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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
04 January 2018 (04.01.2018)

WIPO | PCT

(10) International Publication Number
WO 2018/002024 A1

(51) International Patent Classification:
H04N 19/52 (2014.01) *H04N 19/46* (2014.01)
H04N 19/139 (2014.01) *H04N 19/14* (2014.01)
H04N 19/176 (2014.01) *H04N 19/513* (2014.01)

(21) International Application Number:
PCT/EP2017/065809

(22) International Filing Date:
27 June 2017 (27.06.2017)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
16305827.4 30 June 2016 (30.06.2016) EP

(71) Applicant: **THOMSON LICENSING** [FR/FR]; 1-5, rue Jeanne d'Arc, 92130 Issy-les-Moulineaux (FR).

(72) Inventors: **GALPIN, Franck**; c/o Technicolor R&D France, 975, avenue des Champs Blancs, CS 17616, 35576 CESSON-SEVIGNE (FR). **RACAPE, Fabien**; c/o Technicolor R&D France, 975, avenue des Champs Blancs, CS 17616, 35576 CESSON-SEVIGNE (FR). **POIRIER, Tangi**; c/o Technicolor R&D France, 975, avenue des Champs Blancs, CS 17616, 35576 CESSON-SEVIGNE (FR). **BORDES, Philippe**; c/o Technicolor R&D France, 975, avenue des Champs Blancs, CS 17616, 35576 CESSON-SEVIGNE (FR).

(74) Agent: **HUCHET, Anne** et al.; 1-5, rue Jeanne d'Arc, 92130 Issy-les-Moulineaux, FR (FR).

(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).

(54) Title: METHOD AND APPARATUS FOR VIDEO CODING WITH AUTOMATIC MOTION INFORMATION REFINEMENT

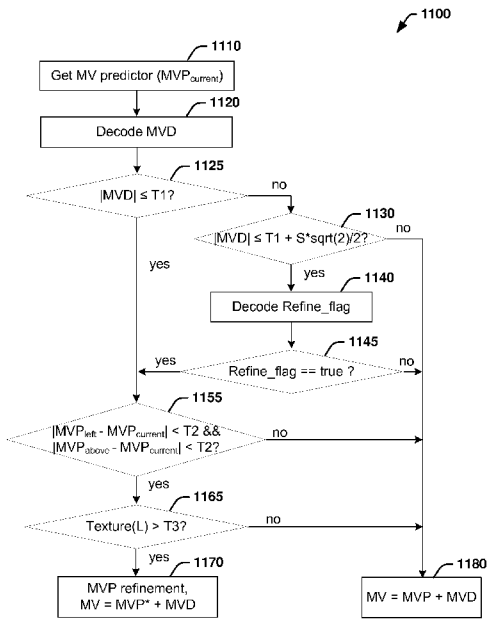

FIG. 11

(57) Abstract: In a particular implementation, a video decoder may refine an initial motion vector predictor decoded from the bitstream for a current block. In order to reduce signaling overhead, the decoder may determine whether or not motion refinement is used based on information that is already available, for example, based on whether the motion field is uniform around the current block, and whether there is a certain level of textureness in adjacent blocks. The motion vector difference decoded from the bitstream can be used to automatically activate or deactivate the motion refinement without receiving explicit signaling in the bitstream. For example, when the motion vector difference is smaller than a threshold, motion refinement is automatically activated, and when the motion vector difference is greater than another threshold, motion refinement is automatically deactivated. A corresponding video encoder may choose whether to use and signal motion refinement based on encoder decisions.

WO 2018/002024 A1

[Continued on next page]

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

Published:

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

METHOD AND APPARATUS FOR VIDEO CODING WITH AUTOMATIC MOTION INFORMATION REFINEMENT

TECHNICAL FIELD

[1] The present principles generally relate to a method and an apparatus for video encoding and decoding, and more particularly, to a method and an apparatus for video encoding and decoding with motion information refinement.

BACKGROUND

[2] To achieve high compression efficiency, image and video coding schemes usually employ prediction and transform to leverage spatial and temporal redundancy in the video content. Generally, intra or inter prediction is used to exploit the intra or inter frame correlation, then the differences between the original image and the predicted image, often denoted as prediction errors or prediction residuals, are transformed, quantized, and entropy coded. To reconstruct the video, the compressed data is decoded by inverse processes corresponding to the prediction, transform, quantization, and entropy coding.

SUMMARY

[3] According to a general aspect, a method for video decoding is presented, comprising: accessing an initial motion vector predictor for a current block of a video, the initial motion vector predictor being associated with a first motion resolution; determining whether or not to refine the initial motion vector predictor, based on a motion vector difference; and if the initial motion vector predictor is determined to be refined, refining the initial motion vector predictor based on motion search to form a refined motion vector predictor and decoding the current block based on the refined motion vector predictor, the refined motion vector predictor being associated with a second motion resolution, and the second motion resolution being higher than the first motion resolution.

[4] When a magnitude of the motion vector difference is smaller than a first threshold, the initial motion vector predictor may be determined to be refined. When a magnitude of the motion vector difference exceeds a second threshold, the initial motion vector predictor may be determined not to be refined, and the current block is decoded based on the initial motion

vector predictor. Here, the activation or deactivation of motion refinement is automatically performed, without explicit signaling.

[5] When a magnitude of the motion vector difference exceeds the first threshold but not the second threshold, a flag may be decoded from a bitstream, and whether or not the initial motion vector predictor is to be refined is based on the decoded flag.

[6] The motion vector of adjacent decoded blocks can also be used to determine whether or not to perform motion refinement. For example, if a difference between the initial motion vector predictor for the current block and the at least one motion vector of adjacent decoded blocks is smaller than a third threshold, the initial motion vector predictor is determined to be refined. In addition, the texture level can also be used. For example, if a texture level of a plurality of pixels of adjacent decoded blocks exceeds a fourth threshold, the initial motion vector predictor is determined to be refined.

[7] According to another general aspect, a method for video encoding is presented, comprising: accessing an initial motion vector predictor, the initial motion vector predictor associated with a first motion resolution; determining whether or not to explicitly signal a refinement of the initial motion vector predictor, based on a motion vector difference corresponding to the initial motion vector predictor, the refinement being associated with a second motion resolution, and the second motion resolution being higher than the first motion resolution; and encoding the motion vector difference.

[8] When a magnitude of the motion vector difference is smaller than the first threshold but not the second threshold, the refinement of the initial motion vector predictor is not explicitly signaled. When a magnitude of the motion vector difference exceeds a second threshold, the refinement of the initial motion vector predictor is deactivated without explicitly signaling.

[9] When a magnitude of the motion vector difference exceeds a first threshold, a flag is encoded into a bitstream to explicitly signal whether or not the initial motion vector predictor is to be refined.

[10] The motion vector of adjacent reconstructed blocks can also be used to determine whether or not to perform motion refinement. For example, if a difference between the initial motion vector predictor for the current block and the at least one motion vector of adjacent reconstructed blocks is smaller than a third threshold, the initial motion vector

predictor is determined to be refined. In addition, the texture level can also be used. For example, if a texture level of the plurality of pixels exceeds a fourth threshold, the initial motion vector predictor is determined to be refined.

[11] The present embodiments also provide an apparatus for performing these methods.

[12] The present embodiments also provide a non-transitory computer readable storage medium having stored thereon instructions for performing any of the methods described above.

[13] The present embodiments also provide a bitstream generated according to the methods described above.

BRIEF DESCRIPTION OF THE DRAWINGS

[14] FIG. 1 illustrates an exemplary HEVC (High Efficiency Video Coding) encoder.

[15] FIG. 2A is a pictorial example depicting the positions of five spatial candidates $\{a_1, b_1, b_0, a_0, b_2\}$ for a current block, FIG. 2B is a pictorial example depicting an exemplary motion vector representation using AMVP (Advanced Motion Vector Prediction), and FIG. 2C is a pictorial example depicting motion-compensated prediction.

[16] FIG. 3 illustrates a block diagram of an exemplary HEVC video decoder.

[17] FIG. 4 illustrates an exemplary method for performing motion vector refinement at a decoder, according to an embodiment of the present principles.

[18] FIG. 5 is a pictorial example depicting pixel positions for integer pixels, half pixels, quarter pixels and eighth pixels.

[19] FIG. 6A illustrates an exemplary PU to be decoded, and FIG. 6B is a pictorial example illustrating an L-shape set of neighboring reconstructed samples and an L-shape set of prediction samples for measuring discontinuity.

[20] FIG. 7A is a pictorial example illustrating local gradients, and FIG. 7B is a pictorial example illustrating second order moments of gradients.

[21] FIG. 8 illustrates an exemplary method for performing motion vector refinement at an encoder, according to an embodiment of the present principles.

[22] FIG. 9 illustrates an exemplary method for performing motion vector predictor refinement at a decoder, according to an embodiment of the present principles.

[23] FIG. 10 illustrates an exemplary method for performing motion vector predictor refinement at an encoder, according to an embodiment of the present principles.

[24] FIG. 11 illustrates an exemplary method for performing motion vector predictor refinement with automatic motion refinement activation at a decoder, according to an embodiment of the present principles.

[25] FIG. 12 illustrates an exemplary method for performing motion vector predictor refinement with automatic motion refinement activation at an encoder, according to an embodiment of the present principles.

[26] FIG. 13 illustrates an exemplary method for determining whether or not to use adaptive motion refinement for a particular MVP, according to an embodiment of the present principles.

[27] FIG. 14 illustrates a block diagram of an exemplary system in which various aspects of the exemplary embodiments of the present principles may be implemented.

DETAILED DESCRIPTION

[28] FIG. 1 illustrates an exemplary HEVC (High Efficiency Video Coding) encoder 100. To encode a video sequence with one or more pictures, a picture is partitioned into one or more slices where each slice can include one or more slice segments. A slice segment is organized into coding units, prediction units and transform units.

[29] The HEVC specification distinguishes between "blocks" and "units," where a "block" addresses a specific area in a sample array (e.g., luma, Y), and the "unit" includes the collocated block of all encoded color components (Y, Cb, Cr, or monochrome), syntax elements and prediction data that are associated with the block (e.g., motion vectors).

[30] For coding, a picture is partitioned into coding tree blocks (CTB) of square shape with a configurable size, and a consecutive set of coding tree blocks is grouped into a slice. A Coding Tree Unit (CTU) contains the CTBs of the encoded color components. A CTB is the root of a quadtree partitioning into Coding Blocks (CB), and a Coding Block is partitioned into one or more Prediction Blocks (PB) and forms the root of a quadtree

partitioning into Transform Blocks (TBs). Corresponding to the Coding Block, Prediction Block and Transform Block, a Coding Unit (CU) includes the Prediction Units (PUs) and the tree-structured set of Transform Units (TUs), a PU includes the prediction information for all color components, and a TU includes residual coding syntax structure for each color component. The size of a CB, PB and TB of the luma component applies to the corresponding CU, PU and TU. In the present application, the term "block" can be used to refer to any of CTU, CU, PU, TU, CB, PB and TB. In addition, the "block" can also be used to refer to a macroblock, a partition and a sub-block as specified in H.264/AVC or other video coding standards, and more generally to refer to an array of data of various sizes.

