method for decoder side motion vector refinement comprising:

- evaluating a cost function at the refinement center and surrounding neighbor positions (such as surrounding horizontal-only and vertical-only x-pixel distance neighbor positions);
- determining the lowest cost position among these evaluated positions;
- performing subsequent iterations of refinement centered on the lowest cost position of the preceding iteration comprising the steps of:
    - determining whether the maximum number of iterations has been exceeded and exiting the refinement loop when it has exceeded
    - evaluating the cost function at Q additional positions, wherein the Q sample positions include a first set of Q_3 additional positions, wherein the Q_3 positions are the horizontal-only and vertical-only x-pixel distance neighbor positions which are of the current iteration center and at which the cost function has not been evaluated in any of the previous iterations, and the Q sample positions further include a second set of Q_2 additional positions that

are diagonal x-pixel neighbor positions to the positions in the first set of additional positions ;

- determining the lowest cost position among the center and these remaining positions at which the cost function has not been evaluated in any of the previous iterations;

- determining whether the lowest cost position is the current iteration's center position and exiting the refinement loop if the lowest cost position is the current iteration's center position;

wherein Q, Q_2, Q_3 is a positive integer and Q is less than or equal to 5, Q_2 is less than or equal to 2, and Q_3 is less than or equal to 3.

[0052] In one exemplary implementation, wherein the Q sample positions consist of the first set of Q_3 positions and the second set of Q_2 positions, and the second set of Q_2 additional positions are diagonal 1-pixel neighbor positions to the positions in the first set of positions and are within current iteration count of distance from the initial center of refinement, wherein the initial center of refinement(e.g. an initial center position) is the position pointed by the estimate of the motion vector(e.g. the initial motion vector).

[0053] In one exemplary implementation, wherein x-pixel distance= 1-pixel distance, or x=1.

[0054] According to a first aspect the invention relates to an integer distance search method for decoder side motion vector refinement comprising:

- Evaluating a cost function at the refinement center and surrounding horizontal-only and vertical-only 1-pixel distance neighbor positions

- Determining the lowest cost position among these evaluated positions

- Performing subsequent iterations of refinement centered on the lowest cost position of the preceding iteration comprising the steps of:

- Determining whether the prescribed maximum number of iterations has been exceeded and exiting the refinement loop when it has exceeded

- Evaluating the cost function at up to 5 additional positions selected to include a first set of up to 3 additional positions that include the horizontal-only and vertical-only 1-pixel distance neighbor positions of the current iteration center at which the cost function has not been evaluated in any of the previous iterations, and then include a second set of up to 2 additional positions that are diagonal 1-pixel neighbor positions to the positions in the first set of additional positions and those that are within current iteration count of distance from the initial center of refinement.

- Determining the lowest cost position among the center and these remaining positions at which the cost function has not been evaluated in any of the previous iterations;

- Determining whether the lowest cost position is the current iteration's center position and exiting the refinement loop if the lowest cost position is the current iteration's center position.

[0055] A method for determining a motion vector for an image prediction block, the method comprising:
Obtaining an estimate of the motion vector (initial motion vector);
performing a search within a first search space, wherein the first search space is defined by a part of a plurality of candidate motion vectors pointing at the sample positions adjacent to the position pointed by the estimate of the motion vector in integer-pel distance;
performing a search within a n-th search space, wherein the n-th search space is defined by un-searched candidate motion vectors of a whole of a plurality of candidate motion vectors pointing at the sample positions adjacent to the position pointed by a previously-found motion vector in integer-pel distance, wherein the previously-found motion vector is found in the (n-1)-th search space according to a cost function, wherein n is a positive integer which is larger or equal to 2;

wherein the motion vector for the image prediction block is found in the n-th search space according to a cost function.

[0056] An encoding method for encoding video images split to prediction blocks into a bitstream, the encoding method comprising:

determining a motion vector for a prediction block;

encoding difference between the prediction block and the predictor given by a prediction block in a position based on the determined motion vector and for generating bitstream including the encoded difference and the initial motion vector.

[0057] A decoding method for decoding from a bitstream video images split to prediction blocks, the decoding method comprising:

parsing from the bitstream an initial motion vector and an encoded difference between a prediction block and a predictor given by a prediction block in a position specified by a refined motion vector;

determining the refined motion vector for the prediction block;

reconstructing the prediction block as a sum of the parsed difference and the predictor given by the prediction block in the position based on the refined motion vector.

[0058] In a second manner, a decoding apparatus is provided, which comprises modules/units/components/circuits to perform at least a part of the steps of the method in the first manner.

[0059] In a third manner, a decoding apparatus is provided, which comprises a memory storing instructions; and a processor coupled to the memory, the processor configured to execute the instructions stored in the memory to cause the processor to perform the method of the first manner.

[0060] In a fourth manner, a computer-readable storage medium is provided, which having a program recorded thereon; where the program makes the computer execute method of the first manner.

[0061] In a fifth manner, computer program is provided, which is configured to cause a computer to execute method of the first manner.

[0062]For the purpose of clarity, any one of the foregoing embodiments may be combined with any one or more of the other foregoing embodiments to create a new embodiment within the scope of the present disclosure.

[0063]These and other features will be more clearly understood from the following detailed description taken in conjunction with the accompanying drawings and claims.

BRIEF DESCRIPTION OF THE DRAWINGS

[0064]For a more complete understanding of this disclosure, reference is now made to the following brief description, taken in connection with the accompanying drawings and detailed description, wherein like reference numerals represent like parts.

[0065]FIG. 1A is a block diagram illustrating an example coding system that may implement embodiments of the invention.

[0066]FIG. 1B is a block diagram illustrating another example coding system that may implement embodiments of the invention.

[0067]FIG. 2 is a block diagram illustrating an example video encoder that may implement embodiments of the invention.

[0068]FIG. 3 is a block diagram illustrating an example of a video decoder that may implement embodiments of the invention.

[0069]FIG. 4 is a schematic diagram of a network device.

[0070]FIG. 5 is a simplified block diagram of an apparatus 500 that may be used as either or both of the source device 12 and the destination device 14 from FIG. 1A according to an exemplary embodiment.

[0071]FIG. 6 is a graphical illustration for MCP based on a translational motion model.

[0072]FIG. 7 is a graphical illustration for an overview block diagram of the HEVC inter-picture prediction.

[0073]FIG. 8 is a graphical illustration for the relationship between spatial MVP candidates and spatially neighboring blocks.

[0074]FIG. 9 is a graphical illustration for the derivation process flow for the two spatial candidates A and B.

[0075]FIG. 10 is a graphical illustration for CTU partitioning.

[0076]FIG. 11 is a graphical illustration for sub-CUs.

[0077]FIG. 12 is a graphical illustration for the relationship between CU and sub-CUs.

[0078]FIG. 13 is a graphical illustration for the bilateral matching to derive motion information of the current CU.

[0079]FIG. 14 is a graphical illustration for template matching to derive motion information of the current CU.

[0080]FIG. 15 is a graphical illustration for the motion associated to the block passing through a 4×4 block.

[0081]FIG. 16 is a graphical illustration forMV0′ and MV1′.

[0082]FIG. 17 is a graphical illustration for3x3 pattern search – (a) First iteration, (b) Sample non-first iteration with 5 additional SAD evaluations, (c) Sample non-first iteration with 3 additional evaluations

[0083]FIG. 18 is a graphical illustration for an alternate prior-art refinement search method.

[0084]FIG. 19 is a graphical illustration for 3x3 pattern search – (a) First iteration – cost evaluation at 5 positions; (b) Illustrative progression to 2nd and 3rd iterations where the 2nd iteration evaluates at 5 additional positions and the 3rd iteration evaluates at another 5 additional positions;

[0085]FIG. 20 is a flowchart illustrating an embodiment of a method;

[0086] FIG. 21 is a graphical illustration for +/-3 rhombus refinement range with average multi-stage cost evaluation;

[0087] FIG. 22 is another flowchart illustrating the refinement process according to an embodiment 5 of the present disclosure.

DETAILED DESCRIPTION

[0088] It should be understood at the outset that although an illustrative implementation of one or more embodiments are provided below, the disclosed systems and/or methods may be implemented using any number of techniques, whether currently known or in existence. The disclosure should in no way be limited to the illustrative implementations, drawings, and techniques illustrated below, including the exemplary designs and implementations illustrated and described herein, but may be modified within the scope of the appended claims along with their full scope of equivalents.

[0089] FIG. 1A is a block diagram illustrating an example coding system 10 that may utilize bidirectional prediction techniques. As shown in FIG. 1A, the coding system 10 includes a source device 12 that provides encoded video data to be decoded at a later time by a destination device 14. In particular, the source device 12 may provide the video data to destination device 14 via a computer-readable medium 16. Source device 12 and destination device 14 may comprise any of a wide range of devices, including desktop computers, notebook (i.e., laptop) computers, tablet computers, set-top boxes, telephone handsets such as so-called "smart" phones, so-called "smart" pads, televisions, cameras, display devices, digital media players, video gaming consoles, video streaming device, or the like. In some cases, source device 12 and destination device 14 may be equipped for wireless communication.

[0090] Destination device 14 may receive the encoded video data to be decoded via computer-readable medium 16. Computer-readable medium 16 may comprise any type of medium or device capable of moving the encoded video data from source device 12 to destination device 14. In one example, computer-readable medium 16 may comprise a communication medium to enable source device 12 to transmit encoded video data directly to destination device 14 in real-time. The encoded video data may be modulated according to a communication standard, such as a wireless communication protocol, and transmitted to destination device 14. The communication medium may comprise any wireless or wired communication

medium, such as a radio frequency (RF) spectrum or one or more physical transmission lines. The communication medium may form part of a packet-based network, such as a local area network, a wide-area network, or a global network such as the Internet. The communication medium may include routers, switches, base stations, or any other equipment that may be useful to facilitate communication from source device 12 to destination device 14.

[0091] In some examples, encoded data may be output from output interface 22 to a storage device. Similarly, encoded data may be accessed from the storage device by input interface. The storage device may include any of a variety of distributed or locally accessed data storage media such as a hard drive, Blu-ray discs, digital video disks (DVD)s, Compact Disc Read-Only Memories (CD-ROMs), flash memory, volatile or non-volatile memory, or any other suitable digital storage media for storing encoded video data. In a further example, the storage device may correspond to a file server or another intermediate storage device that may store the encoded video generated by source device 12. Destination device 14 may access stored video data from the storage device via streaming or download. The file server may be any type of server capable of storing encoded video data and transmitting that encoded video data to the destination device 14. Example file servers include a web server (e.g., for a website), a file transfer protocol (FTP) server, network attached storage (NAS) devices, or a local disk drive. Destination device 14 may access the encoded video data through any standard data connection, including an Internet connection. This may include a wireless channel (e.g., a Wi-Fi connection), a wired connection (e.g., digital subscriber line (DSL), cable modem, etc.), or a combination of both that is suitable for accessing encoded video data stored on a file server. The transmission of encoded video data from the storage device may be a streaming transmission, a download transmission, or a combination thereof.

[0092] The techniques of this disclosure are not necessarily limited to wireless applications or settings. The techniques may be applied to video coding in support of any of a variety of multimedia applications, such as over-the-air television broadcasts, cable television transmissions, satellite television transmissions,

Internet streaming video transmissions, such as dynamic adaptive streaming over HTTP (DASH), digital video that is encoded onto a data storage medium, decoding of digital video stored on a data storage medium, or other applications. In some examples, coding system 10 may be configured to support one-way or two-way video transmission to support applications such as video streaming, video playback, video broadcasting, and/or video telephony.

[0093] In the example of FIG. 1A, source device 12 includes video source 18, video encoder 20, and output interface 22. Destination device 14 includes input interface 28, video decoder 30, and display device 32. In accordance with this disclosure, video encoder 20 of source device 12 and/or the video decoder 30 of the destination device 14 may be configured to apply the techniques for bidirectional prediction. In other examples, a source device and a destination device may include other components or arrangements. For example, source device 12 may receive video data from an external video source, such as an external camera. Likewise, destination device 14 may interface with an external display device, rather than including an integrated display device.

[0094] The illustrated coding system 10 of FIG. 1A is merely one example. Techniques for bidirectional prediction may be performed by any digital video encoding and/or decoding device. Although the techniques of this disclosure generally are performed by a video coding device, the techniques may also be performed by a video encoder/decoder, typically referred to as a "CODEC." Moreover, the techniques of this disclosure may also be performed by a video preprocessor. The video encoder and/or the decoder may be a graphics processing unit (GPU) or a similar device.

[0095] Source device 12 and destination device 14 are merely examples of such coding devices in which source device 12 generates coded video data for transmission to destination device 14. In some examples, source device 12 and destination device 14 may operate in a substantially symmetrical manner such that each of the source and destination devices 12, 14 includes video encoding and decoding components. Hence, coding system 10 may support one-way or two-way

video transmission between video devices 12, 14, e.g., for video streaming, video playback, video broadcasting, or video telephony.

[0096] Video source 18 of source device 12 may include a video capture device, such as a video camera, a video archive containing previously captured video, and/or a video feed interface to receive video from a video content provider. As a further alternative, video source 18 may generate computer graphics-based data as the source video, or a combination of live video, archived video, and computer-generated video.

[0097] In some cases, when video source 18 is a video camera, source device 12 and destination device 14 may form so-called camera phones or video phones. As mentioned above, however, the techniques described in this disclosure may be applicable to video coding in general, and may be applied to wireless and/or wired applications. In each case, the captured, pre-captured, or computer-generated video may be encoded by video encoder 20. The encoded video information may then be output by output interface 22 onto a computer-readable medium 16.

[0098] Computer-readable medium 16 may include transient media, such as a wireless broadcast or wired network transmission, or storage media (that is, non-transitory storage media), such as a hard disk, flash drive, compact disc, digital video disc, Blu-ray disc, or other computer-readable media. In some examples, a network server (not shown) may receive encoded video data from source device 12 and provide the encoded video data to destination device 14, e.g., via network transmission. Similarly, a computing device of a medium production facility, such as a disc stamping facility, may receive encoded video data from source device 12 and produce a disc containing the encoded video data. Therefore, computer-readable medium 16 may be understood to include one or more computer-readable media of various forms, in various examples.

[0099] Input interface 28 of destination device 14 receives information from computer-readable medium 16. The information of computer-readable medium 16 may include syntax information defined by video encoder 20, which is also used by video decoder 30, that includes syntax elements that describe characteristics and/or processing of blocks and other coded units, e.g., group of pictures (GOPs). Display

device 32 displays the decoded video data to a user, and may comprise any of a variety of display devices such as a cathode ray tube (CRT), a liquid crystal display (LCD), a plasma display, an organic light emitting diode (OLED) display, or another type of display device.

[00100] Video encoder 20 and video decoder 30 may operate according to a video coding standard, such as the High Efficiency Video Coding (HEVC) standard presently under development, and may conform to the HEVC Test Model (HM). Alternatively, video encoder 20 and video decoder 30 may operate according to other proprietary or industry standards, such as the International Telecommunications Union Telecommunication Standardization Sector (ITU-T) H.264 standard, alternatively referred to as Motion Picture Expert Group (MPEG)-4, Part 10, Advanced Video Coding (AVC), H.265/HEVC, or extensions of such standards. The techniques of this disclosure, however, are not limited to any particular coding standard. Other examples of video coding standards include MPEG-2 and ITU-T H.263. Although not shown in FIG. 1A, in some aspects, video encoder 20 and video decoder 30 may each be integrated with an audio encoder and decoder, and may include appropriate multiplexer-demultiplexer (MUX-DEMUX) units, or other hardware and software, to handle encoding of both audio and video in a common data stream or separate data streams. If applicable, MUX-DEMUX units may conform to the ITU H.223 multiplexer protocol, or other protocols such as the user datagram protocol (UDP).

[00101] It will thus be understood that terms used in this disclosure may, but not necessarily, have the particular technical meaning and/or definition provided in the HEVC standard or VVC standard. For example, in the HEVC standard, a Coding tree unit (CTU) is the basic processing unit, which conceptually corresponds in structure to macroblock units that were used in several previous video standards. HEVC initially divides the picture into CTUs which are then divided for each luma/chroma component into coding tree blocks (CTBs). A CTB can be 64×64, 32×32, or 16×16 with a larger pixel block size usually increasing the coding efficiency. CTBs are then divided into same size or smaller one or more coding

units (CUs), so that the CTU size is also the largest coding unit size. The arrangement of CUs in a CTB is known as a quadtree since a subdivision results in four smaller regions. CUs are then divided into prediction units (PUs) and / or transform units (Tus) of either intra-picture or inter-picture prediction type which can vary in size from 64×64 to 4×4.

[00102] Video encoder 20 and video decoder 30 each may be implemented as any of a variety of suitable encoder circuitry, such as one or more microprocessors, digital signal processors (DSPs), application specific integrated circuits (ASICs), field programmable gate arrays (FPGAs), discrete logic, software, hardware, firmware or any combinations thereof. When the techniques are implemented partially in software, a device may store instructions for the software in a suitable, non-transitory computer-readable medium and execute the instructions in hardware using one or more processors to perform the techniques of this disclosure. Each of video encoder 20 and video decoder 30 may be included in one or more encoders or decoders, either of which may be integrated as part of a combined encoder/decoder (CODEC) in a respective device. A device including video encoder 20 and/or video decoder 30 may comprise an integrated circuit, a microprocessor, and/or a wireless communication device, such as a cellular telephone.

[00103] Fig. 1B is an illustrative diagram of an example video coding system40 including encoder 20 of fig. 2 and/or decoder 30 of fig. 3 according to an exemplary embodiment. The system 40 can implement techniques of this present application. In the illustrated implementation, video coding system 40 may include imaging device(s) 41, video encoder 20, video decoder 30 (and/or a video coder implemented via logic circuitry 54 of processing unit(s) 46), an antenna 42, one or more processor(s) 43, one or more memory store(s) 44, and/or a display device 45.

[00104] As illustrated, imaging device(s) 41, antenna 42, processing unit(s) 46, logic circuitry 54, video encoder 20, video decoder 30, processor(s) 43, memory store(s) 44, and/or display device 45 may be capable of communication with one

another. As discussed, although illustrated with both video encoder 20 and video decoder 30, video coding system 40 may include only video encoder 20 or only video decoder 30 in various examples.

[00105] As shown, in some examples, video coding system 40 may include antenna 42. Antenna 42 may be configured to transmit or receive an encoded bitstream of video data, for example. Further, in some examples, video coding system 40 may include display device 45. Display device 45 may be configured to present video data. As shown, in some examples, logic circuitry 54 may be implemented via processing unit(s) 46. Processing unit(s) 46 may include application-specific integrated circuit (ASIC) logic, graphics processor(s), general purpose processor(s), or the like. Video coding system 40 also may include optional processor(s) 43, which may similarly include application-specific integrated circuit (ASIC) logic, graphics processor(s), general purpose processor(s), or the like. In some examples, logic circuitry 54 may be implemented via hardware, video coding dedicated hardware, or the like, and processor(s) 43 may implemented general purpose software, operating systems, or the like. In addition, memory store(s) 44 may be any type of memory such as volatile memory (e.g., Static Random Access Memory (SRAM), Dynamic Random Access Memory (DRAM), etc.) or non-volatile memory (e.g., flash memory, etc.), and so forth. In a non-limiting example, memory store(s) 44 may be implemented by cache memory. In some examples, logic circuitry 54 may access memory store(s) 44 (for implementation of an image buffer for example). In other examples, logic circuitry 47 and/or processing unit(s) 46 may include memory stores (e.g., cache or the like) for the implementation of an image buffer or the like.

[00106] In some examples, video encoder 20 implemented via logic circuitry may include an image buffer (e.g., via either processing unit(s) 46 or memory store(s) 44)) and a graphics processing unit (e.g., via processing unit(s) 46). The graphics processing unit may be communicatively coupled to the image buffer. The graphics processing unit may include video encoder 20 as implemented via logic circuitry 47 to embody the various modules as discussed with respect to FIG. 2 and/or any other

encoder system or subsystem described herein. The logic circuitry may be configured to perform the various operations as discussed herein.

[00107] Video decoder 30 may be implemented in a similar manner as implemented via logic circuitry 47 to embody the various modules as discussed with respect to decoder 30 of FIG. 3 and/or any other decoder system or subsystem described herein. In some examples, video decoder 30 may be implemented via logic circuitry may include an image buffer (e.g., via either processing unit(s) 420 or memory store(s) 44)) and a graphics processing unit (e.g., via processing unit(s) 46). The graphics processing unit may be communicatively coupled to the image buffer. The graphics processing unit may include video decoder 30 as implemented via logic circuitry 47 to embody the various modules as discussed with respect to FIG. 3 and/or any other decoder system or subsystem described herein.

[00108] In some examples, antenna 42 of video coding system 40 may be configured to receive an encoded bitstream of video data. As discussed, the encoded bitstream may include data, indicators, index values, mode selection data, or the like associated with encoding a video frame as discussed herein, such as data associated with the coding partition (e.g., transform coefficients or quantized transform coefficients, optional indicators (as discussed), and/or data defining the coding partition). Video coding system 40 may also include video decoder 30 coupled to antenna 42 and configured to decode the encoded bitstream. The display device 45 configured to present video frames.

[00109] FIG. 2 is a block diagram illustrating an example of video encoder 20 that may implement the techniques of the present application. Video encoder 20 may perform intra- and inter-coding of video blocks within video slices. Intra-coding relies on spatial prediction to reduce or remove spatial redundancy in video within a given video frame or picture. Inter-coding relies on temporal prediction to reduce or remove temporal redundancy in video within adjacent frames or pictures of a video sequence. Intra-mode (I mode) may refer to any of several spatial based

coding modes. Inter-modes, such as uni-directional prediction (P mode) or bi-prediction (B mode), may refer to any of several temporal-based coding modes.