[31] In the exemplary encoder 100, a picture is encoded by the encoder elements as described below. The picture to be encoded is processed in units of CUs. Each CU is encoded using either an intra or inter mode. When a CU is encoded in an intra mode, it performs intra prediction (160). In an inter mode, motion estimation (175) and compensation (170) are performed. The encoder decides (105) which one of the intra mode or inter mode to use for encoding the CU, and indicates the intra/inter decision by a prediction mode flag. Prediction residuals are calculated by subtracting (110) the predicted block from the original image block.

[32] CUs in intra mode are predicted from reconstructed neighboring samples within the same slice. A set of 35 intra prediction modes is available in HEVC, including a DC, a planar and 33 angular prediction modes. The intra prediction reference is reconstructed from the row and column adjacent to the current block. The reference extends over two times the block size in horizontal and vertical direction using available samples from previously reconstructed blocks. When an angular prediction mode is used for intra prediction, reference pixels can be copied along the direction indicated by the angular prediction mode.

[33] The applicable luma intra prediction mode for the current block can be coded using two different options. If the applicable mode is included in a constructed list of three most probable modes (MPM), the mode is signaled by an index in the MPM list. Otherwise, the mode is signaled by a fixed-length binarization of the mode index. The three most probable modes are derived from the intra prediction modes of the top and left neighboring blocks.

[34] For an inter CU, the corresponding coding block is further partitioned into one or more prediction blocks. Inter prediction is performed on the PB level, and the

corresponding PU contains the information about how inter prediction is performed.

[35] The motion information (i.e., motion vector and reference index) can be signaled in two methods, namely, "merge mode" and "advanced motion vector prediction (AMVP)."

[36] In the merge mode, a video encoder or decoder assembles a candidate list based on already coded blocks, and the video encoder signals an index for one of the candidates in the candidate list. At the decoder side, the motion vector (MV) and the reference picture index are reconstructed based on the signaled candidate.

[37] The set of possible candidates in the merge mode consists of spatial neighbor candidates, a temporal candidate, and generated candidates. FIG. 2A shows the positions of five spatial candidates $\{a_1, b_1, b_0, a_0, b_2\}$ for a current block 210. For each candidate position, the availability is checked according to the order of a_1, b_1, b_0, a_0, b_2, and then the redundancy in candidates is removed.

[38] The maximum number of merge candidates N is specified in the slice header. If the number of merge candidates is larger than N, only the first N – 1 spatial candidates and the temporal candidate are used. Otherwise, if the number of merge candidates is less than N, the set of candidates is filled up to the maximum number N.

[39] In AMVP, a video encoder or decoder assembles candidate lists based on motion vectors determined from already coded blocks. The video encoder then signals an index in the candidate list to identify a motion vector predictor (MVP) and signals a motion vector difference (MVD). At the decoder side, the motion vector (MV) is reconstructed as MVP+MVD.

[40] Only two spatial motion candidates are chosen in AMVP. The first spatial motion candidate is chosen from left positions $\{a_0, a_1\}$ and the second one from the above positions $\{b_0, b_1, b_2\}$, while keeping the searching order as indicated in the two sets. If the number of motion vector predictors is not equal to two, the temporal MV prediction candidate can be included. If the temporal candidate is not available, a zero motion vector is used to fill the set of candidates.

[41] FIG. 2B illustrates an exemplary motion vector representation using AMVP. For a current block to be encoded (240), a motion vector ($MV_{current}$) can be obtained through motion estimation. Using the motion vector (MV_{left}) from a left block (230) and the motion

vector (MV_{above}) from the above block (220), a motion vector predictor can be chosen as $MVP_{current}$. A motion vector difference then can be calculated as $MVD_{current} = MV_{current} - MVP_{current}$.

[42] In HEVC, the precision of the motion information for motion compensation is one quarter-sample (also referred to as quarter-pel or 1/4-pel) for the luma component and one eighth-sample (also referred to as 1/8-pel) for the chroma components. A 7-tap or 8-tap interpolation filter is used for interpolation of fractional-sample pixel positions, i.e., 1/4, 1/2 and 3/4 of full pixel locations in both horizontal and vertical directions can be addressed for luma.

[43] The prediction residuals are then transformed (125) and quantized (130). The quantized transform coefficients, as well as motion vectors and other syntax elements, are entropy coded (145) to output a bitstream. The encoder may also skip the transform and apply quantization directly to the non-transformed residual signal on a 4x4 TU basis. The encoder may also bypass both transform and quantization, i.e., the residual is coded directly without the application of the transform or quantization process. In direct PCM coding, no prediction is applied and the coding unit samples are directly coded into the bitstream.

[44] The encoder decodes an encoded block to provide a reference for further predictions. The quantized transform coefficients are de-quantized (140) and inverse transformed (150) to decode prediction residuals. Combining (155) the decoded prediction residuals and the predicted block, an image block is reconstructed. A filter (165) is applied to the reconstructed picture, for example, to perform deblocking/SAO (Sample Adaptive Offset) filtering to reduce blockiness artifacts. The filtered image is stored at a reference picture buffer (180).

[45] FIG. 3 illustrates a block diagram of an exemplary HEVC video decoder 300. In the exemplary decoder 300, a bitstream is decoded by the decoder elements as described below. Video decoder 300 generally performs a decoding pass reciprocal to the encoding pass as described in FIG. 1, which performs video decoding as part of encoding video data.

[46] In particular, the input of the decoder includes a video bitstream, which may be generated by video encoder 100. The bitstream is first entropy decoded (330) to obtain transform coefficients, motion vectors, and other coded information. The transform coefficients are de-quantized (340) and inverse transformed (350) to decode the prediction

residuals. Combining (355) the decoded prediction residuals and the predicted block, an image block is reconstructed. The predicted block may be obtained (370) from intra prediction (360) or motion-compensated prediction (i.e., inter prediction) (375). As described above, AMVP and merge mode techniques may be used to derive motion vectors for motion compensation, which may use interpolation filters to calculate interpolated values for sub-integer pixels of a reference block. A filter (365) is applied to the reconstructed image. The filtered image is stored at a reference picture buffer (380).

[47] In video compression, the inter prediction mode allows for predicting one block (for example, Prediction Unit) using at least one motion compensated block from previously reconstructed/decoded pictures. For example, as illustrated in FIG. 2C, a current block (255) is encoded using the inter prediction mode, blocks 225 and 245 are co-located blocks in reference pictures, and blocks 215 and 235 are blocks used for motion-compensated prediction. At least one reference index (e.g., refIdx0) identifying a reconstructed picture in the Decoded Pictures Buffer (DPB) and one motion information (Motion Vector Difference, MVD) are encoded, to enable reconstruction of at least one motion vector (MV, e.g., mv0) at the decoder. When bi-prediction is used, addition reference index (e.g., refIdx1) and motion information are encoded in order to reconstruct the motion vectors (e.g., mv0 and mv1). In some cases, the motion vectors can be inferred from previously decoded data (e.g., merge mode in HEVC).

[48] The reconstruction of MV values can be performed as follows:

$$MV_X = MVP_X + MVD_X,\ MV_Y = MVP_Y + MVD_Y, \quad (1)$$

where MV_X and MV_Y are the horizontal and vertical motion vector components, respectively, MVP (MVP_X, MVP_Y) is the motion vector predictor built from previously reconstructed data, and MVD (MVD_X, MVD_Y) is the motion vector difference that is encoded and transmitted in the bitstream.

[49] The MVD values are usually encoded at a precision corresponding to the decoded MV values. For example, HEVC uses one-quarter pixel (i.e., 1/4-pel) as the motion vector resolution.

[50] Increasing the MV resolution, for example, from 1/4-pel to 1/8-pel, can improve the prediction in general. However, for low bit rates, the coding of the MVD data can have a relatively high bitrate cost with respect to other data encoded per block. Thus, the overall

compression efficiency may not necessarily improve with the MV resolution.

[51] To improve the compression efficiency, there are some existing works on using adaptive motion resolution, for example, to choose a motion resolution between integer pel or 1/4-pel, or to choose between 1/4-pel and 1/6-pel. However, indicating which motion vector resolution is used may cause a degradation in compression efficiency because of the extra side information that needs to be sent in the bitstream.

[52] To reduce overhead, an article by Lakshman, Haricharan, et al., entitled "Conditional motion vector refinement for improved prediction," Picture Coding Symposium (PCS), 2012. IEEE (hereinafter "Lakshman"), defines a set of rules known to both the encoder and decoder to infer the MV resolution, between quarter-sample MV resolution and 1/6-sample MV resolution, without any explicit block-by-block forward signaling. A high resolution MV is transmitted as a regular quarter-sample MV augmented with refinement information, which increases the resolution of the motion vectors from a quarter-sample resolution to one-sixth of a sample. The reconstructed MV components that point to integer or half-sample positions are left unaltered. For the MV components that point to one-quarter or three-quarter positions, the decoder infers the presence of refinement using the following conditions:

- In case of a P-slice, the MV refinement information is always sent.
- In case of a Bi-prediction, the MV refinement is sent for the predictions that access samples from a reference picture that contains high texture.
- In case of Bi-predictions not accessing high texture reference pictures, the MV refinement is sent only for pictures from a pre-defined reference picture list.
- For single hypothesis predictions in B-slices, MV refinement is not used.

[53] An article by Chen, Jianle, et al., entitled "Further improvements to HMKTA-1.0," VCEG-AZ07, ITU-T/SG16 Video Coding Experts Group (VCEG) 52nd Meeting: 19–26 June 2015, Warsaw, Poland (hereinafter "Chen"), describes an Advanced Motion Vector Resolution (AMVR) mode. In particular, MVD between the motion vector and predicted motion vector of a PU can be coded with either quarter-pel resolution or integer-pel resolution. The MVD resolution is controlled at coding unit (CU) level and an integer MVD resolution flag is conditionally signaled for each CU that has at least one non-zero MVD components. When the integer MVD resolution flag is false, or not coded for a CU, the default quarter-pel MV resolution is used for all PUs belonging to the CU. Otherwise, all

PUs coded with AMVP mode belonging to the CU use integer MV resolution, while the PUs coded with merge mode still use quarter-pel MV resolution. When a PU uses integer MV resolution, the AMVP candidate list is filled with integer MV by rounding quarter-pel MVs to integer-pel MVs.