[00110] As shown in FIG. 2, video encoder 20 receives a current video block within a video frame to be encoded. In the example of FIG. 2, video encoder 20 includes mode select unit 40, reference frame memory 64, summer 50, transform processing unit 52, quantization unit 54, and entropy coding unit 56. Mode select unit 40, in turn, includes motion compensation unit 44, motion estimation unit 42, intra-prediction unit 46, and partition unit 48. For video block reconstruction, video encoder 20 also includes inverse quantization unit 58, inverse transform unit 60, and summer 62. A deblocking filter (not shown in FIG. 2) may also be included to filter block boundaries to remove blockiness artifacts from reconstructed video. If desired, the deblocking filter would typically filter the output of summer 62. Additional filters (in loop or post loop) may also be used in addition to the deblocking filter. Such filters are not shown for brevity, but if desired, may filter the output of summer 50 (as an in-loop filter).

[00111] During the encoding process, video encoder 20 receives a video frame or slice to be coded. The frame or slice may be divided into multiple video blocks. Motion estimation unit 42 and motion compensation unit 44 perform inter-predictive coding of the received video block relative to one or more blocks in one or more reference frames to provide temporal prediction. Intra-prediction unit 46 may alternatively perform intra-predictive coding of the received video block relative to one or more neighboring blocks in the same frame or slice as the block to be coded to provide spatial prediction. Video encoder 20 may perform multiple coding passes, e.g., to select an appropriate coding mode for each block of video data.

[00112] Moreover, partition unit 48 may partition blocks of video data into sub-blocks, based on evaluation of previous partitioning schemes in previous coding passes. For example, partition unit 48 may initially partition a frame or slice into largest coding units (LCUs), and partition each of the LCUs into sub-coding units (sub-CUs) based on rate-distortion analysis (e.g., rate-distortion optimization). Mode select unit 40 may further produce a quadtree data structure indicative of

partitioning of a LCU into sub-CUs. Leaf-node CUs of the quadtree may include one or more prediction units (PUs) and one or more transform units (TUs).

[00113] The present disclosure uses the term "block" to refer to any of a CU, PU, or TU, in the context of HEVC, or similar data structures in the context of other standards (e.g., macroblocks and sub-blocks thereof in H.264/AVC). A CU includes a coding node, PUs, and TUs associated with the coding node. A size of the CU corresponds to a size of the coding node and is square in shape. The size of the CU may range from 8×8 pixels up to the size of the treeblock with a maximum of 64×64 pixels or greater. Each CU may contain one or more PUs and one or more TUs. Syntax data associated with a CU may describe, for example, partitioning of the CU into one or more PUs. Partitioning modes may differ between whether the CU is skip or direct mode encoded, intra-prediction mode encoded, or inter-prediction mode encoded. PUs may be partitioned to be non-square in shape. Syntax data associated with a CU may also describe, for example, partitioning of the CU into one or more TUs according to a quadtree. In an embodiment, a CU, PU, or TU can be square or non-square (e.g., rectangular) in shape.

[00114] Mode select unit 40 may select one of the coding modes, intra or inter, e.g., based on error results, and provides the resulting intra- or inter-coded block to summer 50 to generate residual block data and to summer 62 to reconstruct the encoded block for use as a reference frame. Mode select unit 40 also provides syntax elements, such as motion vectors, intra-mode indicators, partition information, and other such syntax information, to entropy coding unit 56.

[00115] Motion estimation unit 42 and motion compensation unit 44 may be highly integrated, but are illustrated separately for conceptual purposes. Motion estimation, performed by motion estimation unit 42, is the process of generating motion vectors, which estimate motion for video blocks. A motion vector, for example, may indicate the displacement of a PU of a video block within a current video frame or picture relative to a predictive block within a reference frame (or other coded unit) relative to the current block being coded within the current frame (or other coded unit). A predictive block is a block that is found to closely match

the block to be coded, in terms of pixel difference, which may be determined by sum of absolute difference (SAD), sum of square difference (SSD), or other difference metrics. In some examples, video encoder 20 may calculate values for sub-integer pixel positions of reference pictures stored in reference frame memory 64. For example, video encoder 20 may interpolate values of one-quarter pixel positions, one-eighth pixel positions, or other fractional pixel positions of the reference picture. Therefore, motion estimation unit 42 may perform a motion search relative to the full pixel positions and fractional pixel positions and output a motion vector with fractional pixel precision.

[00116] Motion estimation unit 42 calculates a motion vector for a PU of a video block in an inter-coded slice by comparing the position of the PU to the position of a predictive block of a reference picture. The reference picture may be selected from a first reference picture list (List 0) or a second reference picture list (List 1), each of which identify one or more reference pictures stored in reference frame memory 64. Motion estimation unit 42 sends the calculated motion vector to entropy encoding unit 56 and motion compensation unit 44.

[00117] Motion compensation, performed by motion compensation unit 44, may involve fetching or generating the predictive block based on the motion vector determined by motion estimation unit 42. Again, motion estimation unit 42 and motion compensation unit 44 may be functionally integrated, in some examples. Upon receiving the motion vector for the PU of the current video block, motion compensation unit 44 may locate the predictive block to which the motion vector points in one of the reference picture lists. Summer 50 forms a residual video block by subtracting pixel values of the predictive block from the pixel values of the current video block being coded, forming pixel difference values, as discussed below. In general, motion estimation unit 42 performs motion estimation relative to luma components, and motion compensation unit 44 uses motion vectors calculated based on the luma components for both chroma components and luma components. Mode select unit 40 may also generate syntax elements associated with the video blocks and the video slice for use by video decoder 30 in decoding the video blocks of the video slice.

[00118] Intra-prediction unit 46 may intra-predict a current block, as an alternative to the inter-prediction performed by motion estimation unit 42 and motion compensation unit 44, as described above. In particular, intra-prediction unit 46 may determine an intra-prediction mode to use to encode a current block. In some examples, intra-prediction unit 46 may encode a current block using various intra-prediction modes, e.g., during separate encoding passes, and intra-prediction unit 46 (or mode select unit 40, in some examples) may select an appropriate intra-prediction mode to use from the tested modes.

[00119] For example, intra-prediction unit 46 may calculate rate-distortion values using a rate-distortion analysis for the various tested intra-prediction modes, and select the intra-prediction mode having the best rate-distortion characteristics among the tested modes. Rate-distortion analysis generally determines an amount of distortion (or error) between an encoded block and an original, unencoded block that was encoded to produce the encoded block, as well as a bitrate (that is, a number of bits) used to produce the encoded block. Intra-prediction unit 46 may calculate ratios from the distortions and rates for the various encoded blocks to determine which intra-prediction mode exhibits the best rate-distortion value for the block.

[00120] In addition, intra-prediction unit 46 may be configured to code depth blocks of a depth map using a depth modeling mode (DMM). Mode select unit 40 may determine whether an available DMM mode produces better coding results than an intra-prediction mode and the other DMM modes, e.g., using rate-distortion optimization (RDO). Data for a texture image corresponding to a depth map may be stored in reference frame memory 64. Motion estimation unit 42 and motion compensation unit 44 may also be configured to inter-predict depth blocks of a depth map.

[00121] After selecting an intra-prediction mode for a block (e.g., a conventional intra-prediction mode or one of the DMM modes), intra-prediction unit 46 may provide information indicative of the selected intra-prediction mode for the block to entropy coding unit 56. Entropy coding unit 56 may encode the information indicating the selected intra-prediction mode. Video encoder 20 may include in the

transmitted bitstream configuration data, which may include a plurality of intra-prediction mode index tables and a plurality of modified intra-prediction mode index tables (also referred to as codeword mapping tables), definitions of encoding contexts for various blocks, and indications of a most probable intra-prediction mode, an intra-prediction mode index table, and a modified intra-prediction mode index table to use for each of the contexts.

[00122] Video encoder 20 forms a residual video block by subtracting the prediction data from mode select unit 40 from the original video block being coded. Summer 50 represents the component or components that perform this subtraction operation.

[00123] Transform processing unit 52 applies a transform, such as a discrete cosine transform (DCT) or a conceptually similar transform, to the residual block, producing a video block comprising residual transform coefficient values. Transform processing unit 52 may perform other transforms which are conceptually similar to DCT. Wavelet transforms, integer transforms, sub-band transforms or other types of transforms could also be used.

[00124] Transform processing unit 52 applies the transform to the residual block, producing a block of residual transform coefficients. The transform may convert the residual information from a pixel value domain to a transform domain, such as a frequency domain. Transform processing unit 52 may send the resulting transform coefficients to quantization unit 54. Quantization unit 54 quantizes the transform coefficients to further reduce bit rate. The quantization process may reduce the bit depth associated with some or all of the coefficients. The degree of quantization may be modified by adjusting a quantization parameter. In some examples, quantization unit 54 may then perform a scan of the matrix including the quantized transform coefficients. Alternatively, entropy encoding unit 56 may perform the scan.

[00125] Following quantization, entropy coding unit 56 entropy codes the quantized transform coefficients. For example, entropy coding unit 56 may perform context adaptive variable length coding (CAVLC), context adaptive binary arithmetic coding (CABAC), syntax-based context-adaptive binary arithmetic

coding (SBAC), probability interval partitioning entropy (PIPE) coding or another entropy coding technique. In the case of context-based entropy coding, context may be based on neighboring blocks. Following the entropy coding by entropy coding unit 56, the encoded bitstream may be transmitted to another device (e.g., video decoder 30) or archived for later transmission or retrieval.

[00126] Inverse quantization unit 58 and inverse transform unit 60 apply inverse quantization and inverse transformation, respectively, to reconstruct the residual block in the pixel domain, e.g., for later use as a reference block. Motion compensation unit 44 may calculate a reference block by adding the residual block to a predictive block of one of the frames of reference frame memory 64. Motion compensation unit 44 may also apply one or more interpolation filters to the reconstructed residual block to calculate sub-integer pixel values for use in motion estimation. Summer 62 adds the reconstructed residual block to the motion compensated prediction block produced by motion compensation unit 44 to produce a reconstructed video block for storage in reference frame memory 64. The reconstructed video block may be used by motion estimation unit 42 and motion compensation unit 44 as a reference block to inter-code a block in a subsequent video frame.

[00127] Other structural variations of the video encoder 20 can be used to encode the video stream. For example, a non-transform based encoder 20 can quantize the residual signal directly without the transform processing unit 52 for certain blocks or frames. In another implementation, an encoder 20 can have the quantization unit 54 and the inverse quantization unit 58 combined into a single unit.

[00128] FIG. 3 is a block diagram illustrating an example of video decoder 30 that may implement the techniques of this present application. In the example of FIG. 3, video decoder 30 includes an entropy decoding unit 70, motion compensation unit 72, intra-prediction unit 74, inverse quantization unit 76, inverse transformation unit 78, reference frame memory 82, and summer 80. Video decoder 30 may, in some examples, perform a decoding pass generally reciprocal to the encoding pass described with respect to video encoder 20 (FIG. 2). Motion compensation unit 72 may generate prediction data based on motion vectors

received from entropy decoding unit 70, while intra-prediction unit 74 may generate prediction data based on intra-prediction mode indicators received from entropy decoding unit 70.

[00129] During the decoding process, video decoder 30 receives an encoded video bitstream that represents video blocks of an encoded video slice and associated syntax elements from video encoder 20. Entropy decoding unit 70 of video decoder 30 entropy decodes the bitstream to generate quantized coefficients, motion vectors or intra-prediction mode indicators, and other syntax elements. Entropy decoding unit 70 forwards the motion vectors and other syntax elements to motion compensation unit 72. Video decoder 30 may receive the syntax elements at the video slice level and/or the video block level.

[00130] When the video slice is coded as an intra-coded (I) slice, intra prediction unit 74 may generate prediction data for a video block of the current video slice based on a signaled intra prediction mode and data from previously decoded blocks of the current frame or picture. When the video frame is coded as an inter-coded (i.e., B, P, or GPB) slice, motion compensation unit 72 produces predictive blocks for a video block of the current video slice based on the motion vectors and other syntax elements received from entropy decoding unit 70. The predictive blocks may be produced from one of the reference pictures within one of the reference picture lists. Video decoder 30 may construct the reference frame lists, List 0 and List 1, using default construction techniques based on reference pictures stored in reference frame memory 82.

[00131] Motion compensation unit 72 determines prediction information for a video block of the current video slice by parsing the motion vectors and other syntax elements, and uses the prediction information to produce the predictive blocks for the current video block being decoded. For example, motion compensation unit 72 uses some of the received syntax elements to determine a prediction mode (e.g., intra- or inter-prediction) used to code the video blocks of the video slice, an inter-prediction slice type (e.g., B slice, P slice, or GPB slice), construction information for one or more of the reference picture lists for the slice, motion vectors for each inter-encoded video block of the slice, inter-prediction

status for each inter-coded video block of the slice, and other information to decode the video blocks in the current video slice.

[00132] Motion compensation unit 72 may also perform interpolation based on interpolation filters. Motion compensation unit 72 may use interpolation filters as used by video encoder 20 during encoding of the video blocks to calculate interpolated values for sub-integer pixels of reference blocks. In this case, motion compensation unit 72 may determine the interpolation filters used by video encoder 20 from the received syntax elements and use the interpolation filters to produce predictive blocks.

[00133] Data for a texture image corresponding to a depth map may be stored in reference frame memory 82. Motion compensation unit 72 may also be configured to inter-predict depth blocks of a depth map.

[00134] As will be appreciated by those in the art, the coding system 10 of FIG. 1A is suitable for implementing various video coding or compression techniques. Some video compression techniques, such as inter prediction, intra prediction, and loop filters, have demonstrated to be effective. Therefore, the video compression techniques have been adopted into various video coding standards, such as H.264/AVC and H.265/HEVC.

[00135] Various coding tools such as adaptive motion vector prediction (AMVP) and merge mode (MERGE) are used to predict motion vectors (MVs) and enhance inter prediction efficiency and, therefore, the overall video compression efficiency.

[00136] The MVs noted above are utilized in bi-prediction. In a bi-prediction operation, two prediction blocks are formed. One prediction block is formed using a MV of list0 (referred to herein as MV0). Another prediction block is formed using a MV of list1 (referred to herein as MV1). The two prediction blocks are then combined (e.g., averaged) in order to form a single prediction signal (e.g., a prediction block or a predictor block).

[00137] Other variations of the video decoder 30 can be used to decode the compressed bitstream. For example, the decoder 30 can produce the output video stream without the loop filtering unit. For example, a non-transform based decoder 30 can inverse-quantize the residual signal directly without the inverse-transform

processing unit 78 for certain blocks or frames. In another implementation, the video decoder 30 can have the inverse-quantization unit 76 and the inverse-transform processing unit 78 combined into a single unit.

[00138] FIG. 4 is a schematic diagram of a network device 400 (e.g., a coding device) according to an embodiment of the disclosure. The network device 400 is suitable for implementing the disclosed embodiments as described herein. In an embodiment, the network device 400 may be a decoder such as video decoder 30 of FIG. 1A or an encoder such as video encoder 20 of FIG. 1A. In an embodiment, the network device 400 may be one or more components of the video decoder 30 of FIG. 1A or the video encoder 20 of FIG. 1A as described above.

[00139] The network device 400 comprises ingress ports 410 and receiver units (Rx) 420 for receiving data; a processor, logic unit, or central processing unit (CPU) 430 to process the data; transmitter units (Tx) 440 and egress ports 450 for transmitting the data; and a memory 460 for storing the data. The network device 400 may also comprise optical-to-electrical (OE) components and electrical-to-optical (EO) components coupled to the ingress ports 410, the receiver units 420, the transmitter units 440, and the egress ports 450 for egress or ingress of optical or electrical signals.

[00140] The processor 430 is implemented by hardware and software. The processor 430 may be implemented as one or more CPU chips, cores (e.g., as a multi-core processor), FPGAs, ASICs, and DSPs. The processor 430 is in communication with the ingress ports 410, receiver units 420, transmitter units 440, egress ports 450, and memory 460. The processor 430 comprises a coding module 470. The coding module 470 implements the disclosed embodiments described above. For instance, the coding module 470 implements, processes, prepares, or provides the various coding operations. The inclusion of the coding module 470 therefore provides a substantial improvement to the functionality of the network device 400 and effects a transformation of the network device 400 to a different state. Alternatively, the coding module 470 is implemented as instructions stored in the memory 460 and executed by the processor 430.

[00141] The memory 460 comprises one or more disks, tape drives, and solid-state drives and may be used as an over-flow data storage device, to store programs when such programs are selected for execution, and to store instructions and data that are read during program execution. The memory 460 may be volatile and/or non-volatile and may be read-only memory (ROM), random access memory (RAM), ternary content-addressable memory (TCAM), and/or static random-access memory (SRAM).

[00142] Fig. 5 is a simplified block diagram of an apparatus 500 that may be used as either or both of the source device 12 and the destination device 14 from Fig. 1A according to an exemplary embodiment. The apparatus 500 can implement techniques of this present application. The apparatus 500 can be in the form of a computing system including multiple computing devices, or in the form of a single computing device, for example, a mobile phone, a tablet computer, a laptop computer, a notebook computer, a desktop computer, and the like.

[00143] A processor 502 in the apparatus 500 can be a central processing unit. Alternatively, the processor 502 can be any other type of device, or multiple devices, capable of manipulating or processing information now-existing or hereafter developed. Although the disclosed implementations can be practiced with a single processor as shown, e.g., the processor 502, advantages in speed and efficiency can be achieved using more than one processor.

[00144] A memory 504 in the apparatus 500 can be a read only memory (ROM) device or a random access memory (RAM) device in an implementation. Any other suitable type of storage device can be used as the memory 504. The memory 504 can include code and data 506 that is accessed by the processor 502 using a bus 512. The memory 504 can further include an operating system 508 and application programs 510, the application programs 510 including at least one program that permits the processor 502 to perform the methods described here. For example, the application programs 510 can include applications 1 through N, which further include a video coding application that performs the methods described here. The apparatus 500 can also include additional memory in the form of a secondary storage 514, which can, for example, be a memory card used with a mobile

computing device. Because the video communication sessions may contain a significant amount of information, they can be stored in whole or in part in the secondary storage 514 and loaded into the memory 504 as needed for processing.

[00145] The apparatus 500 can also include one or more output devices, such as a display 518. The display 518 may be, in one example, a touch sensitive display that combines a display with a touch sensitive element that is operable to sense touch inputs. The display 518 can be coupled to the processor 502 via the bus 512. Other output devices that permit a user to program or otherwise use the apparatus 500 can be provided in addition to or as an alternative to the display 518. When the output device is or includes a display, the display can be implemented in various ways, including by a liquid crystal display (LCD), a cathode-ray tube (CRT) display, a plasma display or light emitting diode (LED) display, such as an organic LED (OLED) display.

[00146] The apparatus 500 can also include or be in communication with an image-sensing device 520, for example a camera, or any other image-sensing device 520 now existing or hereafter developed that can sense an image such as the image of a user operating the apparatus 500. The image-sensing device 520 can be positioned such that it is directed toward the user operating the apparatus 500. In an example, the position and optical axis of the image-sensing device 520 can be configured such that the field of vision includes an area that is directly adjacent to the display 518 and from which the display 518 is visible.

[00147] The apparatus 500 can also include or be in communication with a sound-sensing device 522, for example a microphone, or any other sound-sensing device now existing or hereafter developed that can sense sounds near the apparatus 500. The sound-sensing device 522 can be positioned such that it is directed toward the user operating the apparatus 500 and can be configured to receive sounds, for example, speech or other utterances, made by the user while the user operates the apparatus 500.

[00148] Although FIG. 5 depicts the processor 502 and the memory 504 of the apparatus 500 as being integrated into a single unit, other configurations can be utilized. The operations of the processor 502 can be distributed across multiple

machines (each machine having one or more of processors) that can be coupled directly or across a local area or other network. The memory 504can be distributed across multiple machines such as a network-based memory or memory in multiple machines performing the operations of the apparatus 500. Although depicted here as a single bus, the bus 512of the apparatus 500 can be composed of multiple buses. Further, the secondary storage 514 can be directly coupled to the other components of the apparatus 500 or can be accessed via a network and can comprise a single integrated unit such as a memory card or multiple units such as multiple memory cards. The apparatus 500 can thus be implemented in a wide variety of configurations.

[00149] The present disclosure related to inter-picture prediction, while inter-picture prediction makes use of the temporal correlation between pictures in order to derive a motion-compensated prediction (MCP) for a block of image samples.

[00150] For block-based MCP, a video picture is divided into rectangular blocks. Assuming homogeneous motion inside one block and that moving objects are larger than one block, for each block, a corresponding block in a previously decoded picture can be found that serves as a predictor. The general concept of MCP based on a translational motion model is illustrated in Fig.6. Using a translational motion model, the position of the block in a previously decoded picture is indicated by a motion vector (Δx, Δy), where Δx specifies the horizontal and Δy specifies the vertical displacement relative to the position of the current block. The motion vector (Δx, Δy) could be of fractional sample accuracy to more accurately capture the movement of the underlying object. Interpolation is applied on the reference pictures to derive the prediction signal when the corresponding motion vector has fractional sample accuracy. The previously decoded picture is referred to as the reference picture and indicated by a reference index Δt to a reference picture list. These translational motion model parameters, *i.e.*, motion vectors and reference indices, are further referred to as motion data. Two kinds of inter-picture prediction are allowed in modern video coding standards, namely uni-prediction and bi-prediction.

[00151] In case of bi-prediction, two sets of motion data, i.e., $(\Delta x_0,\Delta y_0,\Delta t_0)and(\Delta x_1,\Delta y_1,\Delta t_1)$, are used to generate two MCPs (possibly from different pictures),which are then combined to get the final MCP. Per default, this is done by averaging but in case of weighted prediction, different weights can be applied to different MCPs, e.g., to compensate for scene fade outs. The reference pictures that can be used in bi-prediction are stored in two separate lists, namely list 0 and list 1. In order to limit the memory bandwidth in slices allowing bi-prediction, the HEVC standard restricts PUs with 4×8 and 8×4 luma prediction blocks to use uni-prediction only. Motion data is derived at the encoder using a motion estimation process. Motion estimation is not specified within video standards so different encoders can utilize different complexity-quality tradeoffs in their implementations.

[00152] An overview block diagram of the HEVC inter-picture prediction is shown inFig.7. The motion data of a block is correlated with the neighboring blocks. To exploit this correlation, motion data is not directly coded in the bitstream but predictively coded based on neighboring motion data. In HEVC, two concepts are used for that. The predictive coding of the motion vectors was improved in HEVC by introducing a new tool called advanced motion vector prediction (AMVP) where the best predictor for each motion block is signaled to the decoder. In addition, a new technique called inter-prediction block merging derives all motion data of a blockfromtheneighboringblocksreplacingthedirectandskipmodesinH.264/AVC.

[00153] Different kinds of inter prediction methods are implemented in the motion data coding module. Generally, the methods are called as inter prediction modes. Several inter prediction modes are introduced as following.

[00154] One of the inter prediction mode is AMVP (Advanced Motion Vector Prediction). As in previous video coding standards, the HEVC motion vectors are coded in terms of horizontal (x) and vertical (y) components as a difference to a so called motion vector predictor (MVP). The calculation of both motion vector difference (MVD) components is shown in Eq. (1.1) and (1.2).

$$MVD_X = \Delta x - MVP_X \quad (1.1)$$

$$MVD_Y = \Delta y - MVP_Y \quad (1.2)$$

[00155] Motion vectors of the current block are usually correlated with the motion vectors of neighboring blocks in the current picture or in the earlier coded pictures. This is because neighboring blocks are likely to correspond to the same moving object with similar motion and the motion of the object is not likely to change abruptly overtime. Consequently, using the motion vectors in neighboring blocks as predictors reduces the size of the signaled motion vector difference. The MVPs are usually derived from already decoded motion vectors from spatial neighboring blocks or from temporally neighboring blocks in the co-located picture. In some cases, the zero motion vector can also be used as MVP. In H.264/AVC, this is done by doing a component wise median of three spatially neighboring motion vectors. Using this approach, no signaling of the predictor is required. Temporal MVPs from a co-located picture are only considered in the so called temporal direct mode of H.264/AVC. The H.264/AVC direct modes are also used to derive other motion data than the motion vectors.

[00156] In HEVC, the approach of implicitly deriving the MVP was replaced by a technique known as motion vector competition, which explicitly signals which MVP from a list of MVPs, is used for motion vector derivation. The variable coding quadtree block structure in HEVC can result in one block having several neighboring blocks with motion vectors as potential MVP candidates. The initial design of AMVP includes five MVPs from three different classes of predictors: three motion vectors from spatial neighbors, the median of the three spatial predictors and a scaled motion vector from a co-located, temporally neighboring block. Furthermore, the list of predictors was modified by reordering to place the most probable motion predictor in the first position and by removing redundant candidates to assure minimal signalling overhead. Then, significant simplifications of the AMVP design are developed such as removing the median predictor, reducing the number of candidates in the list from five to two, fixing the candidate order in the list and reducing the number of redundancy checks. The final design of the AMVP candidate list construction includes the following two MVP candidates:

a. up to two spatial candidate MVPs that are derived from five spatial neighboring blocks; b. one temporal candidate MVPs derived from two temporal, co-located blocks when both spatial candidate MVPs are not available or they are identical; c. zero motion vectors when the spatial, the temporal or both candidates are not available.

[00157] As already mentioned, two spatial MVP candidates A and B are derived from five spatially neighboring blocks which are shown in the right part of Fig.8. The locations of the spatial candidate blocks are the same for both AMVP and inter-prediction block merging. The derivation process flow for the two spatial candidates A and B is depicted in Fig.9. For candidate A, motion data from the two blocks A0 and A1 at the bottom left corner is taken into account in a two pass approach. In the first pass, it is checked whether any of the candidate blocks contain a reference index that is equal to the reference index of the current block. The first motion vector found will be taken as candidate A. When all reference indices from A0 and A1 are pointing to a different reference picture than the reference index of the current block, the associated motion vector cannot be used as is. Therefore, in a second pass, the motion vectors need to be scaled according to the temporal distances between the candidate reference picture and the current reference picture. Eq. (1.3) shows how the candidate motion vector mv_{cand} is scaled according to a scale factor. ScaleFactor is calculated based on the temporal distance between the current picture and the reference picture of the candidate block td and the temporal distance between the current picture and the reference picture of the current block tb. The temporal distance is expressed in terms of difference between the picture order count (POC) values which define the display order of the pictures. The scaling operation is basically the same scheme that is used for the temporal direct mode in H.264/AVC. This factoring allows pre-computation of ScaleFactor at slice-level since it only depends on the reference picture list structure signalled in the slice header. Note that the MV scaling is only performed when the current reference picture and the candidate reference picture are both short term reference pictures. Parameter td is defined as the POC difference between the co-located picture and the reference picture of the co-located candidate block.

$$mv = sign(mv_{cand} \cdot ScaleFactor) \cdot ((|mv_{cand} \cdot ScaleFactor| + 2^{7}) >> 8) \quad (1.3)$$

$$ScaleFactor = clip(-2^{12}, 2^{12}-1, (tb \cdot tx + 2^{5}) >> 6) \quad (1.4)$$

$$tx = \frac{2^{14} + \left\lfloor \frac{|td|}{2} \right\rfloor}{td} \quad (1.5)$$

[00158] For candidate B, the candidates B0 to B2 are checked sequentially in the same way as A0 and A1 which were checked in the first pass. The second pass, however, is only performed when blocks A0 and A1 do not contain any motion information, i.e. are not available or coded using intra-picture prediction. Then, candidate A is set equal to the non-scaled candidate B, if found, and candidate B is set equal to a second, non-scaled or scaled variant of candidate B. The second pass searches for non-scaled as well as for scaled MVs derived from candidates B0to B2. Overall, this design allows to process A0 and A1 independently from B0, B1,and B2. The derivation of B should only be aware of the availability of both A0 andA1 in order to search for a scaled or an additional non-scaled MV derived from B0to B2. This dependency is acceptable given that it significantly reduces the complex motion vector scaling operations for candidate B. Reducing the number of motion vector scaling represents a significant complexity reduction in the motion vector predictor derivation process.

[00159] In HEVC, the block to the bottom right and at the center of the current block have been determined to be the most suitable to provide a good temporal motion vector predictor (TMVP). These candidates are illustrated in the left part of Fig.8where C0 represents the bottom right neighbor and C1 represents the center block. Here again, motion data of C0 is considered first and, if not available, motion data from the co-located candidate block at the center is used to derive the temporal MVP candidate C. The motion data of C0 is also considered as not being

available when the associated PU belongs to a CTU beyond the current CTU row. This minimizes the memory bandwidth requirements to store the co-located motion data. In contrast to the spatial MVP candidates, where the motion vectors may refer to the same reference picture, motion vector scaling is mandatory for the TMVP. Hence, the same scaling operation as for the spatial MVPs is used.

[00160] While the temporal direct mode in H.264/AVC always refers to the first reference picture in the second reference picture list, list 1, and is only allowed in bi-predictive slices, HEVC offers the possibility to indicate for each picture which reference picture is considered as the co-located picture. This is done by signaling in the slice header the co-located reference picture list and reference picture index as well as requiring that these syntax elements in all slices in a picture should specify the same reference picture.

[00161] Since the temporal MVP candidate introduces additional dependencies, it might be desirable to disable its usage for error robustness reasons. In H.264/AVC there is the possibility to disable the temporal direct mode for bi-predictive slices in the slice header (direct_spatial_mv_pred_flag). HEVC syntax extends this signaling by allowing to disable the TMVP at sequence level or at picture level (sps/slice_temporal_mvp_enabled_flag). Although the flag is signaled in the slice header, it is a requirement of bitstream conformance that its value shall be the same for all slices in one picture. Since the signaling of the picture-level flag depends on the SPS flag, signaling it in the PPS would introduce a parsing dependency between SPS and PPS. Another advantage of this slice header signaling is that if you want to change only the value of this flag and no other parameter in the PPS, there is no need to transmit a second PPS.

[00162] In general, motion data signaling in HEVC is similar as in H.264/AVC. An inter-picture prediction syntax element, inter_pred_idc, signals whether reference list 0, 1 or both are used. For each MCP obtained from one reference picture list, the corresponding reference picture (Δt) is signaled by an index to the reference picture list, ref_idx_l0/1, and the MV (Δx, Δy) is represented by an index to the MVP, mvp_l0/1_flag, and its MVD. A newly introduced flag in the slice header, mvd_l1_zero_flag, indicates whether the MVD for the second reference

picture list is equal to zero and therefore not signaled in the bitstream. When the motion vector is fully reconstructed, a final clipping operation assures that the values of each component of the final motion vector will always be in the range of -2^{15} to $2^{15}-1$, inclusive.

[00163] Another one of the inter prediction modes is Inter-picture Prediction Block Merging. The AMVP list only contains motion vectors for one reference list while a merge candidate contains all motion data including the information whether one or two reference picture lists are used as well as a reference index and a motion vector for each list. Overall, the merge candidate list is constructed based on the following candidates: a. up to four spatial merge candidates that are derived from five spatial neighboring blocks; b. one temporal merge candidate derived from two temporal, co-located blocks; c. additional merge candidates including combined bi-predictive candidates and zero motion vector candidates.

[00164] The first candidates in the merge candidate list are the spatial neighbors. Up to four candidates are inserted in the merge list by sequentially checking A1, B1, B0, A0 and B2, in that order, according to the right part of Fig.8.

[00165] Instead of just checking whether a neighboring block is available and contains motion information, some additional redundancy checks are performed before taking all the motion data of the neighboring block as a merge candidate. These redundancy checks can be divided into two categories for two different purposes: a. avoid having candidates with redundant motion data in the list; b. prevent merging two partitions that could be expressed by other means which would create redundant syntax.

[00166] When N is the number of spatial merge candidates, a complete redundancy check would consist of $\frac{N \cdot (N-1)}{2}$ motion data comparisons. In case of the five potential spatial merge candidates, ten motion data comparisons would be needed to assure that all candidates in the merge list have different motion data. During the development of HEVC, the checks for redundant motion data have been reduced to a subset in a way that the coding efficiency is kept while the comparison logic is significantly reduced. In the final design, no more than two comparisons are performed per candidate resulting in five overall comparisons. Given the order

of{A1, B1, B0, A0, B2}, B0 only checks B1, A0 only A1 and B2 only A1 and B1. In an embodiment of the partitioning redundancy check, the bottom PU of a 2N×N partitioning is merged with the top one by choosing candidate B1. This would result in one CU with two PUs having the same motion data which could be equally signaled as a 2N×2N CU. Overall, this check applies for all second PUs of the rectangular and asymmetric partitions 2N×N,2N×nU, 2N×nD,N×2N, nR×2N and nL×2N. It is noted that for the spatial merge candidates, only the redundancy checks are performed and the motion data is copied from the candidate blocks as it is. Hence, no motion vector scaling is needed here.

[00167] The derivation of the motion vectors for the temporal merge candidate is the same as for the TMVP. Since a merge candidate comprises all motion data and the TMVP is only one motion vector, the derivation of the whole motion data only depends on the slice type. For bi-predictive slices, a TMVP is derived for each reference picture list. Depending on the availability of the TMVP for each list, the prediction type is set to bi-prediction or to the list for which the TMVP is available. All associated reference picture indices are set equal to zero. Consequently, for uni-predictive slices, only the TMVP for list 0 is derived together with the reference picture index equal to zero.

[00168] When at least one TMVP is available and the temporal merge candidate is added to the list, no redundancy check is performed. This makes the merge list construction independent of the co-located picture which improves error resilience. Consider the case where the temporal merge candidate would be redundant and therefore not included in the merge candidate list. In the event of a lost co-located picture, the decoder could not derive the temporal candidates and hence not check whether it would be redundant. The indexing of all subsequent candidates would be affected by this.

[00169] For parsing robustness reasons, the length of the merge candidate list is fixed. After the spatial and the temporal merge candidates have been added, it can happen that the list has not yet the fixed length. In order to compensate for the coding efficiency loss that comes along with the non-length adaptive list index signaling, additional candidates are generated. Depending on the slice type, up to

two kind of candidates are used to fully populate the list: a. Combined bi-predictive candidates; b. Zero motion vector candidates.

[00170] In bi-predictive slices, additional candidates can be generated based on the existing ones by combining reference picture list 0 motion data of one candidate with and the list 1 motion data of another one. This is done by copying Δx_0, Δy_0, Δt_0 from one candidate, e.g. the first one, and Δx_1, Δy_1, Δt_1 from another, e.g. the second one. The different combinations are predefined and given in Table 1.1.

Table 1.1

Combination Order	0	1	2	3	4	5	6	7	8	9	10	11
$\Delta x_0, \Delta y_0, \Delta t_0$ from Cand.	0	1	0	2	1	2	0	3	1	3	2	3
$\Delta x_1, \Delta y_1, \Delta t_1$ from Cand.	1	0	2	0	2	1	3	0	3	1	3	2

[00171] When the list is still not full after adding the combined bi-predictive candidates, or for uni-predictive slices, zero motion vector candidates are calculated to complete the list. All zero motion vector candidates have one zero displacement motion vector for uni-predictive slices and two for bi-predictive slices. The reference indices are set equal to zero and are incremented by one for each additional candidate until the maximum number of reference indices is reached. If that is the case and there are still additional candidates missing, a reference index equal to zero is used to create these. For all the additional candidates, no redundancy checks are performed as it turned out that omitting these checks will not introduce a coding efficiency loss.

[00172] For each PU coded in inter-picture prediction mode, a so called merge_flag indicates that block merging is used to derive the motion data. The merge_idx further determines the candidate in the merge list that provides all the motion data needed for the MCP. Besides this PU-level signaling, the number of candidates in the merge list is signaled in the slice header. Since the default value is five, it is represented as a difference to five (five_minus_max_num_merge_cand). That way, the five is signaled with a short codeword for the 0 whereas using only one candidate, is signaled with a longer codeword for the 4. Regarding the impact on the merge candidate list construction process, the overall process remains the same although it terminates after the list contains the maximum number of merge

candidates. In the initial design, the maximum value for the merge index coding was given by the number of available spatial and temporal candidates in the list. When e.g. only two candidates are available, the index can be efficiently coded as a flag. But, in order to parse the merge index, the whole merge candidate list has to be constructed to know the actual number of candidates. Assuming unavailable neighboring blocks due to transmission errors, it would not be possible to parse the merge index anymore.

[00173] A crucial application of the block merging concept in HEVC is its combination with a skip mode. In previous video coding standards, the skip mode was used to indicate fora block that the motion data is inferred instead of explicitly signaled and that the prediction residual is zero, i.e. no transform coefficients are transmitted. In HEVC, at the beginning of each CU in an inter-picture prediction slice, a skip_flag is signaled that implies the following: a. the CU only contains one PU (2N×2N partition type); b. the merge mode is used to derive the motion data (merge_flag equal to 1); c. no residual data is present in the bitstream.

[00174] A parallel merge estimation level was introduced in HEVC that indicates the region in which merge candidate lists can be independently derived by checking whether a candidate block is located in that merge estimation region (MER). A candidate block that is in the same MER is not included in the merge candidate list. Hence, its motion data does not need to be available at the time of the list construction. When this level is e.g. 32, all prediction units in a 32×32 area can construct the merge candidate list in parallel since all merge candidates that are in the same 32×32 MER, are not inserted in the list. As shown in Fig.10, there is a CTU partitioning with seven CUs and ten PUs. All potential merge candidates for the first PU0 are available because they are outside the first 32×32MER. For the second MER, merge candidate lists of PUs 2-6 cannot include motion data from these PUs when the merge estimation inside that MER should be independent. Therefore, when looking at a PU5 for example, no merge candidates are available and hence not inserted in the merge candidate list. In that case, the merge list of PU5 consists only of the temporal candidate (if available) and zero MV candidates. In order to enable an encoder to trade-off parallelism and coding efficiency, the

parallel merge estimation level is adaptive and signaled aslog2_parallel_merge_level_minus2 in the picture parameter set.

[00175] Another one of the inter prediction modes is Sub-CU based motion vector prediction. During the development of the new video coding technique, with QTBT, each CU can have at most one set of motion parameters for each prediction direction. Two sub-CU level motion vector prediction methods are considered in the encoder by splitting a large CU into sub-CUs and deriving motion information for all the sub-CUs of the large CU. Alternative temporal motion vector prediction (ATMVP) method allows each CU to fetch multiple sets of motion information from multiple blocks smaller than the current CU in the collocated reference picture. In spatial-temporal motion vector prediction (STMVP) method motion vectors of the sub-CUs are derived recursively by using the temporal motion vector predictor and spatial neighboring motion vector.

[00176] To preserve more accurate motion field for sub-CU motion prediction, the motion compression for the reference frames is currently disabled.

[00177] While the Sub-CU based motion vector prediction includes Alternative temporal motion vector prediction, Spatial-temporal motion vector prediction, Combined with Merge mode, and Pattern matched motion vector derivation. Which will be introduced in the following:

[00178] In the alternative temporal motion vector prediction (ATMVP) method, the motion vectors temporal motion vector prediction (TMVP) is modified by fetching multiple sets of motion information (including motion vectors and reference indices) from blocks smaller than the current CU. As shown in Fig. 11, the sub-CUs are square N×N blocks (N is set to 4 by default).

[00179] ATMVP predicts the motion vectors of the sub-CUs within a CU in two steps. The first step is to identify the corresponding block in a reference picture with a so-called temporal vector. The reference picture is called the motion source picture. The second step is to split the current CU into sub-CUs and obtain the motion vectors as well as the reference indices of each sub-CU from the block corresponding to each sub-CU, as shown in Figure 6.

[00180] In the first step, a reference picture and the corresponding block is determined by the motion information of the spatial neighboring blocks of the current CU. To avoid the repetitive scanning process of neighboring blocks, the first merge candidate in the merge candidate list of the current CU is used. The first available motion vector as well as its associated reference index are set to be the temporal vector and the index to the motion source picture. This way, in ATMVP, the corresponding block may be more accurately identified, compared with TMVP, wherein the corresponding block (sometimes called collocated block) is always in a bottom-right or center position relative to the current CU.

[00181] In the second step, a corresponding block of the sub-CU is identified by the temporal vector in the motion source picture, by adding the temporal vector to the coordinate of the current CU. For each sub-CU, the motion information of its corresponding block (the smallest motion grid that covers the center sample) is used to derive the motion information for the sub-CU. After the motion information of a corresponding N×N block is identified, it is converted to the motion vectors and reference indices of the current sub-CU, in the same way as TMVP of HEVC, wherein motion scaling and other procedures apply. For example, the decoder checks whether the low-delay condition (i.e. the POCs of all reference pictures of the current picture are smaller than the POC of the current picture) is fulfilled and possibly uses motion vector MVx (the motion vector corresponding to reference picture list X) to predict motion vector MVy (with X being equal to 0 or 1 and Y being equal to 1−X) for each sub-CU.

[00182] In the Spatial-temporal motion vector prediction method, the motion vectors of the sub-CUs are derived recursively, following raster scan order. As shown in Fig. 12, it is considered that an 8×8 CU which contains four 4×4 sub-CUs A, B, C, and D. The neighboring 4×4 blocks in the current frame are labelled as a, b, c, and d.

[00183] The motion derivation for sub-CU A starts by identifying its two spatial neighbors. The first neighbor is the N×N block above sub-CU A (block c). If this

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

[00184] In the Combined with Merge mode, the sub-CU modes are enabled as additional merge candidates and there is no additional syntax element required to signal the modes. Two additional merge candidates are added to merge candidates list of each CU to represent the ATMVP mode and STMVP mode. Up to seven merge candidates are used, if the sequence parameter set indicates that ATMVP and STMVP are enabled. The encoding logic of the additional merge candidates is the same as for the merge candidates in HM, which means, for each CU in P or B slice, two more RD checks is needed for the two additional merge candidates.

[00185] Pattern matched motion vector derivation (PMMVD) mode is based on Frame-Rate Up Conversion (FRUC) techniques. With this mode, motion information of a block is not signaled but derived at decoder side.

[00186] A FRUC flag is signaled for a CU when its merge flag is true. When the FRUC flag is false, a merge index is signaled and the regular merge mode is used. When the FRUC flag is true, an additional FRUC mode flag is signaled to indicate which method (bilateral matching or template matching) is to be used to derive motion information for the block.

[00187] At encoder side, the decision on whether using FRUC merge mode for a CU is based on RD cost selection as done for normal merge candidate. That is the two matching modes (bilateral matching and template matching) are both checked

for a CU by using RD cost selection. The one leading to the minimal cost is further compared to other CU modes. If a FRUC matching mode is the most efficient one, FRUC flag is set to true for the CU and the related matching mode is used.