[54] The present principles are directed to motion vector refinement in video encoding and decoding. In one embodiment, a first resolution MVD (for example, 1/4-pel) is received in the bitstream, and the decoder obtains an initial motion vector based on the MVD, and refines the initial motion vector to obtain a second motion resolution (for example, 1/8-pel) higher than the first resolution, using already decoded neighboring samples. The refinement of motion information is also performed at the encoder to avoid mismatch. Because the motion refinement is performed at both the encoder and decoder, no additional information is transmitted in the bitstream to indicate the motion refinement, for example, no explicit signaling is used to indicate the refinement of a motion vector from 1/4-pel to 1/8-pel. In a variant, merge mode is used for the current block (i.e., no MVD is received in the bitstream), and the initial motion vector is obtained as a motion vector predictor in the candidate list (possibly rounded to the first resolution) is refined to obtain the second motion resolution.

[55] The motion information refinement may be performed using motion estimation at the encoder or decoder. Comparing with an approach that uses the first motion resolution (for example, 1/4-pel) without motion resolution refinement, the proposed embodiment may improve the motion accuracy without the overhead of transmitting the refinement motion information. Comparing with an approach that uses the second motion resolution (for example, 1/8-pel) without motion resolution refinement at the decoder, the proposed embodiment may reduce the overhead of transmitting the motion refinement information. It should be noted that the decoder according to the present principles may also perform motion search, and thus may be more complex than a decoder that does not need motion search.

[56] FIG. 4 illustrates an exemplary method 400 for performing motion vector refinement at a decoder, according to an embodiment of the present principles. In this example, we suppose a decoder can refine a motion vector resolution from 1/4-pel to 1/8-pel. It should be noted that the present principles can be applied to refinement between other motion resolutions, for example, but not limited to, from integer-pel to 1/4-pel, from1/4-pel to 1/6-pel.

[57] For a block to be decoded, the decoder checks whether adaptive motion vector

resolution is enabled (410), for example, using a flag in the bitstream, or using the existing methods as described above. If adaptive motion vector resolution is enabled, the decoder generates MV predictor (MVP) at 1/4-pel resolution (420), for example, using the AMVP mode of HEVC. If the predictor is at 1/8-pel, it is rounded to ¼-pel. The decoder then decodes MVD at 1/4-pel resolution (430). An initial motion vector can be obtained (440) as $MV_0 = MVP + MVD$. Then the initial motion vector (MV_0) can be refined to 1/8-pel resolution (450) and be used for motion-compensated prediction.

[58] At 420, the MV predictors may also be generated at 1/8 pel. Whether the MV predictor is generated at 1/4-pel or 1/8-pel should be consistent with what the encoder has used.

[59] The samples of a motion-compensated prediction block are obtained from those of a corresponding block at a position displaced by the motion vector in a reference picture identified by a reference picture index. When the motion vector is not an integer, fractional sample interpolation is used to generate the prediction samples.

[60] HEVC supports motion vectors at 1/4-pel. Let us represent a motion vector as MV = iMV + sMV, where iMV is the integer part of MV and sMV is the ¼-pel part (first motion resolution), sMV = p/4, p = 0, 1, 2, 3, then the interpolated sample value can be calculated as:

$$Ival[x] = \sum_{i=0}^{N-1} c[p][i] \times s[x + iMV - N/2 + i] \quad (2)$$

where x is the pixel location, $c[p][i], i = 0,\ldots,N-1$, are the filter coefficients corresponding to the p/4-pel position and N is the number of filter taps. The filter coefficients for 1/4-pel interpolation in HEVC are as follows:

c[p=0][]={0, 0, 0, 64, 0, 0, 0, 0}, corresponding to integer position, and there is only scaling,

c[p=1][]={-1, 4, -10, 58, 17, -5, 1, 0}, corresponding to 1/4-pel position, and a 7-tap filter,

c[p=2][]={-1, 4, -11, 40, 40, -11, 4, -1}, corresponding to 1/2-pel position, and a 8-tap filter,

c[p=3][]={0, 1, -5, 17, 58, -10, 4, -1}, corresponding to 1/4-pel position, and a 7-tap filter.

[61] When using a higher resolution 1/8-pel, 1/8-pel interpolation filter is used to interpolate the 1/8-pel part, including sMV = p/8, p = 0, 1, 2, 3, 4, 5, 6, 7, 8. For example the following 1/8-pel interpolation filters (N=4) can be used:

c[p=0][] = { 0, 64, 0, 0 },

c[p=1][] = { -2, 58, 10, -2 },

c[p=2][] = { -4, 54, 16, -2 },

c[p=3][] = { -6, 46, 28, -4 },
c[p=4][] = { -4, 36, 36, -4 },
c[p=5][] = { -4, 28, 46, -6 },
c[p=6][] = { -2, 16, 54, -4 },
c[p=7][] = { -2, 10, 58, -2 }.

[62] The motion refinement process may be regarded as motion estimation, with a range related to the motion resolution before refinement. For example, when a motion vector is refined from a first resolution of 1/4-pel to a second resolution of 1/8-pel, the motion search range include candidates around the initial motion vector. More generally, the search candidates for motion refinement can be $MVX_0 - \Delta MVX \sim MVX_0 + \Delta MVX$, $MVY_0 - \Delta MVY \sim MVY_0 + \Delta MVY$, in the horizontal direction and vertical direction, respectively, wherein MVX_0 and MVY_0 are horizontal and vertical components of the initial motion vector MV_0, respectively, and ΔMVX and ΔMVY define the search range in the horizontal and vertical directions, respectively.

[63] For example, the refinement candidates in the horizontal direction can be $(MVX_0 - k \times step2, MVX_0 + k \times step2$, where k is an integer and is defined as

$$-step1 < (k * step2) < step1, \quad (3)$$

where step1 is the first resolution, and step2 is the second resolution. When step1 = 1/4 and step2 = 1/8, $-2 < k < 2$. FIG. 5 illustrates pixel positions for integer pixels, half pixels, quarter pixels and eighth pixels. For ease of notation, we may also refer to those candidates as motion refinement candidates, and we denote a predicted block built by motion compensation with a motion refinement candidate as a motion-refinement predicted block and samples within a motion-refinement predicted block as motion-refinement prediction samples.

[64] An exemplary set of search candidates includes the positions within box 520 when (MVX_0, MVY_0) corresponds to position 510. The search range, and more generally, the set of search candidates can be different from what is shown in FIG. 5. The same set of search candidates should be used at the encoder and decoder.

[65] If adaptive motion vector resolution is not enabled, the decoder obtains MVP and MVD at a motion resolution that is used for the decoded motion vector, and no motion refinement is performed at the decoder. As shown in FIG. 4, the decoder may get (460, 470)

both MVP and MVD at 1/8-pel resolution, and decode (480) the motion vector at 1/8-pel resolution as $MV = MVP + MVD$. In a variation, the decoder may get (460, 470) both MVP and MVD at 1/4-pel resolution, and decode (480) the motion vector at 1/4-pel resolution as $MV = MVP + MVD$.

[66] In the following, we describe several embodiments that can be used to perform motion vector refinement (450) in further detail.

[67] To refine the motion vector at the decoder, we use the characteristics that a picture signal usually is smooth and continuous. Thus, at the decoder side, if a motion vector is accurate, a decoded block typically should be continuous with respect to the neighboring blocks. In one embodiment, we use the reconstructed neighboring samples to refine the MV. In particular, a set of search candidates as described above are evaluated by measuring the discontinuity between the predicted block built by motion compensation with motion refinement candidates and the previously reconstructed samples, for example, the samples of the neighboring upper and left blocks as illustrated in FIG. 6A.

[68] FIG. 6A illustrates an exemplary PU (650) to be decoded. Other PUs (610, 620, 630, 640) above, or to the left of, the current PU are already decoded, and are available for decoding the current PU. In particular, an L-shape set of decoded samples (670) in neighboring blocks (610, 620, 640) may be used for refining the motion vector for the current PU (650). It should be noted that PUs can be in different sizes or shapes from what are shown in FIG. 6A, and a larger or smaller set of neighboring reconstructed samples can be used for refining motion vector for the current PU. In the present application, we use an L-shape set of samples for motion refinement. More generally, different sets of samples can be used for motion refinement, and the refinement can be applied to a block.

[69] In one embodiment, we use the discontinuity based on the L-shape set of decoded samples (670) in neighboring blocks (referred to as "neighboring L-shape") and an L-shape set of samples (680) in a current prediction block. The discontinuity can be measured as the Sum of Absolute Difference (SAD) between the reconstructed samples (n) and the closest motion-refinement prediction sample (p), as illustrated in FIG. 6B. Mathematically, the refined motion vector difference can be calculated as:

$$MVD^{*}_{refine} = \underset{MVD_{refine}}{\operatorname{argmin}} \sum_{p} |I_{ref}(p + MVP + MVD + MVD_{refine}) - I_{rec}(n)| \quad (4)$$

where p is a motion-refinement prediction sample in the L-shape of the PU with a location at (x_p, y_p), $\sum_p|\ \ |$ is a summation over the L-shape set of the PU, $MV_0 = MVP + MVD$ is the motion vector to be refined, n is the reconstructed sample in the L-shape next to p with a location at (x_n, y_n) (for example $x_n = x_p - 1, y_n = y_p$, if p belongs to the left bound of the PU, and $x_n = x_p$, $y_n = y_p - 1$, if p belongs to the upper bound of the PU), $I_{rec}(n)$ is the reconstructed (or decoded) sample value of the current picture, $I_{ref}(p + MVP + MVD + MVD_{refine})$ is the motion-compensated prediction value when $MVD_{refined}$ is selected.

[70] In a variation, we consider the sum of the residual and the motion-compensated prediction block, and $I_{ref}(p + MV_0 + MVD_{refine}) + Res(p)$ would be the reconstructed value for sample p if MVD_{refine} is selected. Then the refined motion vector difference can be calculated as:

$$MVD^*_{refine} = \underset{MVD_{refine}}{\operatorname{argmin}} \sum_p |I_{ref}(p + MVP + MVD + MVD_{refine}) + Res(p) - I_{rec}(n)| \quad (5)$$

where $Res(p)$ is the residual at sample p.

[71] When an HEVC decoder is modified to include the motion vector refinement according to the present principles, MV_0 can be set to MVD+MVP in AMVP or the one signaled in the candidate list in merge mode, wherein MVD and MVP are obtained as specified by HEVC. Generally, MV_0 is the motion vector the decoder obtained without performing motion search at the decoder.

[72] Motion refinement can be viewed as motion search at the decoder, with a set of candidate motion vectors, at a higher motion resolution, selected from around the initial motion vector. The choice of a best motion vector may be the one that minimizes a certain criterion, for example, the discontinuity measure as described in Eq. (4) or (5). That is, after an initial motion vector MV_0 is obtained, a motion search is further performed to refine the initial motion vector. The extra complexity in a decoder is usually small because only a small set of search candidates around the initial motion vector needs to be checked.

[73] Mathematically, the derived refined MV can be expressed as:

$$MV = MV_0 + MVD^*_{refine} \quad (6)$$

Then the block corresponding to the refined MV in the reference picture is used as the prediction block for decoding the PU, for example, using the interpolation filters as described above. Typically, motion refinement enables an encoder to encode an MVD at a low

resolution, and thus reduces the cost of encoding motion information compared with encoding MVD at a full resolution, while the decoder can still recover the MV at a full resolution.

[74] In the present application, we use SAD as a difference measure in various embodiments. It should be noted that other difference measures, for example, but not limited to, Sum of Squared Error (SSE), can be used instead of SAD.

[75] In another embodiment, we use the property that gradients at adjacent pixels are usually similar, and we compute local gradients at locations (n) and (p) to measure the discontinuity. Assuming that the signal (i.e., picture samples) is spatially stationary, one can locally model the signal as a Taylor series, truncated to the linear term:

$$I(x+d) = I(x) + g(x).d \quad (7)$$

where I(x) is the picture sample value at location x, g is a 2x2 matrix estimated with local gradients.

[76] Using Eq. (7), the relation between the neighboring reconstructed samples R(x) and the motion-refinement predicted block P(x) becomes:

$$R(n) = P(p) + g(p).(n-p) \quad (8)$$

with $P(p) = I_{ref}(p + MV_0 + MVD_{refine})$, and $R(n) = I_{rec}(n)$. In a variation, the residuals can be included when calculating P(p). The local gradients at position $p = (x_p, y_p)$ can be expressed as shown in FIG. 7A:

$G11(p) = R(n) - P(p)$, with $n=(x_p - 1, y_p - 1)$

$G12(p) = R(n) - P(p)$, with $n=(x_p - 1, y_p)$

$G13(p) = R(n) - P(p)$, with $n=(x_p - 1, y_p + 1)$

[77] Then the sum of the absolute value of the local gradients can be used to calculate the discontinuity, and the motion vector refinement that minimizes the discontinuity is chosen:

$$MVD^{*}_{refine} = \underset{MVD_{refine}}{\operatorname{argmin}} \sum_p \{|G11(p)| + |G12(p)| + |G13(p)|\} \quad (9)$$

[78] The second order moments at position $p = (x_p, y_p)$ can be expressed as shown in FIG. 7B (when n and p are located at the vertical left boundary of the L-shape):

$G21(p) = R(n2) – R(n) – G11$, with $n=(x_p -1, y_p -1)$ and $n2=(x_p - 2, y_p - 2)$

$G22(p) = R(n2) – R(n) – G12$, with $n=(x_p -1, y_p)$ and $n2=(x_p - 2, y_p)$

$G23(p) = R(n2) – R(n) – G13$, with $n=(x_p -1, y_p + 1)$ and $n2=(x_p -2, y_p + 2)$

$G24(p) = R(n2) - R(n) - P(p) + P(p2)$, with $n=(x_p - 1, y_p)$ and $n2=(x_p - 1, y_p + 1)$ and $p2=(x_p, y_p + 1)$.

When sample p is at the upper boundary of the PU, the gradients can be derived similarly. For the upper-left corner, it can be processed as it is at the left or upper boundary or can be processed twice (once as the boundary, and once as the upper).

[79] Then the sum of the absolute value of the second order moments of the gradients can be used to calculate the discontinuity, and the motion vector refinement that minimizes the discontinuity is chosen:

$$MVD^{*}_{refine} = \underset{MVD_{refine}}{\operatorname{argmin}} \sum_p \{|G21(p)| + |G22(p)| + |G23(p)| + |G24(p)|\} \quad (10)$$

[80] The discontinuity can also be calculated as the sum of the absolute value of the local gradients and second order moments:

$$MVD^{*}_{refine} = \underset{MVD_{refine}}{\operatorname{argmin}} \sum_p \{|G11(p)| + |G12(p)| + |G13(p)| + |G21(p)| + |G22(p)| + |G23(p)| + |G24(p)|\} \quad (11)$$

It should be noted that the present principles can also be applied to other forms of gradient calculation.

[81] FIG. 8 illustrates an exemplary method 800 for performing motion vector refinement at an encoder, according to an embodiment of the present principles. The output of method 800 may be used as an input bitstream to method 400. At the initialization step (805), the encoder may access a video sequence to be encoded as input. Additionally, the encoder may set the parameters to initial values, for example, set Best_flag = 1, and set Best_RDcost to a large value.

[82] Motion estimation (810) is performed at the 1/8-resolution to obtain a motion vector (MV). The encoder then checks whether it is more efficient to encode the motion vector using adaptive motion refinement or not. From steps 820-840, the encoder checks the MV encoding cost with motion refinement, for example, using the RD (Rate-Distortion) cost. From steps 845-875, the encoder checks the MV encoding cost without motion refinement. Then motion compensation is performed based on the final MV (Best_MV) and the residuals can be calculated (885). The residuals, the final MVD (Best_MVD) and the adaptive motion refinement flag (Best_flag) are encoded (890) into the bitstream.

[83] More particularly, at step 815, a motion vector predictor list is built at the 1/4-pel resolution, for example, using the motion vector predictor candidate list from AMVP, a rounded version of a 1/8-pel initial motion vector predictor or an average of neighboring motion vectors, consistent with how a corresponding decoder builds the motion vector predictor list. Lowering the resolution may make the motion vector predictor into a more "correlated" one (i.e., the motion vector predicted can be accurately predicted such that just an index may be transmitted to indicate the motion vector predictor), and let the motion refinement to obtain the high resolution part (i.e., the less "correlated" portion). The MVP list may contain only one MVP in some cases. For each MVP in the MVP list, a motion vector difference (MVD) is calculated (820) as MVD = MV – MVP, and an initial motion vector can be calculated as MV_0 = MVP + MVD.

[84] In a variation, the motion vector predictor candidate list may be built at 1/8-pel resolution at step 815, and the motion vector difference MVD is rounded to 1/4-pel at step 820. Note that MV_0 may be different from MV because of the rounding applied to the MVP or MVD.

[85] The refinement is performed (825) to obtain MVD_{refine}, for example, as described in Eq. (4) or (5), and a refined motion vector as $MV^* = MV_0 + MVD_{refine}$. The residuals can then be calculated (827) based on MV*. The encoding cost of the adjusted motion vector (MV^*) can be estimated using RD (Rate-Distortion) cost at step 830. At step 835, the encoder checks whether the current adjusted motion vector has a smaller RD cost than the current Best_RDcost. If yes, parameters Best_RDcost, Best_MV, Best_MVD are set to the current RD cost, current adjusted motion vector, and current MVD, and some other relevant encoding information may also be stored.

[86] At step 845, a motion vector predictor list is built at the 1/8-pel resolution, for example, using the motion vector predictor candidate list from AMVP. Step 845 is similar to step 815, except that the encoder does not adjust the motion resolution to 1/4-pel. For each MVP in the MVP list, the MVD is calculated (850) based on the MVP at the 1/8-pel resolution. The residuals can be calculated (852) for the motion vector MV_1 = MVP + MVD and the encoding cost of the motion vector MV_1 can be estimated using RD cost at step 855. At step 865, the encoder checks whether the current motion vector has a smaller RD cost than the current Best_RDcost. If yes, parameters Best_RDcost, Best_MV and Best_MVD are set to the current RD cost, current motion vector, and current MVD, and the

adaptive motion refinement flag is set to false (0). Other relevant encoding information may also be stored.

[87] When MVD is 0, for example, when the merge mode is used, steps 820 and 850 are not needed, and at step 840 and 875, Best_MVD does not need not to be calculated, and at step 890, Best_MVD does not need to be encoded.

[88] The motion refinement can be applied to all motion vectors corresponding to the motion vector predictors, or can be applied to a subset. For example, motion refinement is used for AMVP motion vector predictors only, or to the merge mode only.

[89] In method 800, the encoder decides whether to use adaptive motion refinement based on the encoding cost and signals the choice through a flag in the bitstream. In other embodiments, the encoder may decide whether to use adaptive motion refinement based on the video characteristics without checking the encoding cost, for example, as described further below or in Lakshman and Chen. Consequently, the encoder may only need to perform part of method 800 (for example, the part with motion refinement, or the part without motion refinement).

[90] Note that an additional syntax element Best_flag may need to be sent according to method 800. However, sending Best_flag may still be more efficient than sending the MVD at the 1/8-pel resolution since the high-resolution portion of the motion vector usually is random and expensive to encode.

[91] By refining the motion vector, the encoder can reach a 1/8-pel motion resolution while the motion vector difference MVD is transmitted in the bitstream at a 1/4-pel motion resolution. Accordingly, the MVD may require fewer bits to be encoded. For example, as shown in TABLE 1, a motion vector (MV) for a current block from the motion estimation (810) is 3.625, a motion vector predictor list (815) includes {3.0, ... }. For the motion vector predictor $MVP_0 = 3.0$, when motion refinement is not enabled, MVD (850) is MV – MVP = 0.625 (coded at 1/8-pel). On the other hand, when motion refinement is enabled, MVD is rounded to 0.5 (1/4-pel) and the initial motion vector (820) $MV_0 = MVP + MVD = 3.5$. The motion vector MV_0 then is refined (825) to $MV^* = 3.625$. In this example, the refined motion is the same as the MV obtained from motion estimation and the MVD is transmitted at 1/4-pel, and thus may need fewer bits than when no motion refinement is used (i.e., the MVD is transmitted at 1/8-pel). Consequently, the encoder is likely to choose to

enable motion refinement.

TABLE 1

	Resolution	Example
MVP list (left, upper)	1/4	{3.0…}
MVP	1/4	3.0
MVD	1/4	0.5
MV*	1/8	3.625
MV from ME	1/8	3.625
MV for MC	1/8	3.625

[92] In the above example, the refined motion vector is the same as the motion vector from motion estimation. It should be noted the refined motion vector may be different from the motion vector obtained from motion estimation.

[93] MVP refinement

[94] In the above, we discuss refinement to an initial motion vector that is generated based on a motion vector predictor MVP and a motion vector difference MVD. In another embodiment, the refinement can be performed with respect to the motion vector predictor (without including MVD). Referring back to FIG. 6, after an initial motion vector predictor (MVP_0) is obtained, for example, using AMVP as specified by HEVC, the motion vector predictor can be refined using the reconstructed L-shape (670).