[00188] Motion derivation process in FRUC merge mode has two steps. A CU-level motion search is first performed, then followed by a Sub-CU level motion refinement. At CU level, an initial motion vector is derived for the whole CU based on bilateral matching or template matching. First, a list of MV candidates is generated and the candidate which leads to the minimum matching cost is selected as the starting point for further CU level refinement. Then a local search based on bilateral matching or template matching around the starting point is performed and the MV results in the minimum matching cost is taken as the MV for the whole CU. Subsequently, the motion information is further refined at sub-CU level with the derived CU motion vectors as the starting points.

[00189] For example, the following derivation process is performed for a $W \times H$ CU motion information derivation. At the first stage, MV for the whole $W \times H$ CU is derived. At the second stage, the CU is further split into $M \times M$ sub-CUs. The value of M is calculated as in Eq. (1.8), D is a predefined splitting depth which is set to 3 by default in the JEM. Then the MV for each sub-CU is derived.

$$M = \max\{4, \min\{\frac{W}{2^D}, \frac{H}{2^D}\}\} \qquad (1.8)$$

[00190] As shown in the Fig. 13, the bilateral matching is used to derive motion information of the current CU by finding the closest match between two blocks along the motion trajectory of the current CU in two different reference pictures. Under the assumption of continuous motion trajectory, the motion vectors MV0 and MV1 pointing to the two reference blocks shall be proportional to the temporal distances, i.e., TD0 and TD1, between the current picture and the two reference pictures. When the current picture is temporally between the two reference pictures and the temporal distance from the current picture to the two reference pictures is the same, the bilateral matching becomes mirror based bi-directional MV.

[00191] In the bilateral matching merge mode, bi-prediction is always applied since the motion information of a CU is derived based on the closest match between

two blocks along the motion trajectory of the current CU in two different reference pictures. There is no such limitation for the template matching merge mode. 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 costBi <= factor * min (cost0, cost1)

bi-prediction is used;

Otherwise, if cost0 <= 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 1.25, which means that the selection process is biased toward bi-prediction. The inter prediction direction selection is only applied to the CU-level template matching process.

[00192] As shown in. Fig.14, template matching is used to derive motion information of the current CU by finding the closest match between a template (top and/or left neighbouring blocks of the current CU) in the current picture and a block (same size to the template) in a reference picture. Except the aforementioned FRUC merge mode, the template matching is also applied to AMVP mode. With template matching method, a new candidate is derived. If the newly derived candidate by template matching is different to the first existing AMVP candidate, it is inserted at the very beginning of the AMVP candidate list and then the list size is set to two (meaning remove the second existing AMVP candidate). When applied to AMVP mode, only CU level search is applied.

[00193] The MV candidate set at CU level consists of: a. Original AMVP candidates if the current CU is in AMVP mode; b. all merge candidates; c. several MVs in the interpolated MV field; d. top and left neighbouring motion vectors.

[00194] It is noted that the interpolated MV field mentioned above is generated before coding a picture for the whole picture based on unilateral ME. Then the motion field may be used later as CU level or sub-CU level MV candidates. First, the motion field of each reference pictures in both reference lists is traversed at 4×4 block level. For each 4×4 block, if the motion associated to the block passing through a 4×4 block in the current picture, as shown in Fig.15, and the block has not been assigned any interpolated motion, the motion of the reference block is scaled to the current picture according to the temporal distance TD0 and TD1 (the same way as that of MV scaling of TMVP in HEVC) and the scaled motion is assigned to the block in the current frame. If no scaled MV is assigned to a 4×4 block, the block's motion is marked as unavailable in the interpolated motion field.

[00195] When using bilateral matching, each valid MV of a merge candidate is used as an input to generate a MV pair with the assumption of bilateral matching. For example, one valid MV of a merge candidate is (MVa, refa) at reference list A. Then the reference picture refb of its paired bilateral MV is found in the other reference list B so that refa and refb are temporally at different sides of the current picture. If such a refb is not available in reference list B, refb is determined as a reference which is different from refa and its temporal distance to the current picture is the minimal one in list B. After refb is determined, MVb is derived by scaling MVa based on the temporal distance between the current picture and refa, refb.

[00196] Four MVs from the interpolated MV field are also added to the CU level candidate list. More specifically, the interpolated MVs at the position (0, 0), (W/2, 0), (0, H/2) and (W/2, H/2) of the current CU are added.

[00197] When FRUC is applied in AMVP mode, the original AMVP candidates are also added to CU level MV candidate set.

[00198] At the CU level, up to 15 MVs for AMVP CUs and up to 13 MVs for merge CUs are added to the candidate list.

[00199] The MV candidate set at sub-CU level consists of: a. an MV determined from a CU-level search; b. top, left, top-left and top-right neighbouring MVs; c. scaled versions of collocated MVs from reference pictures; d. up to 4 ATMVP candidates; e. up to 4 STMVP candidates.

[00200] The scaled MVs from reference pictures are derived as follows. All the reference pictures in both lists are traversed. The MVs at a collocated position of the sub-CU in a reference picture are scaled to the reference of the starting CU-level MV.

[00201] ATMVP and STMVP candidates are limited to the four first ones.

[00202] At the sub-CU level, up to 17 MVs are added to the candidate list.

[00203] Motion vector can be refined by different methods combining with the different inter prediction modes.

[00204] MV refinement is a pattern based MV search with the criterion of bilateral matching cost or template matching cost. In the current development, two search patterns are supported – an unrestricted center-biased diamond search (UCBDS) and an adaptive cross search for MV refinement at the CU level and sub-CU level, respectively. For both CU and sub-CU level MV refinement, the MV is directly searched at quarter luma sample MV accuracy, and this is followed by one-eighth luma sample MV refinement. The search range of MV refinement for the CU and sub-CU step are set equal to 8 luma samples.

[00205] In bi-prediction operation, for the prediction of one block region, two prediction blocks, formed using a MV of list0 and a MV of list1, respectively, are combined to form a single prediction signal. In the decoder-side motion vector refinement (DMVR) method, the two motion vectors of the bi-prediction are further refined by a bilateral template matching process. The bilateral template matching applied in the decoder to perform a distortion-based search between a bilateral template and the reconstruction samples in the reference pictures in order to obtain a refined MV without transmission of additional motion information.

[00206] In DMVR, a bilateral template is generated as the weighted combination (i.e. average) of the two prediction blocks, from the initial MV0 of list0 and MV1 of list1, respectively, as shown in Fig. 16. The template matching operation consists

of calculating cost measures between the generated template and the sample region (around the initial prediction block) in the reference picture. For each of the two reference pictures, the MV that yields the minimum template cost is considered as the updated MV of that list to replace the original one. In the current development, nine MV candidates are searched for each list. The nine MV candidates include the original MV and 8 surrounding MVs with one luma sample offset to the original MV in either the horizontal or vertical direction, or both. Finally, the two new MVs, i.e., MV0′ and MV1′ as shown in Fig.16, are used for generating the final bi-prediction results. A sum of absolute differences (SAD) is used as the cost measure.

[00207] DMVR is applied for the merge mode of bi-prediction with one MV from a reference picture in the past and another from a reference picture in the future, without the transmission of additional syntax elements.

[00208] The usage of the TMVP, in AMVP as well as in the merge mode, requires the storage of the motion data (including motion vectors, reference indices and coding modes) in co-located reference pictures. Considering the granularity of motion representation, the memory size needed for storing motion data could be significant. HEVC employs motion data storage reduction (MDSR) to reduce the size of the motion data buffer and the associated memory access bandwidth by sub-sampling motion data in the reference pictures. While H.264/AVC is storing these information on a 4×4 block basis, HEVC uses a 16×16 block where, in case of sub-sampling a 4×4 grid, the information of the top-left 4×4 block is stored. Due to this sub-sampling, MDSR impacts on the quality of the temporal prediction.

[00209] Furthermore, there is a tight correlation between the position of the MV used in the co-located picture, and the position of the MV stored by MDSR. During the standardization process of HEVC, it turned out that storing the motion data of the top left block inside the 16×16 area together with the bottom right and center TMVP candidates provide the best tradeoff between coding efficiency and memory bandwidth reduction.

[00210] In HEVC, motion vector accuracy is one-quarter pel (one-quarter luma sample and one-eighth chroma sample for 4:2:0 video). In the current development, the accuracy for the internal motion vector storage and the merge candidate

increases to 1/16 pel. The higher motion vector accuracy (1/16 pel) is used in motion compensation inter prediction for the CU coded with skip/merge mode. For the CU coded with normal AMVP mode, either the integer-pel or quarter-pel motion is used.

[00211] When a motion vector points to a fractional sample position, motion compensated interpolation is needed. For the luma interpolation filtering, an 8-tap separable DCT-based interpolation filter is used for 2/4 precision samples and a 7-tap separable DCT-based interpolation filter is used for 1/4 precisions samples, as shown in Table 1.2.

Table 1.2

Position	Filter coefficients
1/4	{ -1, 4, -10, 58, 17, -5, 1 }
2/4	{ -1, 4, -11, 40, 40, -11, 4, -1 }
3/4	{ 1, -5, 17, 58, -10, 4, -1 }

[00212] Similarly, a 4-tap separable DCT-based interpolation filter is used for the chroma interpolation filter, as shown in Table 1.3.

Table 1.3

Position	Filter coefficients
1/8	{ -2, 58, 10, -2 }
2/8	{ -4, 54, 16, -2 }
3/8	{ -6, 46, 28, -4 }
4/8	{ -4, 36, 36, -4 }
5/8	{ -4, 28, 46, -6 }
6/8	{ -2, 16, 54, -4 }
7/8	{ -2, 10, 58, -2 }

[00213] For the vertical interpolation for 4:2:2 and the horizontal and vertical interpolation for 4:4:4 chroma channels, the odd positions in Table 1.3 are not used, resulting in $1/4^{th}$ chroma interpolation.

[00214] For the bi-directional prediction, the bit-depth of the output of the interpolation filter is maintained to 14-bit accuracy, regardless of the source bit-depth, before the averaging of the two prediction signals. The actual averaging process is done implicitly with the bit-depth reduction process as:

$$predSamples[x, y] = (predSamplesL0[x, y] + predSamplesL1[x, y] + offset) >> shift \quad (1.9)$$

$$shift = 15 - BitDepth \quad (1.10)$$

$$offset = 1 << (shift - 1) \quad (1.11)$$

[00215] To reduce complexity, bi-linear interpolation instead of regular 8-tap HEVC interpolation is used for both bilateral matching and template matching.

[00216] The calculation of matching cost is a bit different at different steps. When selecting the candidate from the candidate set at the CU level, the matching cost is the SAD of bilateral matching or template matching. After the starting MV is determined, the matching cost C of bilateral matching at sub-CU level search is calculated as follows:

$$C = SAD + w \cdot (|MV_x - MV_x^s| + |MV_y - MV_y^s|) \quad (1.12)$$

where w is a weighting factor which is empirically set to 4, MV and MV^s indicate the current MV and the starting MV, respectively. SAD is still used as the matching cost of template matching at sub-CU level search.

[00217] In FRUC mode, MV is derived by using luma samples only. The derived motion will be used for both luma and chroma for MC inter prediction. After MV is decided, final MC is performed using 8-taps interpolation filter for luma and 4-taps interpolation filter for chroma.

[00218] Whenever an explicit merge-mode index is signaled, the decoder side motion vector refinement starts from the motion vector(s) and reference indices normatively deduced from the signaled index. When an explicit merge-mode index is not signaled, but a flag indicating the use of the decoder side implicit derivation procedure is signaled for each coding unit, a set of initial motion vector candidates are evaluated at the decoder using a cost function (e.g. sum of absolute differences, mean removed sum of absolute differences, sum of squared differences, etc.) and the candidate with the lowest cost is chosen as the starting point for refinement. Thus, irrespective of whether the decoder-side motion vector derivation method is based on a cost function evaluated between predicted/reconstructed neighbor block boundary samples in the current and a similar shaped patch in one or more references (commonly called as template matching cost or TM_cost) (or) based on bilateral matching through difference minimization between the corresponding patches in reference L0 and reference L1 (commonly called as bilateral matching cost or BM_cost) (or) based on the difference between a bilaterally averaged version of the corresponding patches in reference L0 and reference L0 and a displacement in L0/L1

(called as DMVR_template_matching_cost), there is a refinement search that needs to be performed around starting points that can be sub-pixel accurate motion vectors.

[00219] In order to evaluate the cost function, an interpolation needs to be performed to derive the values at the sub-pixel accurate centers based on the values of the reference frames at the integer grid positions. The interpolating filter can be as simple as a bilinear interpolation filter or can be a longer filter such as the 2-D DCT-based separable interpolation filters. In order to reduce the complexity of deriving the interpolated samples for a block again and again at each position considered during the refinement, an integer-pixel distance grid of refinement points centered at the sub-pixel accurate position(s) in L0 and/or L1 had been proposed in another invention. With this, only incremental interpolations need to be performed as a new position close to a current best cost position is considered. Multiple search patterns such as 3x3, diamond-shaped 4-points around a center, cross-shaped 4-points around a center, and hexagonal 6-points around a center, have been proposed in the literature for typical motion estimation. However, from a decoder-side internal memory footprint standpoint, it is preferable for all decoder side motion vector refinement/derivation process and the normative motion compensated prediction process for all coding units of a coded tree block to be completed within one pipeline stage. As the processing spans more than one pipeline stage, the internal memory needs start to multiply linearly with the number of pipeline stages for which a particular data needs to be maintained. Given this memory constraint and the resulting processing constraint that all refinement iterations and final normative motion compensated interpolation need to happen in a single pipeline slot, the timing requirements for the refinements become constrained in a hardware design. Different prior art techniques have tried to reduce the average number of positions at which refinement is evaluated. While such techniques help to achieve lower average running time in software implementation or power savings in clock-gated hardware implementations, they can often result in more stages of dependent processing where each stage gets lesser time to finish and higher idle time for a chosen number of cost function evaluation hardware units. Techniques that try to reduce the number of dependent stages tend to have more unconditional processing with increased

parallelism that can result in a higher gate count due to the need for performing more work (both incremental interpolation and cost function evaluation) in a given time period. Hence, it is important to have a good trade-off between parallelism and number of dependent stages that result in optimal gate count and good utilization of the designed execution units.

[00220] A 3x3 patterned search that evaluates the cost function at 9 positions for first iteration followed by 5 additional positions per iteration (if the best cost is found at the diagonal positions in the current iteration) or 3 additional positions (if the best cost is found at vertically or horizontally only shifted positions) is shown below in Figure-17 (a)-(c). Whenever the center of the 3x3 positions evaluated in an iteration is the best cost position or when the maximum number of iterations (N_ITERS) allowed normatively is reached, the refinement ends. Given the dependence of the subsequent iteration on the previous iteration's results, the 9 cost evaluations for the first iteration need to be done within the total time allowed for refinement divided by N_ITERS. In certain hardware implementations, this can result in 9 different cost evaluation engines to be made available. In subsequent iterations, only 5 or 3 of these engines will be utilized.

[00221] Figure-18 shows an alternative method that attempts to reduce the average number of SAD evaluations at the expense of increased number of dependent stages. It first evaluates the cost function at 5 positions in the first iteration (center, and integer distance positions marked a, b, c, d). Using these costs, it arrives at one diagonal position out of the four diagonal positions at which to perform the next cost evaluation (illustratively marked as e). In every subsequent iteration, it evaluates the cost function at 2 additional points (illustratively marked as g and h). Based on the costs of e, g, and h, a diagonal position (illustratively marked as i) is determined at which to perform the cost evaluation. These 2-position + 1-position evaluation of cost function repeats for every subsequent iteration until either the center is the minimum of all 8 surrounding points or till N_ITERS is reached. For a given N_ITERS, this method requires 2*N_ITERS stages that have a sequential dependency. Thus each of these stages get the total time allowed for refinement divided by (2*N_ITERS), which is the half the time available per stage in the 3x3 pattern based method.

However, the maximum concurrent execution engines required is 5 instead of 9. It also suffers slightly from the problem of estimating one out of the four diagonal positions from the 5 position costs which may introduce some degradation in the quality of the refinement.

[00222] There are other prior art methods that maintain the same diamond-shaped pattern during each iteration. These incur the evaluation of cost function at the 3 additional positions for each iteration. However, these methods cover a rhombus shaped refinement range with a given number of iterations and hence under-utilize the interpolation investment made for the horizontal and vertical displacements. To get a same distance coverage on the diagonal direction as the 3x3 method with N_ITERS, the number of iterations required will be sqrt(2)*N_ITERS. This also results in reduction of the time available per stage by a factor of sqrt(2).

[00223] Hence, there is a need for a refinement search method that keeps the time available per stage at the same level as for the 3x3 method while keeping the number of concurrent execution engines at a lower level than 9 (e.g. 5), while making minimum sacrifices on the quality of the refinement.

[00224] Hardware designs would prefer to evaluate all the refinement search positions simultaneously in order to achieve the best timing. However, the gate count can significantly increase as the number of cost evaluation units will be equal to the number of possible search positions for the normative refinement range defined by DMVR. Hence, there is a need to limit the possible search positions.

[00225] In order to not increase the external memory access when compared to the motion compensated interpolation filtering need of (refinement_block_width + 7)*(refinement_block_height +7) for luma and (refinement_block_width*chroma_horizontal_downsampling_ratio + 4)*(refinement_block_height*chroma_vertical_downsampling_ratio + 4) and to allow pre-fetch of the motion compensation samples independent of refinement, many DMVR methods use the boundary sample within the above limits for motion compensation even if refinement produces non-zero delta motion vector. Given this, the practical refinement ranges get constrained to roughly +/-2 beyond which the effect of MC with padded pixels becomes objectionable.

[00226] Software implementations prefer to not have to evaluate all the positions normatively in order to obtain an average decoding (or encoding) time reduction. Hence, it is preferable to have some conditional processing that saves average number of operations (or power in hardware designs) without having a significant impact on the compression gains of DMVR.

[00227] The invention provides a refinement search method that trades off a small drop in refinement accuracy when compared with the 3x3 pattern search to achieve the goal of requiring only a maximum of 5 concurrent execution engines for cost evaluation, while keeping the time available per stage at the same level as the 3x3 pattern based refinement method. The method starts with evaluation of cost function at 5 positions and limits the maximum positions evaluated at each subsequent iteration also to 5 positions in such a way that all eight 1-pixel distance neighbor positions around the current iteration's center are evaluated across the previous iterations and the current iteration.

[00228] In another embodiment, the total refinement range considered is a rhombus shaped range. Typical maximum horizontal and vertical refinement ranges

used fall in the S = 1, 2, or 3 range (on the plus and minus sides from the center). When S is the same in the horizontal and vertical direction, the number of positions to evaluate is given by 2*S*(S+1) + 1. Hence, compared to the square shaped refinement range requiring (2*S + 1)*(2*S + 1) positions, the rhombus shaped range has half the number of possible positions with square shaped range plus one. Thus, the hardware requirement nearly halves when trying to achieve the best timing by evaluating for all possible positions concurrently.

[00229] An average operations reduction or average power saving is further achieved by normatively not requiring all 2*S*(S+1)+1 positions to be evaluated. S defines the number of iterations. In any iteration, if the center position is the minimum, the refinement stops. If the center position is the best among these 5, the refinement stops. In the first iteration, a set of 5 positions are evaluated corresponding to the center and its horizontal-only and vertical-only 1-pixel distance neighbors. In each additional iteration, up to 5 additional positions are evaluated centered on the position with the best cost from all iterations so far. The additional positions are chosen such that all the horizontal-only and vertical-only 1-pixel distance neighbors of the best cost position from the previous iteration are covered in either the earlier iterations or in the current iteration. In addition, up to 2 diagonal 1-pixel neighbor positions of the newly added horizontal-only and vertical-only 1-pixel neighbors that are at a distance of iteration count of pixel distance from the center of refinement are added. In addition, it is possible to normatively skip additional cost evaluations by seeing whether the cost at the center of refinement is less than a pre-determined bit-depth dependent threshold. Thus, the number of cost evaluations required can range from 1 to 5*S to give an average reduction in the number of operations compared to always doing 2*S*(S+1)+1 cost evaluations. If a hardware implementation chooses to have S number of stages, the number of cost evaluations in each stage is kept to not exceed 5.

[00230] In some embodiments, the cost evaluation is performed only on alternate rows of the block in order to reduce the operation count further.

[00231] In the rhombus shaped refinement range, when the cost evaluations are performed with accumulation only on alternate rows of the blocks, the worst case

number of row level cost evaluation units becomes S*(S+1) units instead of 2*S*(S+1) units. For example, when S=2, either (5+1+1)=7 row level cost accumulations are needed for the even vertical offsets or (3+3)=6 row level cost accumulations are needed for the odd vertical offsets.

[00232] Detailed description of Embodiments of the invention

[00233] Given that decoder side motion vector refinement/derivation is a normative aspect of a coding system, the encoder will also have to perform the same refinement search operation in order to not have any drift between the encoder's reconstruction and the decoder's reconstruction. Hence, all aspects of all embodiments are applicable to both encoding and decoding systems.

[00234] In template matching, the refinement movement happens only in the reference starting from the sub-pixel accurate center that is derived based on the explicitly signaled merge index or implicitly through cost evaluations.