[95] In particular, the decoder may form an MC L-shape (680) corresponding to a motion vector predictor around the initial motion vector predictor, for example, $MVP_0 + \Delta MV, \Delta MV = \{\left(-\frac{1}{8}, 0\right), \ldots, (0, \frac{1}{8})\}$. Then the decoder can compare a difference, for example, the discontinuity or gradients as discussed above, between the reconstructed L-shape (670) and different MC L-shapes (680), and choose the motion refinement (ΔMV^*) that yields the smallest difference as the refinement to the initial motion vector predictor, that is, the refined motion vector predictor can be calculated as $MVP^* = MVP_0 + \Delta MV^*$. Note that both the reconstructed L-shape (670) and MC L-shape (680) are based on decoded samples, such that the same process can be performed at both the encoder and decoder sides.

[96] Generally, the motion refinement for the motion vector predictor can be viewed as a motion search with a search range including a few sub-sample displacements at a full motion resolution around the initial motion vector predictor. For example, a motion vector predictor MVP_0 at 510 may be refined using the search candidates within 520 as illustrated in

FIG. 5. Similarly to refining the motion vector (with MVD), different sets of search candidates may be used from what is shown in FIG. 5.

[97] In a variation, the motion refinement can be performed on a rounded version of the initial motion vector predictor ($\mathrm{round}(\mathrm{MVP}_0)$), and the search range for the motion refinement includes sub-sample displacements at a full motion resolution around the rounded motion vector predictor. For example, when the refined motion resolution is 1/8 pel, the rounded version of MVP_0 may be at an interger-pel, half-pel or quarter-pel resolution. This is because the actual motion vector is more likely to be concentrated around a motion vector at the lower resolution, and the rounded version of the initial motion vector predictor may provide a better starting point for search.

[98] In another variation, the motion refinement may be performed with respect to an average value of neighboring motion vectors that are used in the MVP candidate list. For example, the motion refinement may use an average of the left and above motion vectors $(\mathrm{MV}_{\mathrm{left}} + \mathrm{MV}_{\mathrm{above}})/2$ as the initial motion vector predictor, and then perform a motion search around the initial motion vector predictor. We observe that motion refinement is usually selected in an almost, or relatively, uniform motion area. By averaging the motion vectors adjacent to the current block, we may provide a better starting point for search. When motion refinement is selected, i.e., when the motion field is uniform, the predictor candidate list may be reduced to use only one (left or above) candidate in the list, and thus, reducing the number of possible candidates. This may improve the compression efficiency as no index of the AMVP list needs to be encoded or transmitted in the bitstream.

[99] FIG. 9 illustrates an exemplary method 900 for performing motion vector predictor refinement at a decoder, according to an embodiment of the present principles. In this example, we suppose a decoder can refine a motion vector predictor resolution from 1/4-pel to 1/8-pel.

[100] For a block, the decoder checks (910) whether adaptive motion vector resolution is enabled, for example, using a flag in the bitstream, or using the existing methods as described above. If adaptive motion vector resolution is enabled, the decoder generates MV predictor (MVP) at 1/4-pel resolution (920). The initial motion vector predictor can be generated, for example, but not limited to, using the AMVP mode of HEVC, using an average of neighboring motion vectors, or using a rounded version of a motion vector predictor. The motion vector predictor can then be refined to 1/8-pel resolution (930), for example, using the

embodiments described above. The decoder decodes MVD at 1/8-pel resolution (940). The motion vector can be then obtained (950) as $MV = MVP^* + MVD$ and be used for motion-compensated prediction.

[101] If adaptive motion vector resolution is not enabled, the decoder obtains MVP and MVD at a motion resolution that is used for the decoded motion vector, and no motion refinement is performed at the decoder. As shown in FIG. 9, the decoder may get (960, 970) both MVP and MVD at 1/8-pel resolution, and decode (980) the motion vector at 1/8-pel resolution as $MV = MVP + MVD$. In a variation, the decoder may get (960, 970) both MVP and MVD at 1/4-pel resolution, and decode (980) the motion vector at 1/4-pel resolution as $MV = MVP + MVD$.

[102] FIG. 10 illustrates an exemplary method 1000 for performing motion vector predictor refinement at an encoder, according to an embodiment of the present principles. The output of method 1000 may be used as an input bitstream to method 900. At the initialization step (1005), the encoder may access a video sequence to be encoded as input. Additionally, the encoder may set the parameters to initial values, for example, set Best_flag = 1, and set Best_RDcost to a large value.

[103] Motion estimation (1010) is performed in the 1/8 resolution to obtain a motion vector MV. The encoder then checks whether it is more efficient to encode the motion vector using adaptive motion refinement or not. From steps 1020-1040, the encoder checks the MV encoding cost with motion refinement, for example, using the RD (Rate-Distortion) cost. From steps 1045-1075, the encoder checks the MV encoding cost without motion refinement. Then motion compensation is performed based on the final MV (Best_MV) and the residuals can be calculated (1085). The residuals, the final MVD (Best_MVD) and the adaptive motion refinement flag (Best_flag) are encoded (1090) into the bitstream.

[104] More particularly, at step 1015, a motion vector predictor list is built at the 1/4-pel resolution, for example, using the motion vector predictor candidate list from AMVP, an rounded version (for example, to the closet integer at the desired resolution) of an initial motion vector predictor or an average of neighboring motion vectors, consistent with how a corresponding decoder builds the motion vector predictor list. The MVP list may contain only one MVP in some cases. At step 1015, the motion vector predictor list can also be built at the 1/8-pel resolution. However, one advantage of using a lower resolution (1/4-pel)

is that it may avoid drift in flat areas, where the RD optimization cannot differentiate between MV at the 1/8-pel or 1/4-pel resolution. Using a reduced resolution allows a sort of smoothing of the MV predictor which may reduce the MV noise. Also, if the size of the search window for the refinement is relatively small (±1/4 for 1/8 refinement), the precision of the starting/center MV for the search window may change the result.

[105] For each MVP in the MVP list, MVP refinement is performed (1020), for example, as described above for the decoder. The MVD is calculated based on the refined MVP (MVP^*) (1025), and the adjusted motion vector (MV^*) that can be used for motion compensation ($MV^* = MVD + MVP^*$) can also be calculated. The encoding cost of the adjusted motion vector (MV^*) can be estimated using RD cost at step 1030. At step 1035, the encoder checks whether the current adjusted motion vector has a smaller RD cost than the current Best_RDcost. If yes, parameters Best_RDcost, Best_MV and Best_MVD are set (1040) to the current RD cost, current adjusted motion vector, and current MVD, and Best_flag remains true (1).

[106] At step 1045, a motion vector predictor list is built at the 1/8-pel resolution, for example, using the motion vector predictor candidate list from AMVP. For each MVP in the MVP list, the MVD is calculated (1050) based on the MVP at the 1/8-pel resolution. The encoding cost of the motion vector (MV) can be estimated using RD cost at step 1055. At step 1065, the encoder checks whether the current motion vector has a smaller RD cost than the current Best_RDcost. If yes, parameters Best_RDcost, Best_MV and Best_MVD are set (1075) to the current RD cost, current motion vector, and current MVD, and the adaptive motion refinement flag (Best_flag) is set to false (0).

[107] In method 1000, the encoder decides whether to use adaptive motion refinement based on the encoding cost and signals the choice through a flag in the bitstream. In other embodiments, the encoder may decide whether to use adaptive motion refinement based on the video characteristics without checking the encoding cost, for example, as described further below or in Lakshman and Chen. Consequently, the encoder may only need to perform part of method 1000 (for example, the part with motion refinement, or the part without motion refinement).

[108] By refining the motion vector predictor, the MV predictor can be more accurate. Accordingly, the MVD may statistically have lower values, and the coding cost can then be

reduced. For example, as shown in TABLE 2, a motion vector (MV) for a current block from the motion estimation (1010) is 3.625, a motion vector predictor list (1015) includes {3.0, … }. For the motion vector predictor MVP_0 = 3.0, it is refined (1020) by motion refinement to MVP* = 3.5. Subsequently, the motion vector difference is MVD = MV – MVP* = 0.125. Comparing with the motion vector difference without using motion refinement MV – MVP_0 = 0.625, the MVD to be encoded is smaller and may need fewer bits to be encoded. In this example, using motion refinement may improve the compression efficiency, and thus, the encoder is likely to choose to enable adaptive motion refinement. The motion vector used for motion compensation is MVP* + MVD = 3.625.

TABLE 2

	Resolution	Example
MV from ME	1/8	3.625
MVP list (left, upper)	1/4	{3.0…}
MVP	1/4	3.0
MVP-refined (MVP*)	1/8	3.5
MVD	1/8	0.125
MV for MC	1/8	3.625

[109] When MVD is 0, for example, when the merge mode is used, steps 1025 and 1050 are not needed, and at step 1040 and 1075, Best_MVD does not need to be calculated, and at step 1090, Best_MVD does not need to be encoded. Note that when MVD is 0, method 1000 may become the same as method 800. The motion refinement can be applied to all motion vectors corresponding to the motion vector predictors, or can be applied to a subset. For example, motion refinement is used for AMVP motion vector predictors only, or to the merge mode only.

[110] Automatic Motion Refinement Activation

[111] In the above embodiments, we describe that a flag (for example, Best_flag as shown in FIG. 8 or FIG. 10) can be used to indicate whether adaptive motion vector resolution is enabled. In the following, we describe various embodiments that automatically activate or deactivate motion vector resolution refinement. Thus, the encoder can indicate whether adaptive motion vector resolution is enabled without an explicit flag, and the decoder can also decide whether adaptive motion vector resolution without referring to an explicit flag.

[112] In one embodiment, we propose to automatically activate the adaptive motion refinement based on one or more of the following criteria:

- Motion similarity: The motion of the current block is similar to the motion of surrounding blocks, such that an encoder or decoder can use motion vectors of surrounding blocks for motion refinement.
- Textureness (or texture level): The current block and surrounding blocks contain some texture which may allow a robust sub-pixel motion refinement.
- Motion vector difference (MVD): MVD is available at both the encoder and decoder, and can be used to determine whether to use automatic adaptive motion refinement or to explicitly signal the adaptive motion refinement.

[113] The motion similarity criterion may be measured using one or more of the following conditions:

1) MVP_{left} and MVP_{above} (optionally $MVP_{above\text{-}left}$) exist, namely, a motion field exists around the current block. More generally, there are one or more neighboring blocks that have motion vectors.

2) $|MVP_{left} - MVP_{current}| < T2$, $|MVP_{above} - MVP_{current}| < T2$, and $|MVP_{leftabove} - MVP_{current}| < T2$. This condition can also be a different logical combination of these three sub-conditions: $|MVP_{left} - MVP_{current}| < T2$, $|MVP_{above} - MVP_{current}| < T2$, and $|MVP_{leftabove} - MVP_{current}| < T2$. That is, the motion vectors of the surrounding blocks are similar and motion field around the current block is somewhat uniform. Here we use the same T2 to check the difference between MVP_{left} and $MVP_{current}$, between MVP_{above} and $MVP_{current}$, and between $MVP_{leftabove}$ and $MVP_{current}$. It should be noted that different thresholds can be used for these differences.