[00235] In bilateral matching (with or without averaged template), the refinements start in the L0 and L1 references starting from the respective sub-pixel accurate centers that are derived based on the explicitly signaled merge index or implicitly through cost evaluations.

[00236] It should be noted that when a bilateral matching cost is evaluated based on equal and opposite horizontal and vertical displacements in L1 for a given horizontal and vertical displacement in L0, the positions shown in the figures are assumed to correspond to L0 and the positions in L1 are derived by negating the displacement in the horizontal and vertical directions with respect to the current iteration center in L1.

[00237] Figure-19 illustrates the first embodiment. Figure-19(a) shows the 5 positions at which the cost function is evaluated in the first iteration. These are the center and its left, top, right, and bottom 1-pixel distance displacements.

[00238] If the maximum number of allowed iterations is not 1, the second iteration is centered on the position with the lowest cost out of the 5 evaluated positions in the first iteration. Of the eight 1-pixel distance (in horizontal and/or vertical) neighbors of this center, the positions that were not evaluated in the first iteration are identified and cost function evaluations are performed for these positions. In the second

iteration, the number of such additional positions can be 4 (if the center of the first iteration is the center for the second iteration also) or 5 (if the center of the first iteration is not the center of the second iteration). The position with the lowest cost out of the set of 9 positions (center and its eight 1-pixel distance neighbors) is determined. If the center is the position with the lowest cost, then the refinement process ends. If the maximum number of allowed iterations is 1, then the refinement process ends.

[00239] If the maximum number of allowed iterations is not yet reached, the subsequent iterations are performed centered on the position with the lowest cost from the previous iteration. Of the eight 1-pixel distance (in horizontal and/or vertical) neighbors of this center position, the positions that were not evaluated in the earlier iterations are identified and cost function evaluations are performed for these positions. The number of such additional positions will never exceed 5. The position with the lowest cost out of the set of 9 positions (center and its eight 1-pixel distance neighbors) is determined. If the center is the position with the lowest cost, then the refinement process ends. If the maximum number of allowed iterations is 1, then the refinement process ends.

[00240] Method embodiment 1

As can be seen, with the number of dependent stages being exactly same as the 3x3 pattern search, the time available for a stage is the same as in the 3x3 pattern search case. However, the maximum number of concurrent units has been restricted to 5. This is achieved by not unconditionally investing in 9 evaluations in the first iteration.

[00241] Method Embodiment 2

A minor modification to embodiment 1 is where the exit out of the loop happens when the center position has the lowest cost after the first iteration. In this case, the diagonal points will not be checked.

The benefit of this embodiment is a lower average number of computations in software implementations and lower power consumption in clock-gated hardware implementations. Not checking the diagonal points in the first iteration does not have a major BDRATE impact.

[00242] Embodiment 3

The method mentioned in embodiment 1 and embodiment 2 can be used in decoder side motion vector refinement based on bilateral matching.

Whenever an explicit merge-mode index is signaled, the decoder side motion vector refinement starts from the motion vector(s) and reference indices normatively deduced from the signaled index. When an explicit merge-mode index is not signaled, but a flag indicating the use of the decoder side implicit derivation procedure is signaled for each coding unit, a set of initial motion vector candidates are evaluated at the decoder using a cost function (e.g. sum of absolute differences, mean removed sum of absolute differences, sum of squared differences, etc.) and the candidate with the lowest cost is chosen as the starting point for refinement.

After get the starting point for the refinement, bilateral matching can be used to do the refinement. Like, based on bilateral matching through difference minimization between the corresponding patches in reference L0 and reference L1(commonly called as bilateral matching cost or BM_cost), or based on the difference between a bilaterally averaged version of the corresponding patches in reference L0 and reference L0 and a displacement in L0/L1(called as DMVR_template_matching_cost).

Here note that, for BM_cost case, the bilateral matching cost is evaluated based on equal and opposite horizontal and vertical displacements in L1 for a given horizontal and vertical displacement in L0. the positions are assumed to correspond to L0 and the positions in L1 are derived by negating the displacement in the horizontal and vertical directions with respect to the current iteration center in L1.

[00243] Embodiment 4

The method mentioned in embodiment 1 and embodiment 2 can be used in decoder side motion vector refinement based on template matching.

Whenever an explicit merge-mode index is signaled, the decoder side motion vector refinement starts from the motion vector(s) and reference indices normatively deduced from the signaled index. When an explicit merge-mode index is not signaled, but a flag indicating the use of the decoder side implicit derivation procedure is signaled for each coding unit, a set of initial motion vector candidates are evaluated at

the decoder using a cost function (e.g. sum of absolute differences, mean removed sum of absolute differences, sum of squared differences, etc.) and the candidate with the lowest cost is chosen as the starting point for refinement.

After get the starting point for the refinement, template matching can be used to do the refinement.

The decoder-side motion vector derivation method is based on a cost function evaluated between predicted/reconstructed neighbor block boundary samples in the current and a similar shaped patch in one or more references (commonly called as template matching cost or TM_cost).

[00244] Embodiment 5

In this embodiment, instead of a square search range, a rhombus shaped refinement search range is used. As described before, for this shape, if S is the maximum horizontal or vertical displacement, then the maximum number of positions to evaluate will be 2*S*(S+1) + 1. This is equal to half of the maximum number of positions that need to be evaluated in the square search range case for the same maximum horizontal or vertical displacement from the center of refinement plus 1.

From a normative perspective, the search is set up in stages/iterations. However, it should be noted that it does not preclude the hardware from evaluating all possible evaluation positions concurrently. In the first stage, the costs evaluated at 5 positions comprising the center of refinement and its 1-pixel distance horizontal-only and vertical-only neighbors are considered. If the center position cost is the least of these 5 costs, then the integer distance refinement stops. Else, the next stage considers the costs evaluated at up to 5 additional positions centered on the position with the least cost from the first 5 positions. The best cost position among all the positions evaluated across the first and second stages will be best integer distance refinement delta motion vector after two stages. In the most preferred embodiment, only 2 stages are considered. However, in a more generic scheme, additional stages can continue in a similar manner centered on the best cost position from the previous stages with the additional cost evaluations limited to not exceed 5.

These 5 additional positions are selected in the following manner:

- The horizontal-only and vertical-only 1-pixel distance neighbors of the best integer distance refinement position till now and positions that have not been already evaluated in an earlier iteration are first included. These can result in up to 3 positions.
- The diagonal 1-pixel distance neighbors of the neighbors chosen in the above step that fall within the current iteration count distance from the initial center of refinement are then included. These can result in only up to 2 additional positions over the positions selected in the above step.

Figure 21 shows an example for +/-3 rhombus refinement range with average multi-stage cost evaluation. Filled 5 circles are evaluated in the first stage. Assuming that (1,0) was the best cost delta displacement from the first stage, the second stage evaluates the 5 filled squares next. Assuming that (1,1) is the best cost position after stage-2, the third stage evaluates the 4 filled rhombuses. The unfilled circles indicate the positions within the search range at which cost

An example of this embodiment is shown in figure-21 for the case of S=3. The filled circle positions where the center position corresponds to the merge motion vectors of L0 and L1 are evaluated first. In the case of symmetric bilateral matching, the delta movement in L0 is flipped in sign in L1. In this example, the best position among the first 5 evaluations is determined to be corresponding to delta MV of (1,0) in L0. For the next set of evaluations, the filled square positions are selected according to the rules specified above for selecting the additional positions. The best cost position among all 10 evaluations is determined to be corresponding to delta MV of (1,1). For the preferred embodiment of S=2, the integer distance refinement will stop at this point. When S=3, the third stage of refinement evaluates cost at 4 additional positions corresponding to the filled rhombuses. The best position in this example is determined to be corresponding to delta MV of (1,2). Hence, compared to the worst-case of evaluating 25 positions, the method only evaluates 5+5+4=14 positions.

A flow chart describing the refinement process under embodiment 5 is illustrated in figure-22.

In addition, in a variant of the above embodiment, only alternate rows of the blocks are used for cost evaluation. With this, the total possible evaluation positions of 2*S*(S+1)+1 can be broken into two sets of size S*(S+1)+1 positions and S*(S+1) positions. This is because a given line in the block of (width + 2*S)*(height + 2*S) samples falls on the odd vertical position for the first set and even vertical position for the second set. Since alternate row based cost evaluation happens only for either

the odd rows or the even rows for all positions, for each new line, only S*(S+1)+1 concurrent cost evaluations are required. However, the number of cost accumulators will still be equal to 2*S(S+1)+1.

The technical benefit(s) / advantage(s) of Embodiment 5

Compared to a square search range, the number of maximum possible search positions is nearly halved. For the sweet spot case of S=2, the maximum possible positions is 13. This allows hardware designs to consider evaluating all costs concurrently to achieve the lowest latency or the best timing.

The normative staged constraints enable software implementations to achieve average operations reduction and hardware/software to derive average power saving benefits compared to always evaluating all possible search positions. At the same time, it does not preclude concurrent evaluation of all positions to achieve the best timing and then enforce these constraints normatively.

The alternate row based cost evaluation combined with the rhombus search allows just S*(S+1)+1 row-level cost evaluation units to be sufficient even when evaluating all possible positions concurrently in order to achieve the best timing.

[00245] An integer distance search method for decoder side motion vector refinement comprising:

- Evaluating a cost function at the refinement center and surrounding horizontal-only and vertical-only 1-pixel distance neighbor positions;
- Determining the lowest cost position among these evaluated positions;
- Performing subsequent iterations of refinement centered on the lowest cost position of the preceding iteration comprising the steps of:
    - Determining whether the prescribed maximum number of iterations has been exceeded and exiting the refinement loop when it has exceeded
    - Evaluating the cost function at additional positions within the eight 1-pixel distance neighbor positions at which the cost function has not been evaluated in any of the previous iterations

- Determining the lowest cost position among the center and these remaining positions at which the cost function has not been evaluated in any of the previous iterations;
- Determining whether the lowest cost position is the current iteration's center position and exiting the refinement loop if the lowest cost position is the current iteration's center position.

An integer distance search method variant of the above wherein a rhombus shaped refinement range is employed. An early exit check is applied at the center position to see whether the center position cost is lower than a pre-determined bit depth dependent threshold. Instead of evaluating additional positions to complete eight 1-pixel distance neighbors around a current iteration center, the additional positions are evaluated to complete the horizontal-only and vertical-only 1-pixel distance neighbors of the center and their diagonal 1-pixel neighbors that fall within the current iteration count distance from the initial refinement center. In addition, in some embodiments, the cost evaluation is done using only alternate lines of pixels within each search position block.

[00246] The invention relates to versatile video coding standardization which was earlier pursued as a Joint Exploratory Model (JEM) within Joint Video Exploration Team which is a joint work between Q16 of VCEG and MPEG (SC29/WG11). Document JVET-G1001 and other Huawei prior art relating to decoder side motion vector refinement and decoder side motion vector derivation can be used to get a list of contribution documents and patents related to this invention.

[00247] All embodiments of the present invention are applicable to both DMVD and DMVR methods.

[00248] In one or more examples, the functions described may be implemented in hardware, software, firmware, or any combination thereof. If implemented in software, the functions may be stored on or transmitted over as one or more instructions or code on a computer-readable medium and executed by a hardware-based processing unit. Computer-readable media may include computer-readable storage media, which corresponds to a tangible medium such as data storage media,

or communication media including any medium that facilitates transfer of a computer program from one place to another, e.g., according to a communication protocol. In this manner, computer-readable media generally may correspond to (1) tangible computer-readable storage media which is non-transitory or (2) a communication medium such as a signal or carrier wave. Data storage media may be any available media that can be accessed by one or more computers or one or more processors to retrieve instructions, code and/or data structures for implementation of the techniques described in this disclosure. A computer program product may include a computer-readable medium.

[00249] By way of example, and not limitation, such computer-readable storage media can comprise RAM, ROM, EEPROM, CD-ROM or other optical disk storage, magnetic disk storage, or other magnetic storage devices, flash memory, or any other medium that can be used to store desired program code in the form of instructions or data structures and that can be accessed by a computer. Also, any connection is properly termed a computer-readable medium. For example, if instructions are transmitted from a website, server, or other remote source using a coaxial cable, fiber optic cable, twisted pair, digital subscriber line (DSL), or wireless technologies such as infrared, radio, and microwave, then the coaxial cable, fiber optic cable, twisted pair, DSL, or wireless technologies such as infrared, radio, and microwave are included in the definition of medium. It should be understood, however, that computer-readable storage media and data storage media do not include connections, carrier waves, signals, or other transitory media, but are instead directed to non-transitory, tangible storage media. Disk and disc, as used herein, includes compact disc (CD), laser disc, optical disc, digital versatile disc (DVD), floppy disk and Blu-ray disc, where disks usually reproduce data magnetically, while discs reproduce data optically with lasers. Combinations of the above should also be included within the scope of computer-readable media.

[00250] Instructions may be executed by one or more processors, such as one or more digital signal processors (DSPs), general purpose microprocessors, application specific integrated circuits (ASICs), field programmable logic arrays (FPGAs), or other equivalent integrated or discrete logic circuitry. Accordingly, the

term "processor," as used herein may refer to any of the foregoing structure or any other structure suitable for implementation of the techniques described herein. In addition, in some aspects, the functionality described herein may be provided within dedicated hardware and/or software modules configured for encoding and decoding, or incorporated in a combined codec. Also, the techniques could be fully implemented in one or more circuits or logic elements.

[00251] The techniques of this disclosure may be implemented in a wide variety of devices or apparatuses, including a wireless handset, an integrated circuit (IC) or a set of ICs (e.g., a chip set). Various components, modules, or units are described in this disclosure to emphasize functional aspects of devices configured to perform the disclosed techniques, but do not necessarily require realization by different hardware units. Rather, as described above, various units may be combined in a codec hardware unit or provided by a collection of interpretable hardware units, including one or more processors as described above, in conjunction with suitable software and/or firmware.

[00252] While several embodiments have been provided in the present disclosure, it should be understood that the disclosed systems and methods might be embodied in many other specific forms without departing from the spirit or scope of the present disclosure. The present examples are to be considered as illustrative and not restrictive, and the intention is not to be limited to the details given herein. For example, the various elements or components may be combined or integrated in another system or certain features may be omitted, or not implemented.

[00253] In addition, techniques, systems, subsystems, and methods described and illustrated in the various embodiments as discrete or separate may be combined or integrated with other systems, modules, techniques, or methods without departing from the scope of the present disclosure. Other items shown or discussed as coupled or directly coupled or communicating with each other may be indirectly coupled or communicating through some interface, device, or intermediate component whether electrically, mechanically, or otherwise. Other examples of changes, substitutions, and alterations are ascertainable by one skilled in the art and could be made without departing from the spirit and scope disclosed herein.

CLAIMS

1. An apparatus for determination of a motion vector for a prediction block including one or more processing circuitry configured to:

obtain an estimate of the motion vector;

perform a search within a first search space wherein searched sample positions are P additional sample positions adjacent to a position corresponding to the estimate of the motion vector in integer-pel distance, and the P additional sample positions are horizontal-only and vertical-only x-pixel distance neighbor positions and the P sample positions are pointed by P candidate motion vectors, wherein P is a positive integer and P is less than 5;

perform a search within a n-th search space, wherein searched sample positions are Q additional sample positions that are not previously searched, and the Q sample positions includes a first set of Q_3 sample positions and a second set of Q_2 sample positions, wherein the Q_3 sample positions are horizontal-only and vertical-only x-pixel distance neighbor positions that are adjacent to a position pointed by a (n-1)-th refined motion vector, wherein the Q_2 sample positions are diagonal x-pixel neighbor positions to the positions in the first set of sample positions, and the Q sample positions are pointed by Q candidate motion vectors, wherein Q, Q_2, Q_3 is a positive integer and Q is less than 5 or equal to 5, Q_2 is less than 2 or equal to 2, Q_3 is less than 3 or equal to 3; wherein the (n-1)-th refined motion vector is found in the (n-1)-th search space, wherein n is a positive integer which is larger than 1;

wherein the motion vector for the image prediction block is found in the n-th search space.

2. The apparatus according to claim 1, the processing circuitry is configured to:

determine the first search space consisting of (P+1) or P sample positions,

calculate matching cost for the respective candidate motion vector;

comparing matching cost of the estimate of the motion vector with the matching cost of the respective candidate motion vector, and

perform a search within the second search space if the calculated matching cost of one of the P candidate motion vectors is the lowest cost among the (P+1) or P sample positions.

3. The apparatus according to claim 1 or 2, within the first search space, the processing circuitry is further configured to:

calculate the matching cost for the respective candidate motion vector;

comparing the matching cost of the starting sample position with the matching cost of the respective candidate motion vector, and

determining the motion vector for the prediction block if the matching cost of the estimate of the motion vector is lower than the calculated matching cost of each of a plurality of the candidate motion vectors.

4. The apparatus according to claim 2 or 3, wherein the matching cost comprises bilateral matching cost or template matching cost.

5. The apparatus according to any one of the preceding of claims, wherein the processing circuitry is specifically configured to:

determining the first search space consisting of the P or (P+1) sample positions;

performing template matching in the first search space to obtain a best matching sample position,

determining the n-th search space based on the best matching sample position of the (n-1)-th iteration, wherein n=2,3,…,K, and K is a predefined threshold; and

performing template matching in the n-th search space to obtain a best matching position.

6. The apparatus according to any one of the preceding of claims, wherein if n is equal to a predefined threshold K, the motion vector for the prediction block is found in the n-th search space according to a cost function.

7. The apparatus according to claim 6, wherein the cost function is based on a predetermined template and indicates, for the respective candidate motion vector, a level of similarity between the predetermined template and a predictor pointed to by the respective candidate motion vector.

8. The apparatus according to claim 6 or 7, wherein in the n-th search space, a motion vector pointing to a sample position where the cost function is minimized is a part or whole of the motion vector for the prediction block.

9. The apparatus according to any one of the preceding of claims, wherein the first search space consists of P or (P+1) sample positions to which P or (P+1) candidate motion vectors point respectively, and the position pointed by the estimate of the motion vector is located substantially in the center of the first search space, P=4.

10. The apparatus according to any one of the preceding of claims, wherein the first search space consists of a central position pointed by the estimate of the motion vector and surrounding horizontal-only and vertical-only integer-pixel distance neighbor positions; or, wherein the first search space consists of horizontal-only and vertical-only sample positions surrounding a central position pointed by the estimate of the motion vector.

11. The apparatus according to any one of the preceding of claims, wherein the n-th search space consists of Q or (Q+1) sample positions to which Q or (Q+1) candidate motion vectors point respectively, and the position pointed by a (n-1)-

th refined motion vector is located substantially in the center of the n-th search space, 1<Q<=5.

12. The apparatus according to any one of the preceding of claims, wherein the first search space, the n-th search space have integer resolution, wherein n=2, 3,...K, K is a predefined threshold.

13. The apparatus according to any one of the preceding of claims, wherein the processing circuitry is specifically configured to determine the estimate of the motion vector from a list of motion vectors including motion vectors of at least one block adjacent to a current block.

14. The apparatus according to any one of the preceding of claims, wherein the first search space comprises a search space in a first reference picture and a search space in a second reference picture; and/or,

the n-th search space comprises a n_1-th search space in a first reference picture and a n_2-th search space in a second reference picture.

15. The apparatus according to any one of the preceding of claims, wherein the Q_2 positions are diagonal 1-pixel neighbor positions to the positions in the first set of positions and are within a current iteration count of distance from the position pointed by the estimate of the motion vector, wherein the current iteration count is equal to n.

16. The apparatus according to any one of the preceding of claims, wherein x-pixel distance= 1-pixel distance, or x=1.

17. An apparatus including one or more processing circuitry configured for:

- evaluating a cost function at the refinement center and surrounding neighbor positions;

- determining the lowest cost position among these evaluated positions;

- performing subsequent iterations of refinement centered on the lowest cost position of the preceding iteration comprising the steps of:
    - determining whether the maximum number of iterations has been exceeded and exiting the refinement loop when it has exceeded;
    - evaluating the cost function at Q additional positions, wherein the Q sample positions include a first set of Q_3 additional positions, wherein the Q_3 positions are the horizontal-only and vertical-only x-pixel distance neighbor positions which are of the current iteration center and at which the cost function has not been evaluated in any of the previous iterations, and the Q sample positions further include a second set of Q_2 additional positions that are diagonal x-pixel neighbor positions to the positions in the first set of additional positions ;
    - determining the lowest cost position among the center and these remaining positions at which the cost function has not been evaluated in any of the previous iterations;
- determining whether the lowest cost position is the current iteration's center position and exiting the refinement loop if the lowest cost position is the current iteration's center position;

wherein Q, Q_2, Q_3 is a positive integer and Q is less than or equal to 5, Q_2 is less than or equal to 2, and Q_3 is less than or equal to 3.

18. The apparatus according to any one of the preceding of claims, wherein the Q sample positions consist of the first set of Q_3 positions and the second set of Q_2 positions, and the second set of Q_2 additional positions are diagonal 1-pixel neighbor positions to the positions in the first set of positions and are within current iteration count of distance from the initial center of refinement, wherein the initial center of refinement is the position pointed by the estimate of the motion vector.

19. The apparatus according to any one of the preceding of claims, wherein x-pixel distance= 1-pixel distance, or x=1.

20. An encoding apparatus for encoding video images split to prediction blocks into a bitstream, the encoding apparatus comprising:

the apparatus for determination of a motion vector for a prediction block according to any of claims 1 to 19;

an encoding circuitry for encoding difference between the prediction block and the predictor given by a prediction block in a position based on the determined motion vector and for generating bitstream including the encoded difference and the estimate of the motion vector.