[114] The textureness criterion may be measured using one or more of the following conditions:

1) Texture(L) > T3, where L is the neighboring area used for performing motion refinement, for example, 670 as shown in FIG. 6A, and texture(X) is a measure of texturing, for example, the variance in luminance of X. That is, the neighboring area has some texture.

2) Err(mc(L, MVP, Iref), L) < Err(mc(L, MVP+MVr, Iref),L) + T4, where MVP is the selected motion vector predictor that corresponds to MVD signaled in the bitstream, such that both the encoder and decoder can use the same MVP for checking this condition, Err(X,Y) is a measure of error between a group of pixels X and a group of pixels Y, for example, SAD, HAD or SSE, mc(X, v, I) is the motion compensation of the group of pixels X using the motion vector v in the reference image I, Iref is the

reference image associated with the motion vector predictor MVP, T4 is a threshold on error similarity, MVP+MVr is the motion vector predictor after refinement. The MVP may be at a lower motion resolution as described before, for example, being integer rounded.

[115] The automatic motion refinement activation may be based on all conditions under both motion similarity or textureness criteria, or may also use a subset of the conditions. Whether to signal the motion refinement activation or deactivation can be based on the motion vector difference (MVD). When $|MVD| \leq T1$, wherein MVD is the motion vector difference indicated in the bitstream and T1 is a motion threshold, in a typical example, $T1 = \sqrt{2}/2$, we may choose to activate motion vector refinement without explicit signalling. That is, when the motion vector difference is small and the current motion vector is close to the current motion vector predictor, the encoder and decoder could automatically activate motion refinement. In addition, both the encoder and decoder have access to MVD, and can use MVD to determine the motion refinement activation in the same manner in order to synchronize the encoder and decoder. On the other hand, when |MVD| becomes large, the encoder and decoder may automatically deactivate motion refinement, without explicit signalling.

[116] In another example, when a temporal candidate is used to build the motion vector predictor list, that is, when MVP_{left} and MVP_{above} do not co-exist or are the same, conditions 1) and 2) in the motion similarity criterion are not used. The encoder may choose whether or not to use the temporal candidate for automatic refinement.

[117] FIG. 11 illustrates an exemplary method 1100 for performing motion vector predictor refinement with automatic motion refinement activation or deactivation at a decoder, according to an embodiment of the present principles.

[118] For a block to be decoded, the decoder generates (1110) MV predictor ($MVP_{current}$) and decodes (1120) the MV difference (MVD). When $|MVD| \leq T1$ (1125) is true, the decoder checks other conditions to see whether motion refinement is to be activated. Otherwise, if $|MVD| > T1$, the decoder checks whether $|MVD| \leq T1 + S * \sqrt{2}/2$. In one example, $S = 1/2$, indicating that the motion search for the refinement is in [-1/2, 1/2]. If $|MVD| \leq T1 + S * \sqrt{2}/2$ (1130), the decoder decides whether motion refinement is used based on a flag decoded (1140) from the bitstream. Otherwise, if $|MVD| > T1 + S * \sqrt{2}/2$,

the decoder does not perform motion refinement (i.e., the motion refinement is automatically deactivated) and decodes (1180) the motion vector based on the MVP and MVD, i.e., MV = MVP + MVD.

[119] More particularly, the decoder checks (1155) whether $|MVP_{left} - MVP_{current}| < T2$ and $|MVP_{above} - MVP_{current}| < T2$. If yes, the decoder checks (1165) whether Texture(L) > T3. If both conditions at 1155 and 1165 are satisfied, the decoder performs motion vector predictor refinement, for example, using the various embodiments described above, and decodes (1170) the motion vector based on the refined motion vector predictor (MVP*) and MVD, i.e., MV = MVP* + MVD. Here, the motion refinement is automatically activated without a flag.

[120] When $T1 < |MVD| \leq T1 + S * \sqrt{2}/2$, the decoder decodes (1140) a flag (for example, Refine_flag) indicating whether motion refinement may be used. If the flag is true (1145), the decoder continues to step 1155. Otherwise, if the flag is false (1145), the motion vector is obtained (1180) based on the MVP and MVD.

[121] In FIG. 11, the decoder automatically deactivates or activates motion refinement in some conditions, and relies on a flag to deactivate or activate motion refinement in other conditions. The same conditions are used by a corresponding encoder to ensure the synchronization between the encoder and decoder.

[122] When MVD is 0, for example, when the merge mode is used, the condition at step 1125 is always true and can be removed, and steps 1120, 1130, 1140, 1145 and 1180 are not needed. Alternatively, in the merge mode, motion refinement can be disabled in order to avoid adding noise on the merge deduced motion vector predictor.

[123] Different from what is shown in FIG. 11, method 1100 can also skip steps 1130-1145. That is, when |MVD| > T1, the decoder performs step 1180 without motion refinement. Other variations can also be implemented, for example, but not limited to, skipping step 1155 and/or step 1165, checking one of the two conditions at step 1155.

[124] FIG. 12 illustrates an exemplary method 1200 for performing automatic motion refinement at an encoder, according to an embodiment of the present principles. The output of method 1200 may be used as an input bitstream to method 1100. At the initialization step (1205), the encoder may access a video sequence to be encoded as input. Additionally, the encoder may set the parameters to initial values, for example, set Best_RDcost to a large

value.

[125] Motion estimation (1210) is performed at the 1/8-pel resolution to obtain a motion vector MV. At step 1220, a motion vector predictor list is built, for example, using the motion vector predictor candidate list from AMVP. A rounded version (for example, to the closet integer at the desired resolution) of an initial motion vector predictor or an average of neighboring motion vectors may be used, consistent with how a corresponding decoder builds the motion vector predictor list. The MVP list may contain only one MVP in some cases. At step 1220, the motion vector predictor list can also be built at other resolutions.

[126] At step 1225, for a particular MVP, the encoder then checks whether it is more efficient to encode the motion vector using adaptive motion refinement or not and sets a Refine_flag to 0 or 1. When motion refinement is more efficient, Refine_flag is set to 1, and otherwise to 0. A corresponding MVD is also computed at step 1225.

[127] FIG. 13 illustrates an exemplary method 1300 for determining whether or not to use adaptive motion refinement for a particular MVP, according to an embodiment of the present principles. Method 1300 can be used to implement step 1225 of method 1200.

[128] Particularly, the encoder checks (1305) whether $|MVP_{left} - MVP_{current}| < T2$, $|MVP_{above} - MVP_{current}| < T2$, and Texture(L) > T3. If the conditions at 1305 are satisfied, the encoder performs motion vector predictor refinement (1330), for example, using the various embodiments described above, and computes (1340) the motion vector difference based on the refined motion vector predictor (MVP*), i.e., MVD = MV - MVP*. A RD Cost (RDCost1) is estimated (1350) when motion refinement is used. The encoder also computes (1360) the motion vector difference without the refined motion vector predictor, i.e., MVD = MV - MVP. A RD Cost (RDCost2) is estimated (1370) when motion refinement is not used. The RD costs with and without motion refinement are compared (1375). If RDCost1 is smaller, then Refine_flag is set (1380) to 1, and MVD is set to MV – MVP*. Otherwise, Refine_flag is set (1390) to 0, and MVD is set to MV – MVP.

[129] If the conditions at 1305 are not satisfied, the motion vector difference is computed (1310) based on the motion vector predictor (MVP), i.e., MVD = MV – MVP, and Refine_flag is set (1320) to 0.

[130] After the MVD and Refine_flag are determined at step 1225, at step 1230, the encoder checks whether $|MVD| \leq T1$. If yes, the encoder further checks (1235) whether Refine_flag

is equal to 1. If Refine_flag is set to 1, the encoder estimates (1270) the RD cost. If the RD cost is smaller than the current Best_RDCost (1275), parameters Best_RDcost and Best_MVD are set (1280) to the current RD cost and current MVD, and Write_flag is set to no, indicating that no explicit signaling is used to indicate adaptive motion refinement. That is, if the current MVP is selected for encoding, the motion refinement would be automatically activated without the need to send a flag.

[131] When |MVD| > T1 (1230), the encoder estimates (1240) the RD cost. If the RD cost is smaller than the Best_RDCost (1245), parameters Best_RDcost and Best_MVD are set (1250) to the current RD cost and current MVD. The encoder then checks (1255) whether $|MVD| \leq T1 + S * \sqrt{2}/2$. If yes, Write_flag is set to yes. Here, if a motion vector predictor corresponding to conditional branch 1260 is selected, whether motion refinement is activated would be explicitly signaled based on Refine_flag.

[132] Otherwise if the condition at 1255 is not satisfied, Write_flag is set (1265) to no. That is, if a motion vector predictor corresponding to conditional branch 1265 is selected, the motion refinement would be automatically deactivated without the need to send a flag.

[133] The encoder checks (1285) whether the end of the MVP list is reached. If yes, the encoder encodes the Best_MVD, the index of the selected MVP, and corresponding residual if any. If Write_flag is set to yes, the Refine_flag is also encoded. Otherwise, if the condition at 1285 is not satisfied, the control is returned to step 1225. If the merge mode is selected, Best_MVD does not need to be encoded at step 1290. Corresponding to method 1100, method 1200 can also be varied from what is shown in FIG. 12, for example, steps 1240-1260 can be skipped.

[134] In the above, automatic activation of deactivation of motion refinement is discussed with respect to a motion vector predictor. It should be noted that the automatic activation or deactivation can be applied to other types of motion information, for example, but not limited to, the motion vector that already includes the motion vector difference.

[135] Various numeric values are used in the present application, for example, to determine the motion similarity or textureness based on thresholds. It should be noted that the specific values are for exemplary purposes and the present principles are not limited to these specific values.

[136] In the above, various embodiments are described with respect to the HEVC standard.

For example, various motion refinement or automatic activation methods as described above can be used to modify the motion estimation module (175) of the HEVC encoder as shown in FIG. 1 or the motion compensation module (375) of the HEVC decoder as shown in FIG. 3. However, the present principles are not limited to HEVC, and can be applied to other standards, recommendations, and extensions thereof.

[137] In the above, we discuss motion refinement from from1/4-pel to 1/8-pel. It should be noted that the present principles can be applied to refinement between other motion resolutions, for example, but not limited to, from integer-pel to 1/4-pel, from1/4-pel to 1/6-pel.

[138] FIG. 14 illustrates a block diagram of an exemplary system in which various aspects of the exemplary embodiments of the present principles may be implemented. System 1400 may be embodied as a device including the various components described below and is configured to perform the processes described above. Examples of such devices, include, but are not limited to, personal computers, laptop computers, smartphones, tablet computers, digital multimedia set top boxes, digital television receivers, personal video recording systems, connected home appliances, and servers. System 1400 may be communicatively coupled to other similar systems, and to a display via a communication channel as shown in FIG. 14 and as known by those skilled in the art to implement the exemplary video system described above.