21. A decoding apparatus for decoding from a bitstream video images split to prediction blocks, the decoding apparatus comprising:

a parsing unit for parsing from the bitstream an estimate of the motion vector and an encoded difference between a prediction block and a predictor given by a prediction block in a position specified by a refined motion vector;

the apparatus for determination of the refined motion vector for the prediction block according to any of claims 1 to 19;

decoding circuitry for reconstructing the prediction block as a sum of the parsed difference and the predictor given by the prediction block in the position specified by the refined motion vector.

22. A method for determining a motion vector for an image prediction block, the method comprising:

obtaining an estimate of the motion vector;

performing a search within a first search space wherein searched sample positions are P additional sample positions adjacent to a position corresponding to the estimate of the motion vector in integer-pel distance, and the P additional

sample positions are horizontal-only and vertical-only x-pixel distance neighbor positions and the P sample positions are pointed by P candidate motion vectors, wherein P is a positive integer and P is less than 5;

performing a search within a n-th search space, wherein searched sample positions are Q additional sample positions that are not previously searched, and the Q sample positions includes a first set of Q_3 sample positions and a second set of Q_2 sample positions, wherein the Q_3 sample positions are horizontal-only and vertical-only x-pixel distance neighbor positions that are adjacent to a position pointed by a (n-1)-th refined motion vector, wherein the Q_2 sample positions are diagonal x-pixel neighbor positions to the positions in the first set of sample positions, and the Q sample positions are pointed by Q candidate motion vectors, wherein Q, Q_2, Q_3 is a positive integer and Q is less than or equal to 5, Q_2 is less than or equal to 2, Q_3 is less than or equal to 3; wherein the (n-1)-th refined motion vector is found in the (n-1)-th search space, wherein n is a positive integer which is larger than 1;

wherein the motion vector for the image prediction block is found in the n-th search space.

23. The method according to claim 22, wherein the obtaining the estimate of the motion vector, comprises: obtaining an initial motion vector and a template for the prediction block.

24. The method according to claim22, the performing a search within a first search space, comprises:

determining the first search space consisting of (P+1) or P sample positions,

calculate matching cost for the respective candidate motion vector;

comparing matching cost of the estimate of the motion vector with the matching cost of the respective candidate motion vector, and

performing a search within the second search space if the calculated matching cost of one of the P candidate motion vectors is the lowest cost among the (P+1) or P sample positions.

25. The method according to according to any one of the preceding of claims,, within the first search space, the performing a search within a first search space, further comprises:

calculate the matching cost for the respective candidate motion vector;

comparing the matching cost of the starting sample position with the matching cost of the respective candidate motion vector, and

determining the motion vector for the prediction block if the matching cost of the estimate of the motion vector is lower than the calculated matching cost of each of a plurality of the candidate motion vectors.

26. The method according to claim 24 or 25, wherein the matching cost comprises bilateral matching cost or template matching cost.

27. The method according to any one of the preceding of claims, wherein the performing a search within a first search space, comprises:

determining the first search space consisting of the P or (P+1) sample positions;

performing template matching in the first search space to obtain a best matching sample position,

the performing a search within a n-th search space, comprises:

determining the n-th search space based on the best matching sample position of the (n-1)-th iteration, wherein n=2,3,…,K, and K is a predefined threshold; and

performing template matching in the n-th search space to obtain a best matching position.

28. The method according to any one of the preceding of claims, wherein if n is equal to a predefined threshold K, the motion vector for the prediction block is found in the n-th search space according to a cost function.

29. The method according to claim 28, wherein the cost function is based on a predetermined template and indicates, for the respective candidate motion vector, a level of similarity between the predetermined template and a predictor pointed to by the respective candidate motion vector.

30. The method according to claim 28 or 29, wherein in the n-th search space, a motion vector pointing to a sample position where the cost function is minimized is a part or whole of the motion vector for the prediction block.

31. The method according to any one of the preceding of claims, wherein the first search space consists of P or (P+1) sample positions to which P or (P+1) candidate motion vectors point respectively, and the position pointed by the estimate of the motion vector is located substantially in the center of the first search space, P=4.

32. The method according to any one of the preceding of claims, wherein the first search space consists of a central position pointed by the estimate of the motion vector and surrounding horizontal-only and vertical-only integer-pixel distance neighbor positions; or, wherein the first search space consists of horizontal-only and vertical-only sample positions surrounding a central position pointed by the estimate of the motion vector.

33. The method according to any one of the preceding of claims, wherein the n-th search space consists of Q or (Q+1) sample positions to which Q or (Q+1) candidate motion vectors point respectively, and the position pointed by a (n-1)-th refined motion vector is located substantially in the center of the n-th search space, $1<Q<=5$.

34. The method according to any one of the preceding of claims, wherein the first search space, the n-th search space have integer resolution, wherein n=2, 3,…K, K is a predefined threshold.

35. The method according to any one of the preceding of claims, wherein the obtaining an estimate of the motion vector comprises: determining the estimate of the motion vector from a list of motion vectors including motion vectors of at least one block adjacent to a current block.

36. The method according to any one of the preceding of claims, wherein the first search space comprises a search space in a first reference picture and a search space in a second reference picture; and/or,

the n-th search space comprises a n_1-th search space in a first reference picture and a n_2-th search space in a second reference picture.

37. The method according to any one of the preceding of claims, wherein the method is used for Decoder Side Motion Vector Refinement (DMVR) or Decoder Side Motion Vector derivation (DMVD).

38. The method according to any one of the preceding of claims, wherein the Q_2 positions are diagonal 1-pixel neighbor positions to the positions in the first set of positions and are within a current iteration count of distance from the position corresponding to the estimate of the motion vector, wherein the current iteration count is equal to n.

39. The method according to any one of the preceding of claims, wherein x-pixel distance= 1-pixel distance, or x=1.

40. An integer distance search method for decoder side motion vector refinement comprising:

- evaluating a cost function at the refinement center and surrounding neighbor positions;
- determining the lowest cost position among these evaluated positions;
- performing subsequent iterations of refinement centered on the lowest cost position of the preceding iteration comprising the steps of:
  - determining whether the maximum number of iterations has been exceeded and exiting the refinement loop when it has exceeded
  - evaluating the cost function at Q additional positions, wherein the Q sample positions include a first set of Q_3 additional positions, wherein the Q_3 positions are the horizontal-only and vertical-only x-pixel distance neighbor positions which are of the current iteration center and at which the cost function has not been evaluated in any of the previous iterations, and the Q sample positions further include a second set of Q_2 additional positions that are diagonal x-pixel neighbor positions to the positions in the first set of additional positions ;
  - determining the lowest cost position among the center and these remaining positions at which the cost function has not been evaluated in any of the previous iterations;

determining whether the lowest cost position is the current iteration's center position and exiting the refinement loop if the lowest cost position is the current iteration's center position;

wherein Q, Q_2, Q_3 is a positive integer and Q is less than or equal to 5, Q_2 is less than or equal to 2, and Q_3 is less than or equal to 3.

41. The method according to any one of the preceding of claims, wherein the Q sample positions consist of the first set of Q_3 positions and the second set of Q_2 positions, and the second set of Q_2 additional positions are diagonal 1-pixel neighbor positions to the positions in the first set of positions and are within current iteration count of distance from the initial center of refinement, wherein the initial center of

refinement is the position pointed by the estimate of the motion vector.

42. The method according to any one of the preceding of claims, wherein x-pixel distance= 1-pixel distance, or x=1.

43. A method for determining a motion vector for an image prediction block, the method comprising:

Obtaining an estimate of the motion vector;

performing a search within a first search space, wherein the first search space is defined by a part of a plurality of candidate motion vectors pointing at the sample positions adjacent to the position pointed by the estimate of the motion vector in integer-pel distance;

performing a search within a n-th search space, wherein the n-th search space is defined by un-searched candidate motion vectors of a whole of a plurality of candidate motion vectors pointing at the sample positions adjacent to the position pointed by a previously-found motion vector in integer-pel distance, wherein the previously-found motion vector is found in the (n-1)-th search space, wherein n is a positive integer which is larger or equal to 2;

wherein the motion vector for the image prediction block is found in the n-th search space.

44. An encoding method for encoding video images split to prediction blocks into a bitstream, the encoding method comprising:

determining a motion vector for a prediction block according to any of claims 22 to 43;

encoding difference between the prediction block and the predictor given by a prediction block in a position based on the determined motion vector and for

generating bitstream including the encoded difference and the initial motion vector.

45. A decoding method for decoding from a bitstream video images split to prediction blocks, the decoding method comprising:

parsing from the bitstream an initial motion vector and an encoded difference between a prediction block and a predictor given by a prediction block in a position specified by a refined motion vector;

determining the refined motion vector for the prediction block according to any of claims 22 to 43;

reconstructing the prediction block as a sum of the parsed difference and the predictor given by the prediction block in the position based on the refined motion vector.

46. Computer readable medium storing instructions which when executed on a processor cause the processor to perform the method according to any of claims 22 to 43.

47. An apparatus for determination of a motion vector for a prediction block or coding unit including a processing circuitry configured to:

obtain an estimate of the motion vector;

perform a search within a first search space, wherein searched sample positions are P sample positions, adjacent to a position corresponding to the estimate of the motion vector in integer-pel distance, and the P sample positions correspond to P candidate motion vectors, wherein P is a positive integer and P is less than 8, preferably equal to 5 or less than 5;

perform a search within a n-th search space, wherein searched sample positions are Q sample positions that are adjacent to a position pointed by a (n-1)th refined motion vector in integer-pel distance and that are not previously searched, and the Q sample positions are pointed by Q candidate motion vectors, wherein Q is a positive integer and Q is less than 5 or equal to 5; wherein the (n-1)-th refined motion vector is found in the (n-1)-th search space, wherein n is a positive integer which is larger than 1;

wherein the motion vector for the image prediction block is found in the n-th search space.

48. The apparatus according to claim 47, the processing circuitry is configured to:

determine the first search space consisting of (P+1) or P sample positions,

calculate matching cost for the respective candidate motion vector;

comparing matching cost of the estimate of the motion vector with the matching cost of the respective candidate motion vector, and

perform a search within the n-th search space if the matching cost of the estimate of the motion vector is lower than the calculated matching cost of each of a plurality of the candidate motion vectors.

49. The apparatus according to claim 47 or 48, within the first search space, the processing circuitry is further configured to:

calculate the matching cost for the respective candidate motion vector;

comparing the matching cost of the starting sample position with the matching cost of the respective candidate motion vector, and

determining the motion vector for the prediction block if the matching cost of the estimate of the motion vector is lower than the calculated matching cost of each of a plurality of the candidate motion vectors.

50. The apparatus according to claim 48 or 49, wherein the matching cost comprises bilateral matching cost or template matching cost.

51. The apparatus according to any one of claims 47-50, wherein the processing circuitry is specifically configured to:

determining the first search space consisting of the P or (P+1) sample positions;

performing template matching in the first search space to obtain a best matching sample position,

determining the n-th search space based on the best matching sample position, wherein n=2,3,…,K, and K is a predefined threshold; and

performing template matching in the n-th search space to obtain a best matching position.

52. The apparatus according to any one of claims 47-51, wherein if n is equal to a predefined threshold K, the motion vector for the prediction block is found in the n-th search space according to a cost function.

53. The apparatus according to claim 52, wherein the cost function is based on a predetermined template and indicates, for the respective candidate motion vector, a level of similarity between the predetermined template and a predictor pointed to by the respective candidate motion vector.

54. The apparatus according to claim 52 or 53, wherein in the n-th search space, a motion vector pointing to a sample position where the cost function is minimized is a part or whole of the motion vector for the prediction block.

55. The apparatus according to any one of claims 47-54, wherein the first search space consists of P or (P+1) sample positions to which P or (P+1) candidate motion vectors point respectively, and the position pointed by the estimate of the motion vector is located substantially in the center of the first search space, P=4.

56. The apparatus according to any one of claims 47-55, wherein the first search space consists of a central position pointed by the estimate of the motion vector and surrounding horizontal-only and vertical-only integer-pixel distance neighbor positions; or, wherein the first search space consists of horizontal-only and vertical-only sample positions surrounding a central position pointed by the estimate of the motion vector.

57. The apparatus according to any one of claims 47-56, wherein the n-th search space consists of Q or (Q+1) sample positions to which Q or (Q+1) candidate motion vectors point respectively, and the position pointed by a (n-1)-th refined motion vector is located substantially in the center of the n-th search space, $1<Q<=5$.

58. The apparatus according to any one of claims 47-57, wherein the first search space, the n-th search space haveinteger resolution, wherein n=2, 3,...K, K is a predefined threshold.

59. The apparatus according to any one of claims 47-58, wherein the processing circuitry is specifically configured to determine the estimate of the motion vector from a list of motion vectors including motion vectors of at least one block adjacent to a current block.

60. The apparatus according to any one of claims 47-59, wherein the first search space comprises a search space in a first reference picture and a search space in a second reference picture; and/or,

the n-th search space comprises a n_1-th search space in a first reference picture and a n_2-th search space in a second reference picture.

61. An apparatus including a processing circuitry configured for:

- Evaluating a cost function at the refinement center and surrounding horizontal-only and vertical-only 1-pixel distance neighbor positions

- Determining the lowest cost position among these evaluated positions
- Performing subsequent iterations of refinement centered on the lowest cost position of the preceding iteration comprising the steps of:
    - Determining whether the prescribed maximum number of iterations has been exceeded and exiting the refinement loop when it has exceeded
    - Evaluating the cost function at additional positions within the eight 1-pixel distance neighbor positions at which the cost function has not been evaluated in any of the previous iterations
    - Determining the lowest cost position among the center and these remaining positions at which the cost function has not been evaluated in any of the previous iterations;
    - Determining whether the lowest cost position is the current iteration's center position and exiting the refinement loop if the lowest cost position is the current iteration's center position.

62. An encoding apparatus for encoding video images split to prediction blocks into a bitstream, the encoding apparatus comprising:

the apparatus for determination of a motion vector for a prediction block according to any of claims 47 to 61;

an encoding circuitry for encoding difference between the prediction block and the predictor given by a prediction block in a position based on the determined motion vector and for generating bitstream including the encoded difference and the estimate of the motion vector.

63. A decoding apparatus for decoding from a bitstream video images split to prediction blocks, the decoding apparatus comprising:

a parsing unit for parsing from the bitstream an estimate of the motion vector and an encoded difference between a prediction block and a predictor given by a prediction block in a position specified by a refined motion vector;

the apparatus for determination of the refined motion vector for the prediction block according to any of claims 47 to 61;

decoding circuitry for reconstructing the prediction block as a sum of the parsed difference and the predictor given by the prediction block in the position specified by the refined motion vector.

64. A method for determining a motion vector for an image prediction block, the method comprising:

obtaining an estimate of the motion vector;

performing a search within a first search space,wherein searched sample positions are P sample positions adjacent to a position pointed by the estimate of the motion vector in integer-pel distance, and the P sample positions are pointed by a first plurality of candidate motion vectors, wherein P is a positive integer and P is less than 8;

performing a search within a n-th search space, wherein searched sample positions are Q sample positions that are adjacent to a position pointed by a (n-1)-th refined motion vector in integer-pel distance and that are not previously searched, and the Q sample positions are pointed by a second plurality of candidate motion vectors, wherein Q is a positive integer and Q is less than 5 or equal to 5; wherein the (n-1)-th refined motion vector is found in the (n-1)-th search space, wherein n is a positive integer which is larger than 1;

wherein the motion vector for the image prediction block is found in the n-th search space.

65. The method according to claim 64, wherein the obtaining the estimate of the motion vector, comprises: obtaining an initial motion vector and a template for the prediction block.

66. The method according to claim 64, the performing a search within a first search space, comprises:

determining the first search space consisting of (P+1) or P sample positions,

calculate matching cost for the respective candidate motion vector;

comparing matching cost of the estimate of the motion vector with the matching cost of the respective candidate motion vector, and

performing a search within the n-th search space if the matching cost of the estimate of the motion vector is lower than the calculated matching cost of each ofa plurality of the candidate motion vectors.

67. The method according to according to any one of claims 64-66, within the first search space, the performing a search within a first search space, further comprises:

calculate the matching cost for the respective candidate motion vector;

comparing the matching cost of the starting sample position with the matching cost of the respective candidate motion vector, and

determining the motion vector for the prediction block if the matching cost of the estimate of the motion vector is lower than the calculated matching cost of each of a plurality of the candidate motion vectors.

68. The method according to claim 66 or 67, wherein the matching cost comprises bilateral matching cost or template matching cost.

69. The method according to any one of claims 64-68, wherein the performing a search within a first search space, comprises:

determining the first search space consisting of the P or (P+1) sample positions;

performing template matching in the first search space to obtain a best matching sample position,

the performing a search within a n-th search space, comprises:

determining the n-th search space based on the best matching sample position, wherein n=2,3,…,K, and K is a predefined threshold; and

performing template matching in the n-th search space to obtain a best matching position.

70. The method according to any one of claims 64-69, wherein if n is equal to a predefined threshold K, the motion vector for the prediction block is found in the n-th search space according to a cost function.

71. The method according to claim 70, wherein the cost function is based on a predetermined template and indicates, for the respective candidate motion vector, a level of similarity between the predetermined template and a predictor pointed to by the respective candidate motion vector.

72. The method according to claim 70 or 71, wherein in the n-th search space, a motion vector pointing to a sample position where the cost function is minimized is a part or whole of the motion vector for the prediction block.

73. The method according to any one of claims 64-72, wherein the first search space consists of P or (P+1) sample positions to which P or (P+1) candidate motion vectors point respectively, and the position pointed by the estimate of the motion vector (initial motion vector) is located substantially in the center of the first search space, P=4.

74. The method according to any one of claims 64-73, wherein the first search space consists of a central position pointed by the estimate of the motion vector and surrounding horizontal-only and vertical-only integer-pixel distance neighbor positions; or, wherein the first search space consists of horizontal-only and

vertical-only sample positions surrounding a central position pointed by the estimate of the motion vector.

75. The method according to any one of claims 64-74, wherein the n-th search space consists of Q or (Q+1) sample positions to which Q or (Q+1) candidate motion vectors point respectively, and the position pointed by a (n-1)-th refined motion vector is located substantially in the center of the n-th search space, 1<Q<=5.

76. The method according to any one of claims 64-75, wherein the first search space, the n-th search space haveinteger resolution, wherein n=2, 3,...K, K is a predefined threshold.

77. The method according to any one of claims 64-76, wherein the obtaining an estimate of the motion vector comprises: determining the estimate of the motion vector from a list of motion vectors including motion vectors of at least one block adjacent to a current block.

78. The method according to any one of claims 64-77, wherein the first search space comprises a search space in a first reference picture and a search space in a second reference picture; and/or,

the n-th search space comprises a n_1-th search space in a first reference picture and a n_2-th search space in a second reference picture.

79. The method according to any one of claims 64-78, wherein the method is used for Decoder Side Motion Vector Refinement (DMVR) or Decoder Side Motion Vector derivation (DMVD).

80. An integer distance search method for decoder side motion vector refinement comprising:

- Evaluating a cost function at the refinement center and surrounding horizontal-only and vertical-only 1-pixel distance neighbor positions

- Determining the lowest cost position among these evaluated positions
- Performing subsequent iterations of refinement centered on the lowest cost position of the preceding iteration comprising the steps of:
    - Determining whether the prescribed maximum number of iterations has been exceeded and exiting the refinement loop when it has exceeded
    - Evaluating the cost function at additional positions within the eight 1-pixel distance neighbor positions at which the cost function has not been evaluated in any of the previous iterations
    - Determining the lowest cost position among the center and these remaining positions at which the cost function has not been evaluated in any of the previous iterations;
    - Determining whether the lowest cost position is the current iteration's center position and exiting the refinement loop if the lowest cost position is the current iteration's center position.

81. A method for determining a motion vector for an image prediction block, the method comprising:

Obtaining an estimate of the motion vector;

performing a search within a first search space, wherein the first search space is defined by a part of a plurality of candidate motion vectors pointing at the sample positions adjacent to the position pointed by the estimate of the motion vector in integer-pel distance;

performing a search within a n-th search space, wherein the n-th search space is defined by un-searched candidate motion vectors of a whole of a plurality of candidate motion vectors pointing at the sample positions adjacent to the position pointed by a previously-found motion vector in integer-pel distance, wherein the previously-found motion vector is found in the (n-1)-th search space, wherein n is a positive integer which is larger or equal to 2;

wherein the motion vector for the image prediction block is found in the n-th search space.

82. An encoding method for encoding video images split to prediction blocks into a bitstream, the encoding method comprising:

determining a motion vector for a prediction block according to any of claims 64 to 81;

encoding difference between the prediction block and the predictor given by a prediction block in a position based on the determined motion vector and for generating bitstream including the encoded difference and the initial motion vector.

83. A decoding method for decoding from a bitstream video images split to prediction blocks, the decoding method comprising:

parsing from the bitstream an initial motion vector and an encoded difference between a prediction block and a predictor given by a prediction block in a position specified by a refined motion vector;

determining the refined motion vector for the prediction block according to any of claims 64 to 81;

reconstructing the prediction block as a sum of the parsed difference and the predictor given by the prediction block in the position based on the refined motion vector.

84. Computer readable medium storing instructions which when executed on a processor cause the processor to perform the method according to any of claims 64 to 81.

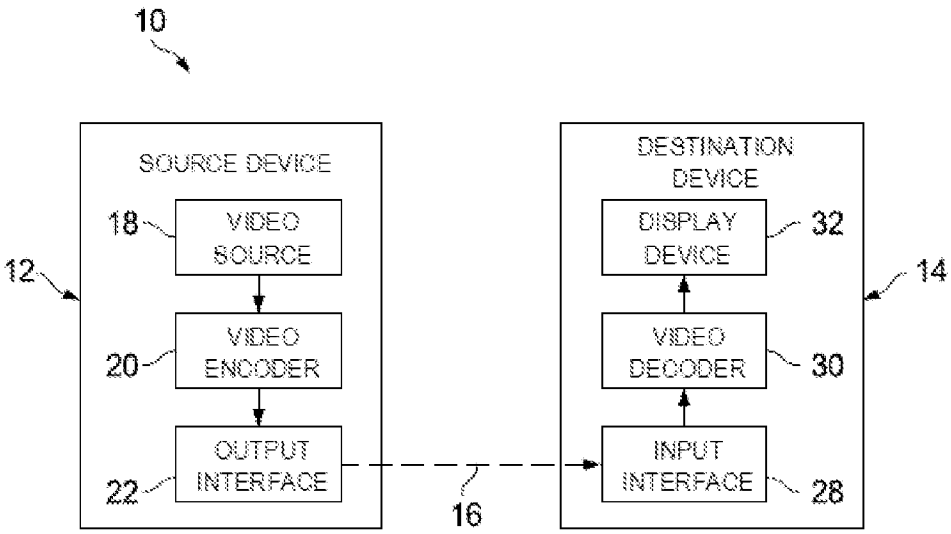
10
SOURCE DEVICE
18
VIDEO SOURCE
12
20
VIDEO ENCODER
22
OUTPUT INTERFACE
16
DESTINATION DEVICE
DISPLAY DEVICE
32
14
VIDEO DECODER
30
INPUT INTERFACE
28

FIG. 1A

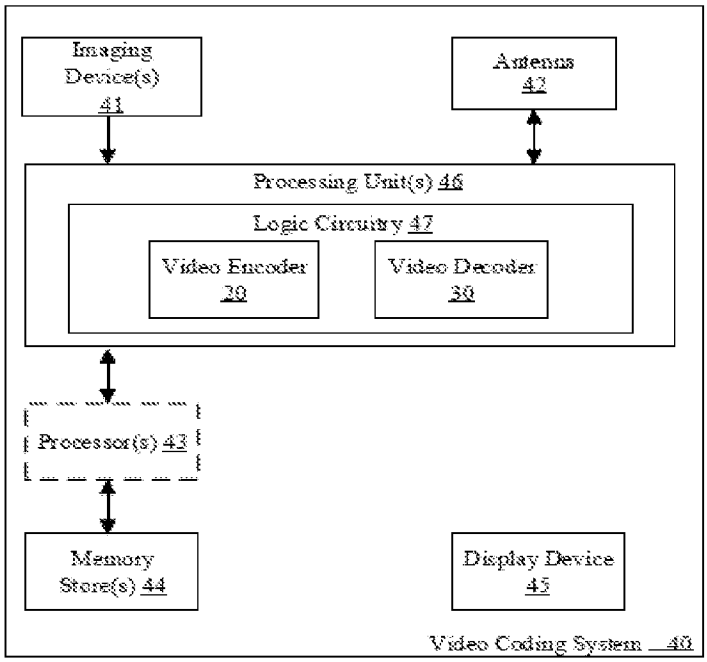
Imaging Device(s) 41
Antenna 42
Processing Unit(s) 46
Logic Circuitry 47
Video Encoder 20
Video Decoder 30
Processor(s) 43
Memory Store(s) 44
Display Device 45
Video Coding System 40

FIG. 1B

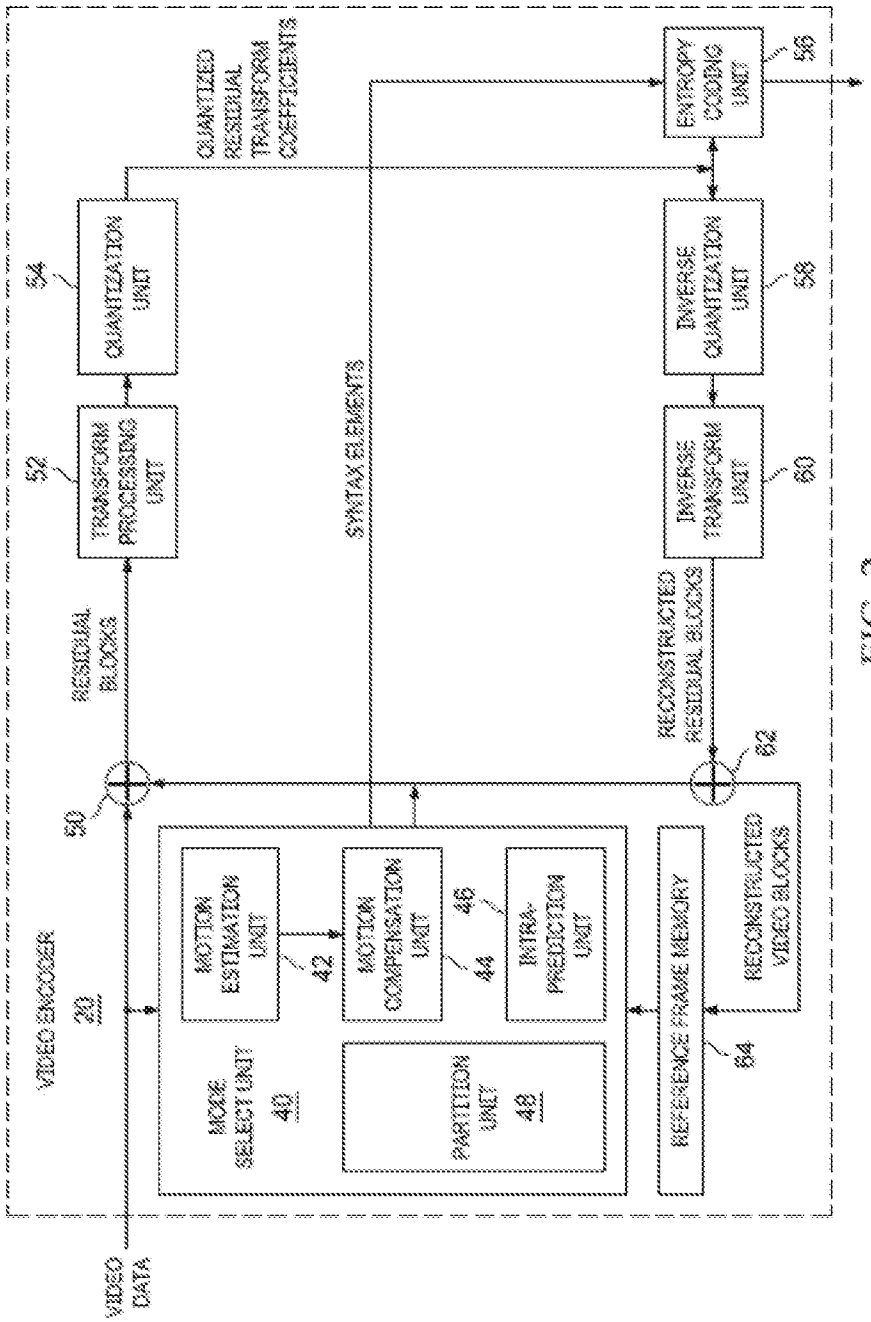
VIDEO ENCODER
20
VIDEO DATA
50
RESIDUAL BLOCKS
52
TRANSFORM PROCESSING UNIT
54
QUANTIZATION UNIT
QUANTIZED RESIDUAL TRANSFORM COEFFICIENTS
SYNTAX ELEMENTS
ENTROPY CODING UNIT
56
INVERSE QUANTIZATION UNIT
58
INVERSE TRANSFORM UNIT
60
RECONSTRUCTED RESIDUAL BLOCKS
62
RECONSTRUCTED VIDEO BLOCKS
MODE SELECT UNIT
40
MOTION ESTIMATION UNIT
42
MOTION COMPENSATION UNIT
44
INTRA-PREDICTION UNIT
46
PARTITION UNIT
48
REFERENCE FRAME MEMORY
64

FIG. 2

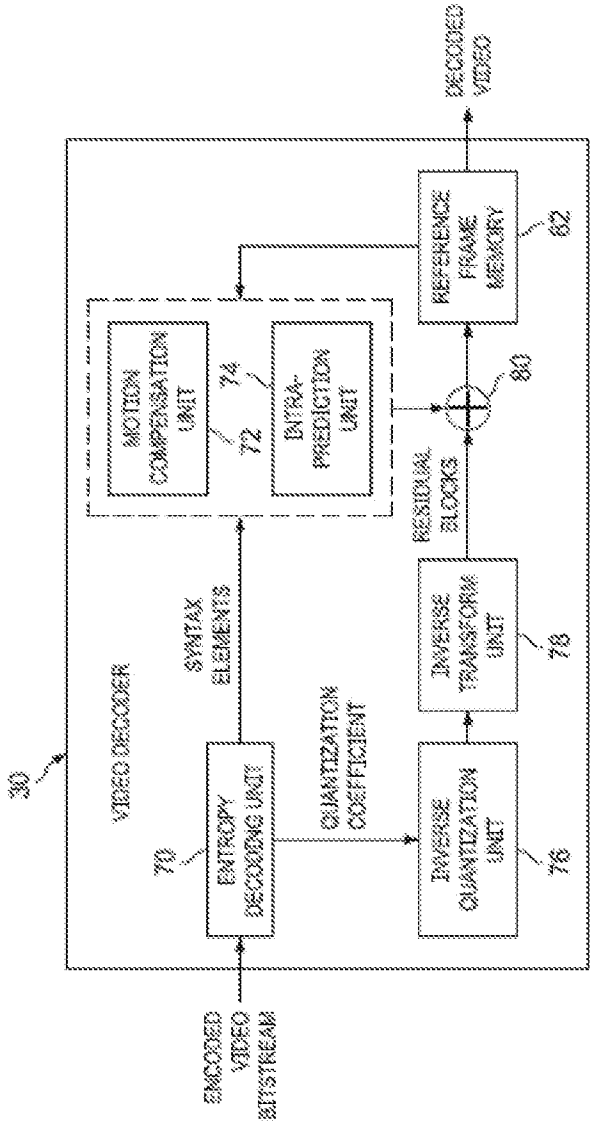
ENCODED VIDEO BITSTREAM
30
VIDEO DECODER
70
ENTROPY DECODING UNIT
SYNTAX ELEMENTS
QUANTIZATION COEFFICIENT
INVERSE QUANTIZATION UNIT
76
INVERSE TRANSFORM UNIT
78
RESIDUAL BLOCKS
80
MOTION COMPENSATION UNIT
72
74
INTRA-PREDICTION UNIT
REFERENCE FRAME MEMORY
82
DECODED VIDEO

FIG. 3

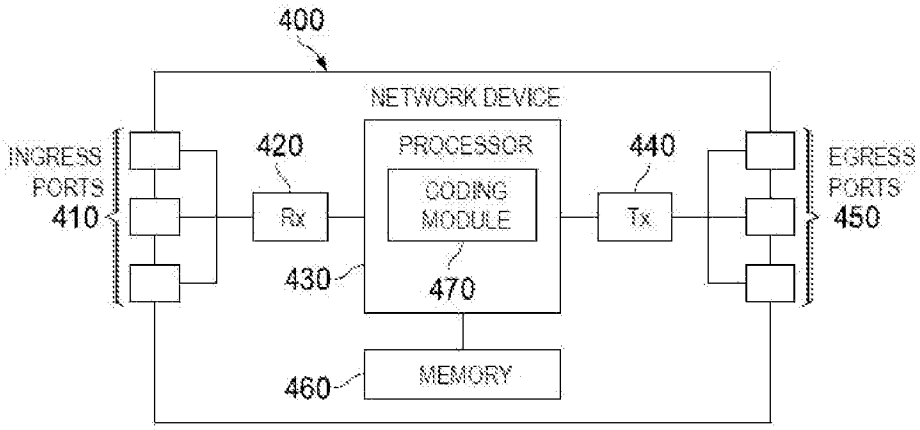
400
NETWORK DEVICE
INGRESS PORTS
410
420
Rx
PROCESSOR
CODING MODULE
430
470
440
Tx
EGRESS PORTS
450
460
MEMORY

FIG. 4

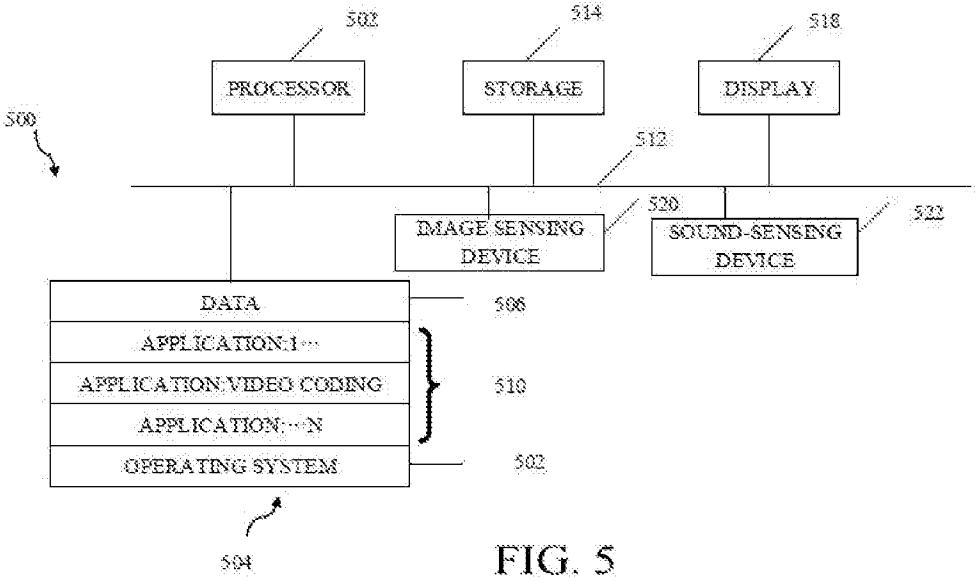
500
502
PROCESSOR
514
STORAGE
516
DISPLAY
512
520
IMAGE SENSING DEVICE
522
SOUND-SENSING DEVICE
DATA
506
APPLICATION:1 ···
APPLICATION:VIDEO CODING
APPLICATION: ··· N
510
OPERATING SYSTEM
502
504

FIG. 5

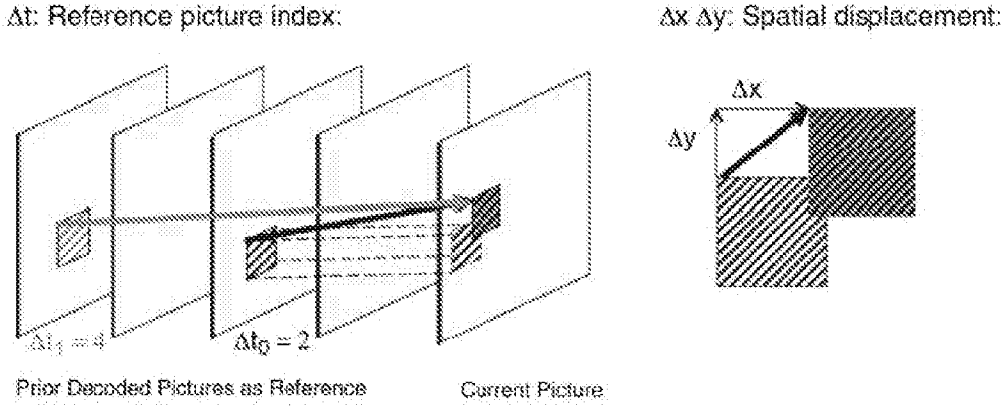
Δt: Reference picture index:
Δx Δy: Spatial displacement:
Δx
Δy
Prior Decoded Pictures as Reference
Current Picture

Fig. 6

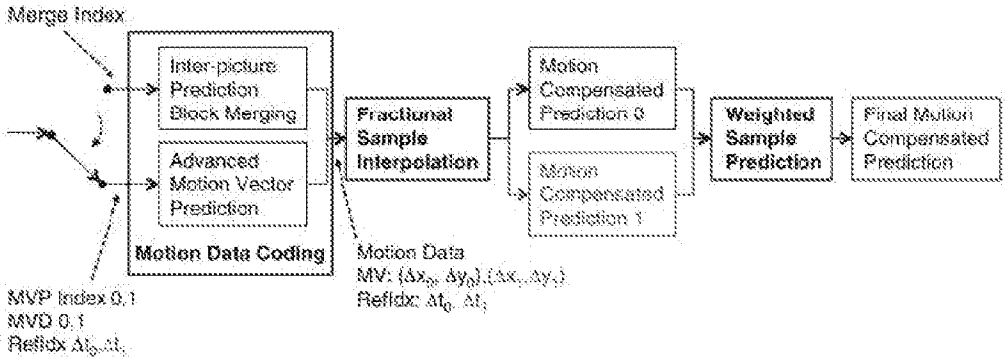
Merge Index
Inter-picture Prediction Block Merging
Advanced Motion Vector Prediction
Motion Data Coding
MVP Index 0,1
MVD 0,1
Refidx Δt0,Δt1
Fractional Sample Interpolation
Motion Data
Refidx: Δt0, Δt1
Motion Compensated Prediction 0
Motion Compensated Prediction 1
Weighted Sample Prediction
Final Motion Compensated Prediction

Fig. 7

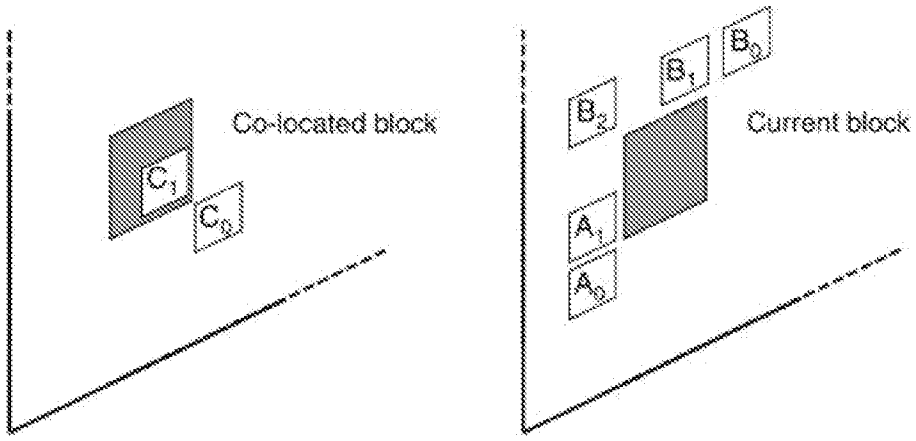
Co-located block
C1
C0
Current block
B2
B1
B0
A1
A0

Fig. 8

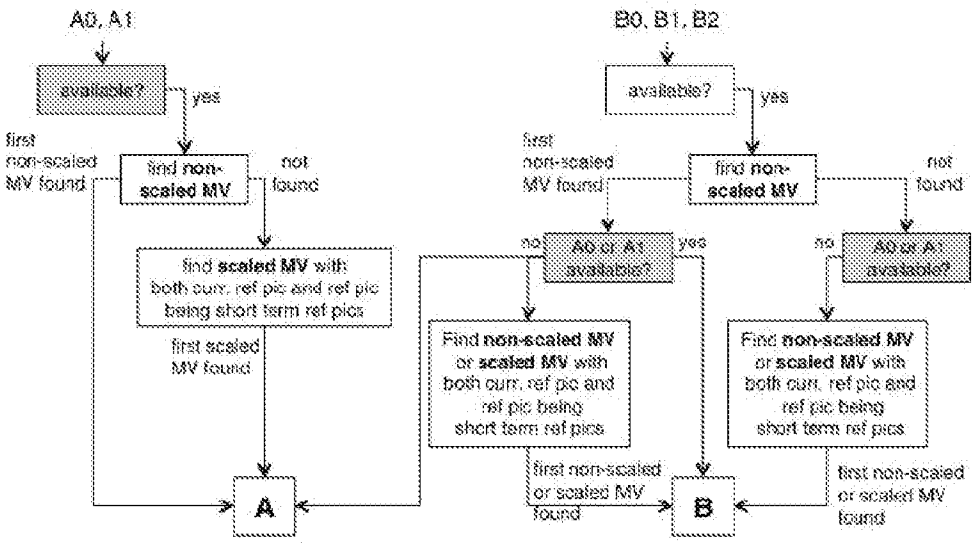
A0, A1
available?
yes
first non-scaled MV found
find non-scaled MV
not found
find scaled MV with both curr. ref pic and ref pic being short term ref pics
first scaled MV found
A
B0, B1, B2
available?
yes
first non-scaled MV found
find non-scaled MV
not found
no
A0 or A1 available?
yes
no
A0 or A1 available?
Find non-scaled MV or scaled MV with both curr. ref pic and ref pic being short term ref pics
Find non-scaled MV or scaled MV with both curr. ref pic and ref pic being short term ref pics
first non-scaled or scaled MV found
first non-scaled or scaled MV found
B