[139] The system 1400 may include at least one processor 1410 configured to execute instructions loaded therein for implementing the various processes as discussed above. Processor 1410 may include embedded memory, input output interface and various other circuitries as known in the art. The system 1400 may also include at least one memory 1420 (e.g., a volatile memory device, a non-volatile memory device). System 1400 may additionally include a storage device 1440, which may include non-volatile memory, including, but not limited to, EEPROM, ROM, PROM, RAM, DRAM, SRAM, flash, magnetic disk drive, and/or optical disk drive. The storage device 1440 may comprise an internal storage device, an attached storage device and/or a network accessible storage device, as non-limiting examples. System 1400 may also include an encoder/decoder module 1430 configured to process data to provide an encoded video or decoded video.

[140] Encoder/decoder module 1430 represents the module(s) that may be included in a

device to perform the encoding and/or decoding functions. As is known, a device may include one or both of the encoding and decoding modules. Additionally, encoder/decoder module 1430 may be implemented as a separate element of system 1400 or may be incorporated within processors 1410 as a combination of hardware and software as known to those skilled in the art.

[141] Program code to be loaded onto processors 1410 to perform the various processes described hereinabove may be stored in storage device 1340 and subsequently loaded onto memory 1420 for execution by processors 1410. In accordance with the exemplary embodiments of the present principles, one or more of the processor(s) 1410, memory 1420, storage device 1440 and encoder/decoder module 1430 may store one or more of the various items during the performance of the processes discussed herein above, including, but not limited to the input video, the bitstream, equations, formula, matrices, variables, operations, and operational logic.

[142] The system 1400 may also include communication interface 1450 that enables communication with other devices via communication channel 1460. The communication interface 1450 may include, but is not limited to a transceiver configured to transmit and receive data from communication channel 1460. The communication interface may include, but is not limited to, a modem or network card and the communication channel may be implemented within a wired and/or wireless medium. The various components of system 1400 may be connected or communicatively coupled together using various suitable connections, including, but not limited to internal buses, wires, and printed circuit boards.

[143] The exemplary embodiments according to the present principles may be carried out by computer software implemented by the processor 1410 or by hardware, or by a combination of hardware and software. As a non-limiting example, the exemplary embodiments according to the present principles may be implemented by one or more integrated circuits. The memory 1420 may be of any type appropriate to the technical environment and may be implemented using any appropriate data storage technology, such as optical memory devices, magnetic memory devices, semiconductor-based memory devices, fixed memory and removable memory, as non-limiting examples. The processor 1410 may be of any type appropriate to the technical environment, and may encompass one or more of microprocessors, general purpose computers, special purpose computers and processors based on a multi-core architecture, as non-limiting examples.

[144] The implementations described herein may be implemented in, for example, a method or a process, an apparatus, a software program, a data stream, or a signal. Even if only discussed in the context of a single form of implementation (for example, discussed only as a method), the implementation of features discussed may also be implemented in other forms (for example, an apparatus or program). An apparatus may be implemented in, for example, appropriate hardware, software, and firmware. The methods may be implemented in, for example, an apparatus such as, for example, a processor, which refers to processing devices in general, including, for example, a computer, a microprocessor, an integrated circuit, or a programmable logic device. Processors also include communication devices, such as, for example, computers, cell phones, portable/personal digital assistants ("PDAs"), and other devices that facilitate communication of information between end-users.

[145] Reference to "one embodiment" or "an embodiment" or "one implementation" or "an implementation" of the present principles, as well as other variations thereof, mean that a particular feature, structure, characteristic, and so forth described in connection with the embodiment is included in at least one embodiment of the present principles. Thus, the appearances of the phrase "in one embodiment" or "in an embodiment" or "in one implementation" or "in an implementation", as well any other variations, appearing in various places throughout the specification are not necessarily all referring to the same embodiment.

[146] Additionally, this application or its claims may refer to "determining" various pieces of information. Determining the information may include one or more of, for example, estimating the information, calculating the information, predicting the information, or retrieving the information from memory.

[147] Further, this application or its claims may refer to "accessing" various pieces of information. Accessing the information may include one or more of, for example, receiving the information, retrieving the information (for example, from memory), storing the information, processing the information, transmitting the information, moving the information, copying the information, erasing the information, calculating the information, determining the information, predicting the information, or estimating the information.

[148] Additionally, this application or its claims may refer to "receiving" various pieces of information. Receiving is, as with "accessing", intended to be a broad term. Receiving the information may include one or more of, for example, accessing the information, or retrieving the information (for example, from memory). Further, "receiving" is typically involved, in

one way or another, during operations such as, for example, storing the information, processing the information, transmitting the information, moving the information, copying the information, erasing the information, calculating the information, determining the information, predicting the information, or estimating the information.

[149] As will be evident to one of skill in the art, implementations may produce a variety of signals formatted to carry information that may be, for example, stored or transmitted. The information may include, for example, instructions for performing a method, or data produced by one of the described implementations. For example, a signal may be formatted to carry the bitstream of a described embodiment. Such a signal may be formatted, for example, as an electromagnetic wave (for example, using a radio frequency portion of spectrum) or as a baseband signal. The formatting may include, for example, encoding a data stream and modulating a carrier with the encoded data stream. The information that the signal carries may be, for example, analog or digital information. The signal may be transmitted over a variety of different wired or wireless links, as is known. The signal may be stored on a processor-readable medium.

CLAIMS

1. A method (1100) for video decoding, comprising:

accessing (1110) a motion vector predictor and a motion vector difference for a current block of a video, said motion vector predictor being associated with a first motion resolution; and

determining (1125, 1130) whether or not to refine said motion vector predictor, based on said motion vector difference, wherein refining (1170) said motion vector predictor includes:

forming a refined motion vector predictor based on motion search, said refined motion vector predictor being associated with a second motion resolution, and said second motion resolution being higher than said first motion resolution, and

forming (1170) a motion vector for said current block based on said refined motion vector predictor and said motion vector difference, wherein said current block is decoded based on said formed motion vector.

2. An apparatus (1400) comprising at least one memory and one or more processors, said one or more processors configured to:

access a motion vector predictor and a motion vector difference for a current block of a video, said motion vector predictor being associated with a first motion resolution; and

determine whether or not to refine said motion vector predictor, based on said motion vector difference, wherein said one or more processors are configured to refine said motion vector predictor by performing:

forming a refined motion vector predictor based on motion search, said refined motion vector predictor being associated with a second motion resolution, and said second motion resolution being higher than said first motion resolution, and

forming a motion vector for said current block based on said refined motion vector predictor and said motion vector difference, wherein said current block is decoded based on said formed motion vector.

3. The method of claim 1, or the apparatus of claim 2, wherein said motion vector predictor is determined to be refined when a magnitude of said motion vector difference is smaller than a first threshold.

4. The method of claim 1 or 3, or the apparatus of claim 2 or 3, wherein said motion vector predictor is determined not to be refined when a magnitude of said motion vector difference exceeds a second threshold, further comprising decoding said current block based on said motion vector predictor and said motion vector difference.

5. The method of any one of claims 1 and 3-4, or the apparatus of any one of claims 2-4, when a magnitude of said motion vector difference exceeds a first threshold, further comprising decoding a flag from a bitstream, wherein whether or not said motion vector predictor is to be refined is based on said decoded flag.

6. The method of any one of claims 1 and 3-5, further comprising, or the apparatus of any one of claims 2-5, said one or more processors further configured to perform:

accessing at least one motion vector of adjacent decoded blocks, wherein said motion vector predictor is determined to be refined if a difference between said motion vector predictor for said current block and said at least one motion vector is smaller than a third threshold.

7. The method of any one of claims 1 and 3-6, further comprising, or the apparatus of any one of claims 2-6, said one or more processors further configured to perform:

accessing a plurality of pixels of adjacent decoded blocks, wherein said motion vector predictor is determined to be refined if a texture level of said plurality of pixels exceeds a fourth threshold.

8. A method (1200) for video encoding, comprising:

accessing (1220) a motion vector predictor, said motion vector predictor being associated with a first motion resolution;

determining (1225) a motion vector difference corresponding to said motion vector predictor;

determining (1225) whether to refine said motion vector predictor, said refinement being associated with a second motion resolution, and said second motion resolution being higher than said first motion resolution;

determining (1280, 1260, 1265) whether or not to explicitly signal said refinement of said motion vector predictor, based on said determined motion vector difference; and

encoding (1290) said motion vector difference.

9. An apparatus (1400) comprising at least one memory and one or more processors, said one or more processors configured to:

access a motion vector predictor, said motion vector predictor being associated with a first motion resolution;

determine a motion vector difference corresponding to said motion vector predictor;

determine whether to refine said motion vector predictor, said refinement being associated with a second motion resolution, and said second motion resolution being higher than said first motion resolution;

determine whether or not to explicitly signal said refinement of said motion vector predictor, based on said determined motion vector difference; and

encode said motion vector difference.

10. The method of claim 8, or the apparatus of claim 9, wherein said refinement of said motion vector predictor is not explicitly signaled when a magnitude of said motion vector difference is smaller than a first threshold.

11. The method of claim 8 or 10, or the apparatus of claim 9 or 10, wherein said refinement of said motion vector predictor is deactivated without explicitly signaling when a magnitude of said motion vector difference exceeds a second threshold.

12. The method of any of claims 8 and 10-11, or the apparatus of any one of claims 9-11, when a magnitude of said motion vector difference exceeds a first threshold, further comprising encoding a flag into a bitstream to explicitly signal whether or not said motion vector predictor is to be refined.

13. The method of any of claims 8 and 10-12, further comprising, or the apparatus of any of claims 9-12, said one or more processors further configured to perform:

accessing at least one motion vector of adjacent reconstructed blocks, wherein said motion vector predictor is determined to be refined if a difference between said motion vector predictor for said current block and said at least one motion vector is smaller than a third

threshold.

14. The method of any of claims 8 and 10-13, further comprising, or the apparatus of any of claims 9-13, said one or more processors further configured to perform:

accessing a plurality of pixels of adjacent reconstructed blocks, wherein said motion vector predictor is determined to be refined if a texture level of said plurality of pixels exceeds a fourth threshold.

15. A non-transitory computer readable storage medium having stored thereon instructions for implementing a method according to any of claims 1, 3-8 and 10-14.