Fig. 9

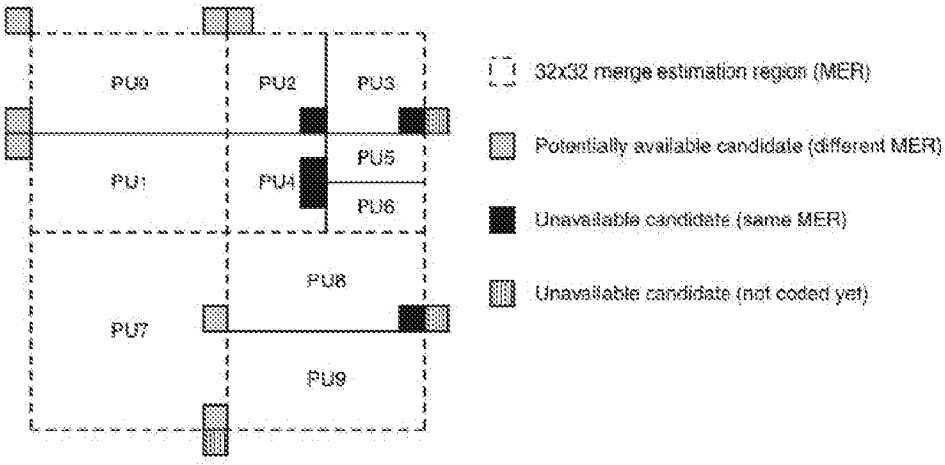

Fig. 10

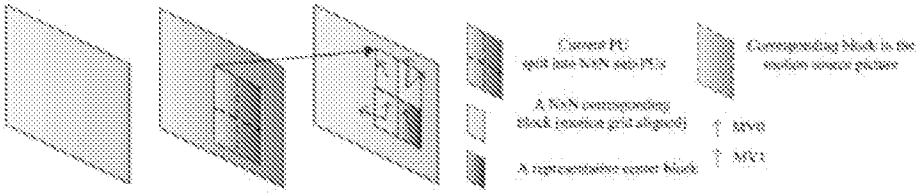

Fig. 11

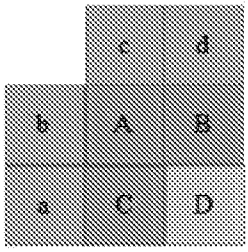

Fig. 12

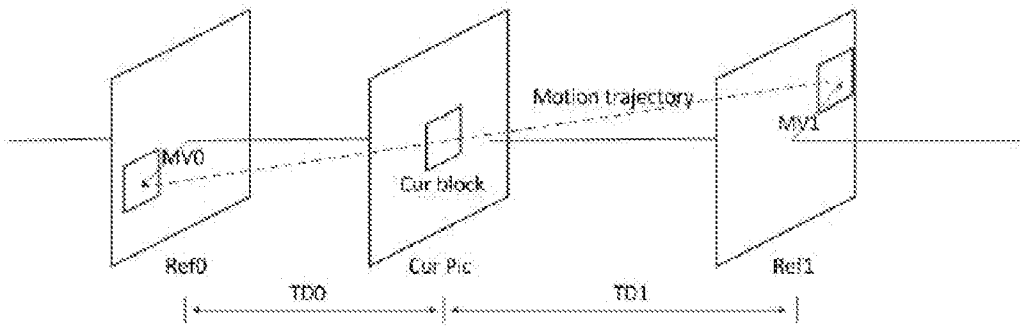
Motion trajectory
MV1
MV0
Cur block
Ref0
Cur Pic
Ref1
TD0
TD1

Fig. 13

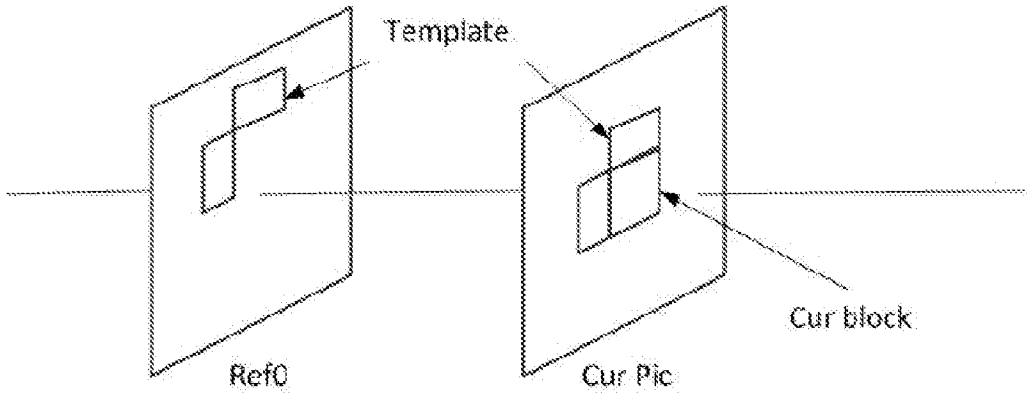
Template
Cur block
Ref0
Cur Pic

Fig. 14

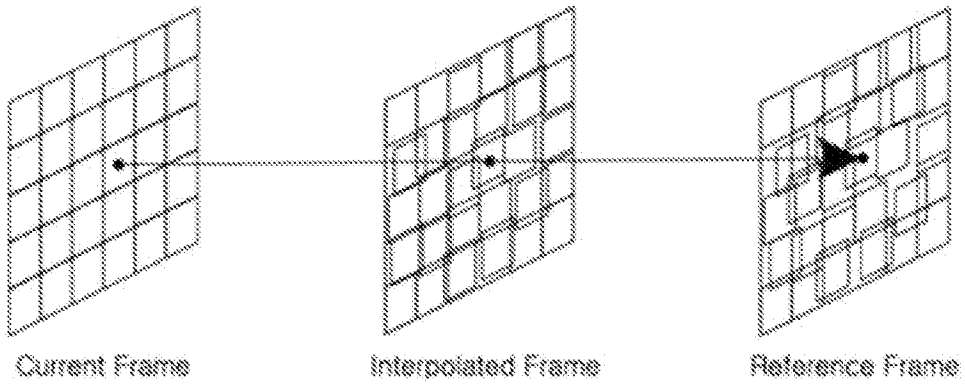
Current Frame
Interpolated Frame
Reference Frame

Fig. 15

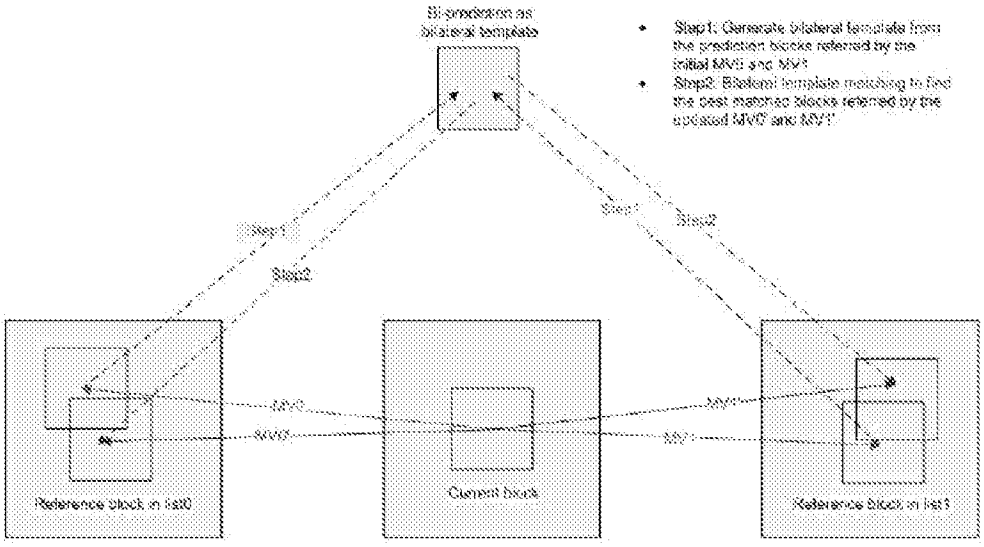
Bi-prediction as bilateral template
Step1: Generate bilateral template from the prediction blocks referred by the initial MV0 and MV1
Step2: Bilateral template matching to find the best matched blocks referred by the updated MV0' and MV1'
Step1
Step2
Step1
Step2
MV0
MV0'
MV1
MV1'
Reference block in list0
Current block
Reference block in list1

Fig. 16

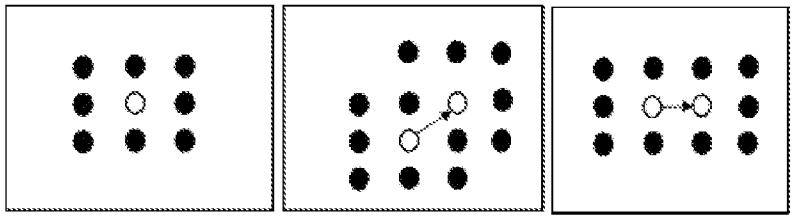

Fig. 17

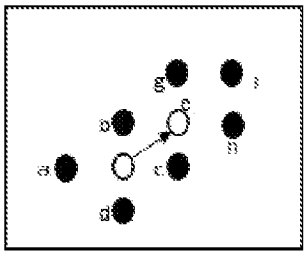

Fig. 18

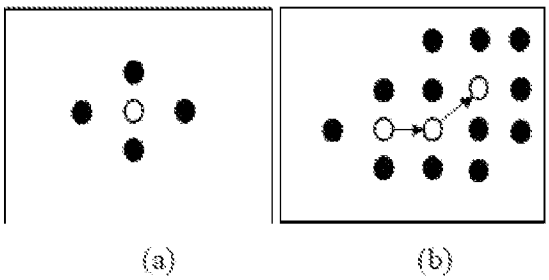

Fig. 19

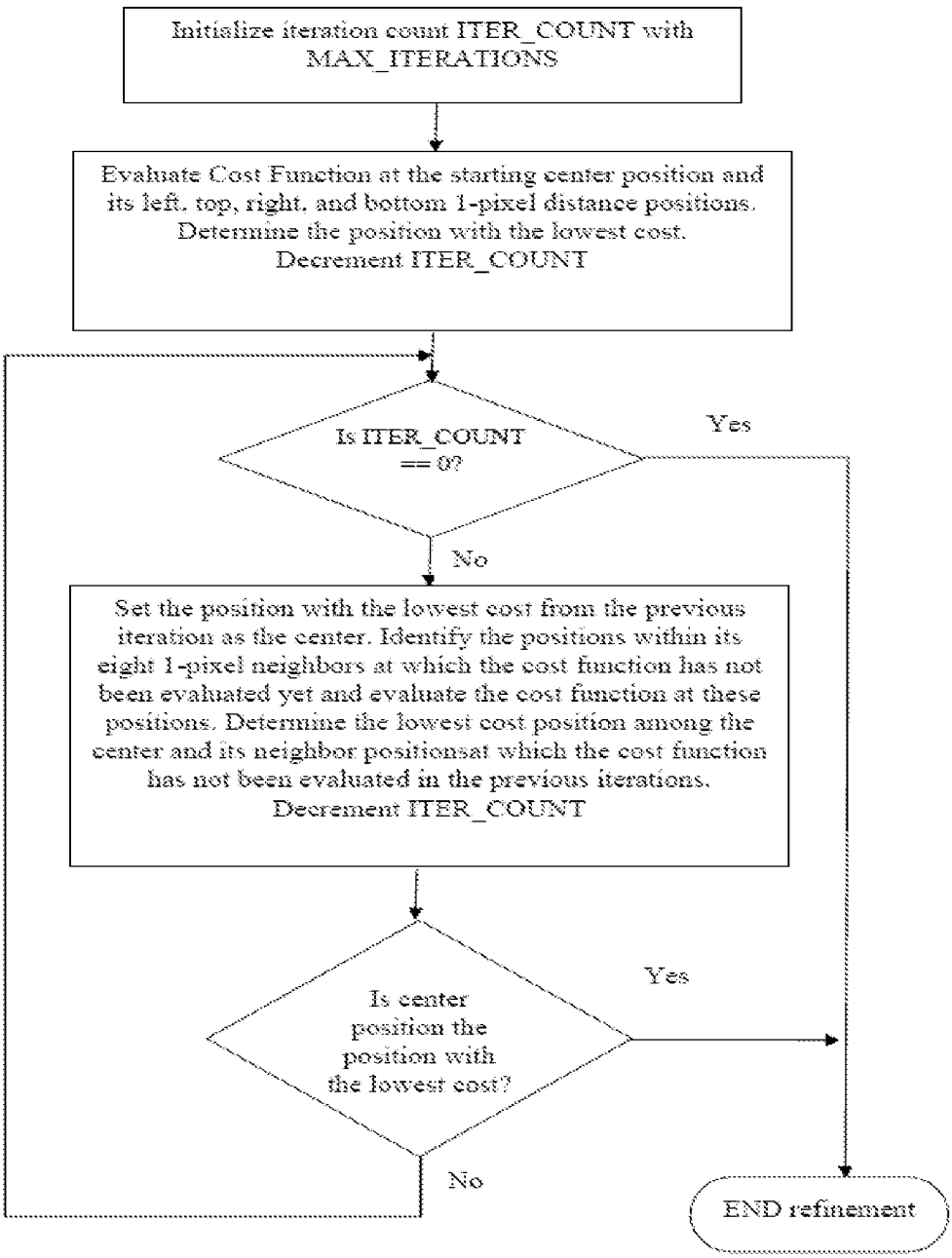
Initialize iteration count ITER_COUNT with MAX_ITERATIONS
Evaluate Cost Function at the starting center position and its left, top, right, and bottom 1-pixel distance positions. Determine the position with the lowest cost. Decrement ITER_COUNT
Is ITER_COUNT == 0?
Yes
No
Set the position with the lowest cost from the previous iteration as the center. Identify the positions within its eight 1-pixel neighbors at which the cost function has not been evaluated yet and evaluate the cost function at these positions. Determine the lowest cost position among the center and its neighbor positionsat which the cost function has not been evaluated in the previous iterations. Decrement ITER_COUNT
Is center position the position with the lowest cost?
Yes
No
END refinement

Fig. 20

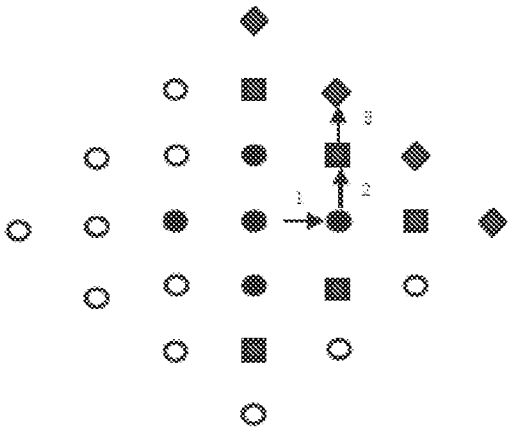

Figure 21

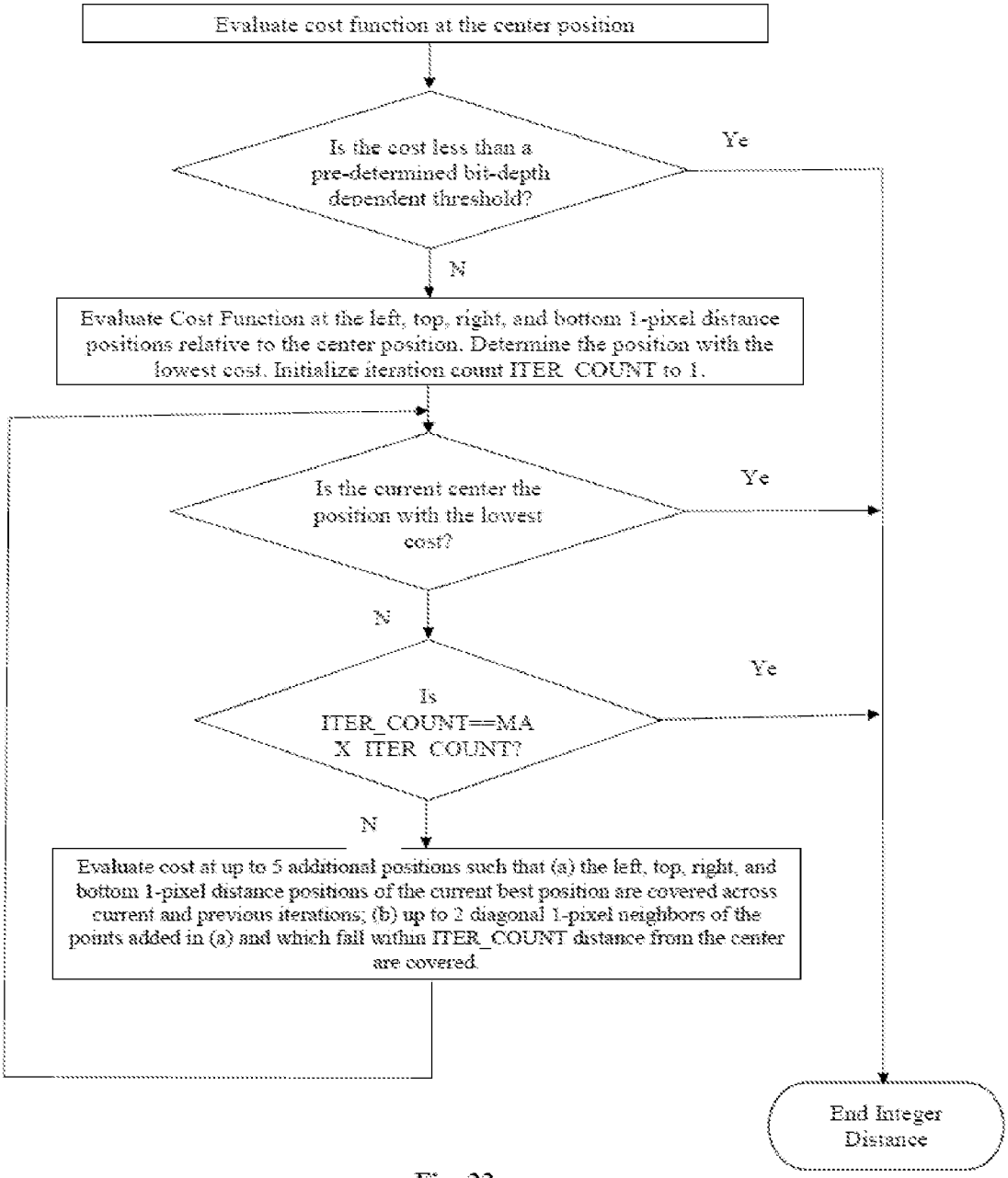
Evaluate cost function at the center position
Is the cost less than a pre-determined bit-depth dependent threshold?
Ye
N
Evaluate Cost Function at the left, top, right, and bottom 1-pixel distance positions relative to the center position. Determine the position with the lowest cost. Initialize iteration count ITER_COUNT to 1.
Is the current center the position with the lowest cost?
Ye
N
Is ITER_COUNT==MAX_ITER_COUNT?
Ye
N
Evaluate cost at up to 5 additional positions such that (a) the left, top, right, and bottom 1-pixel distance positions of the current best position are covered across current and previous iterations; (b) up to 2 diagonal 1-pixel neighbors of the points added in (a) and which fall within ITER_COUNT distance from the center are covered.
End Integer Distance

Fig. 22

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2019/094370

A. CLASSIFICATION OF SUBJECT MATTER

H04N 19/57(2014.01)i

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

H04N

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

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

CNPAT,CNKI,WPI,EPODOC,IEEE: motion vector, encode, video, predict, search, region, space, neighbor, adjacent, position, cost, horizon, vertical

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2015117543 A1 (FOUNDATION OF SOONGSIL UNIVERSITY-INDUSTRY COOPERATION) 30 April 2015 (2015-04-30) description, paragraphs 0074-0088, figures 7-9	1-84
A	WO 2008003220 A1 (HONG KONG APPLIED SCIENCE AND TECHNOLOGY RESEARCH INSTITUTE CO., LTD.) 10 January 2008 (2008-01-10) the whole document	1-84
A	CN 1756354 A (TENCENT TECHNOLOGIES SHENZHEN CO., LTD.) 05 April 2006 (2006-04-05) the whole document	1-84
A	CN 101931803 A (HUAWEI TECHNOLOGIES CO., LTD.) 29 December 2010 (2010-12-29) the whole document	1-84

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

* Special categories of cited documents:

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

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

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

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

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

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

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

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

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
03 September 2019	**27 September 2019**

Name and mailing address of the ISA/CN	Authorized officer
National Intellectual Property Administration, PRC **6, Xitucheng Rd., Jimen Bridge, Haidian District, Beijing 100088** **China**	**SHENG,Jianjun**
Facsimile No. **(86-10)62019451**	Telephone No. **86-(10)-53961819**

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2019/094370

Patent document cited in search report			Publication date (day/month/year)	Patent family member(s)			Publication date (day/month/year)
US	2015117543	A1	30 April 2015	KR	101347062	B1	10 January 2014
				WO	2014003254	A1	03 January 2014
				US	10063881	B2	28 August 2018
WO	2008003220	A1	10 January 2008	US	2008002772	A1	03 January 2008
CN	1756354	A	05 April 2006	CN	100469141	C	11 March 2009
CN	101931803	A	29 December 2010	AU	2010265451	A1	19 January 2012
				CN	102883160	A	16 January 2013
				WO	2010148919	A1	29 December 2010
				CN	101931803	B	09 January 2013
				BR	PI1012992	A2	16 January 2018
				CN	102883160	B	29 June 2016
				EP	2448266	A4	03 April 2013
				KR	20120026092	A	16 March 2012
				EP	2448266	A1	02 May 2012
				AU	2010265451	B2	19 June 2014
				US	9432692	B2	30 August 2016
				KR	101443169	B1	23 September 2014
				US	2012106645	A1	03 May 2012