16. A bitstream, generated according to a method of any of claims 8 and 10-14.

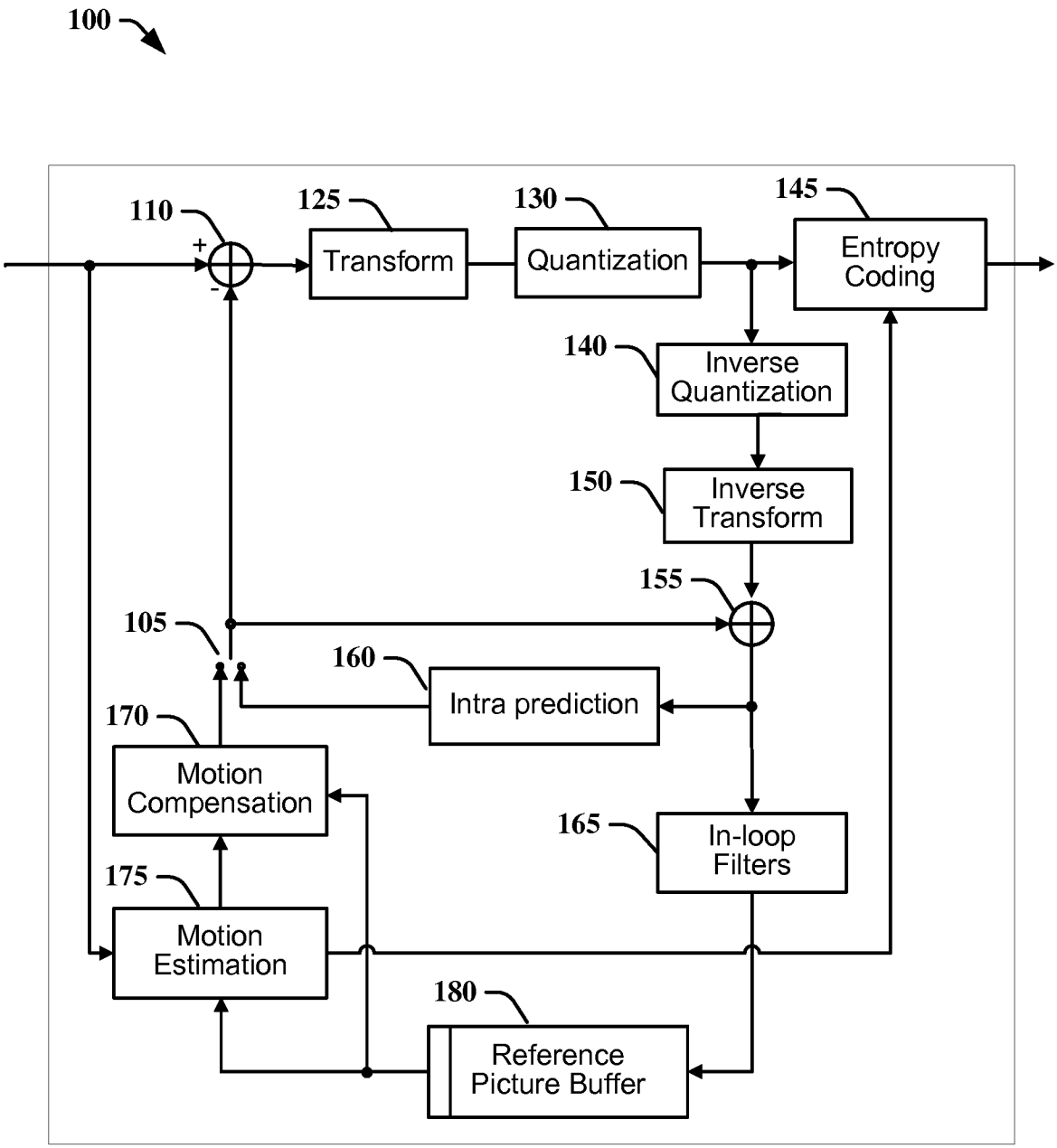
100
110
+
-
125
Transform
130
Quantization
145
Entropy Coding
140
Inverse Quantization
150
Inverse Transform
155
105
160
Intra prediction
170
Motion Compensation
165
In-loop Filters
175
Motion Estimation
180
Reference Picture Buffer

FIG. 1

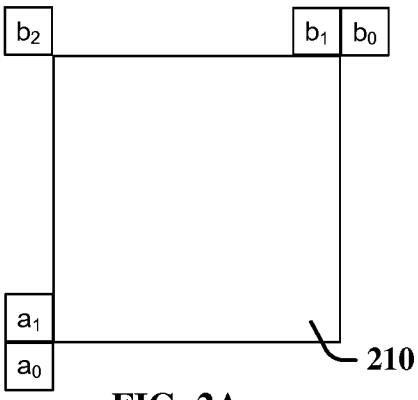

FIG. 2A

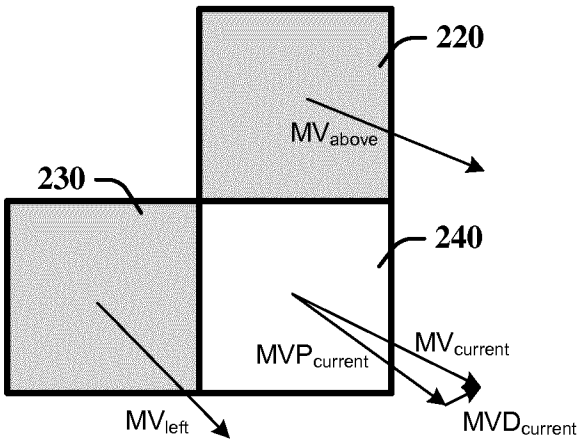

FIG. 2B

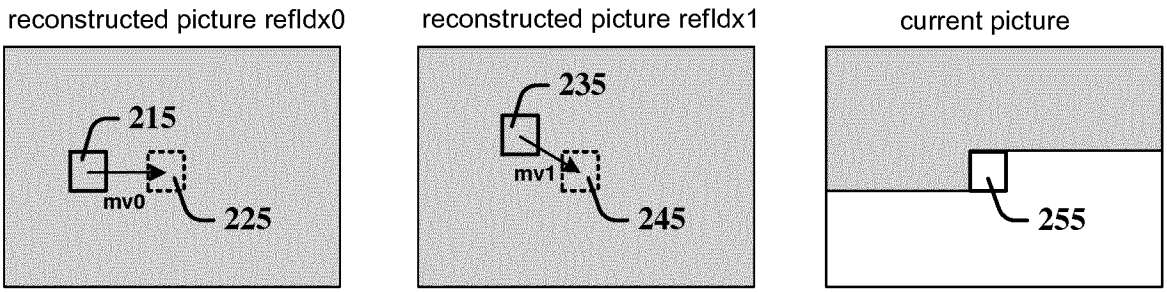

FIG. 2C

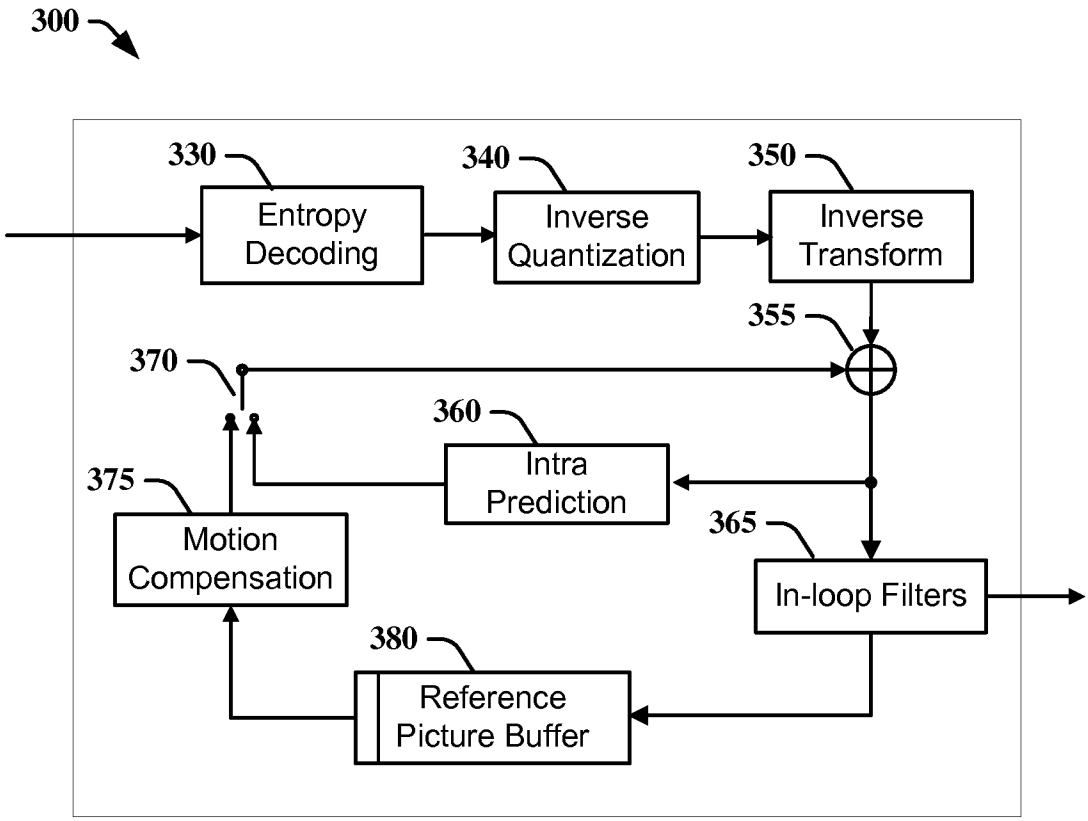
300
330
Entropy Decoding
340
Inverse Quantization
350
Inverse Transform
355
370
360
Intra Prediction
375
Motion Compensation
365
In-loop Filters
380
Reference Picture Buffer

FIG. 3

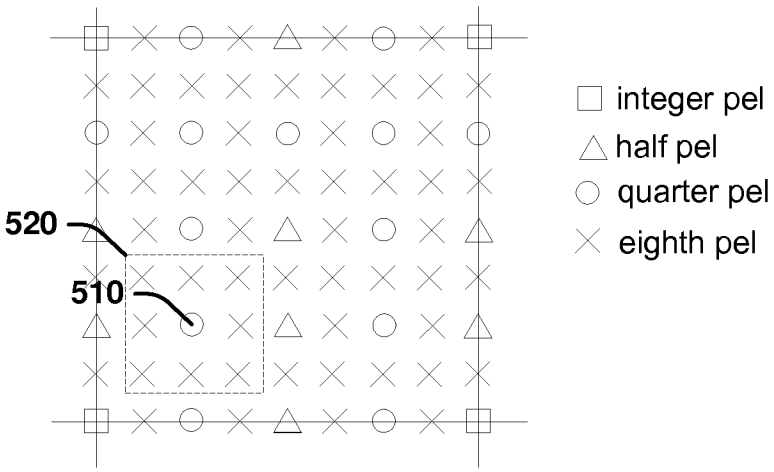
520
510
integer pel
half pel
quarter pel
eighth pel

FIG. 5

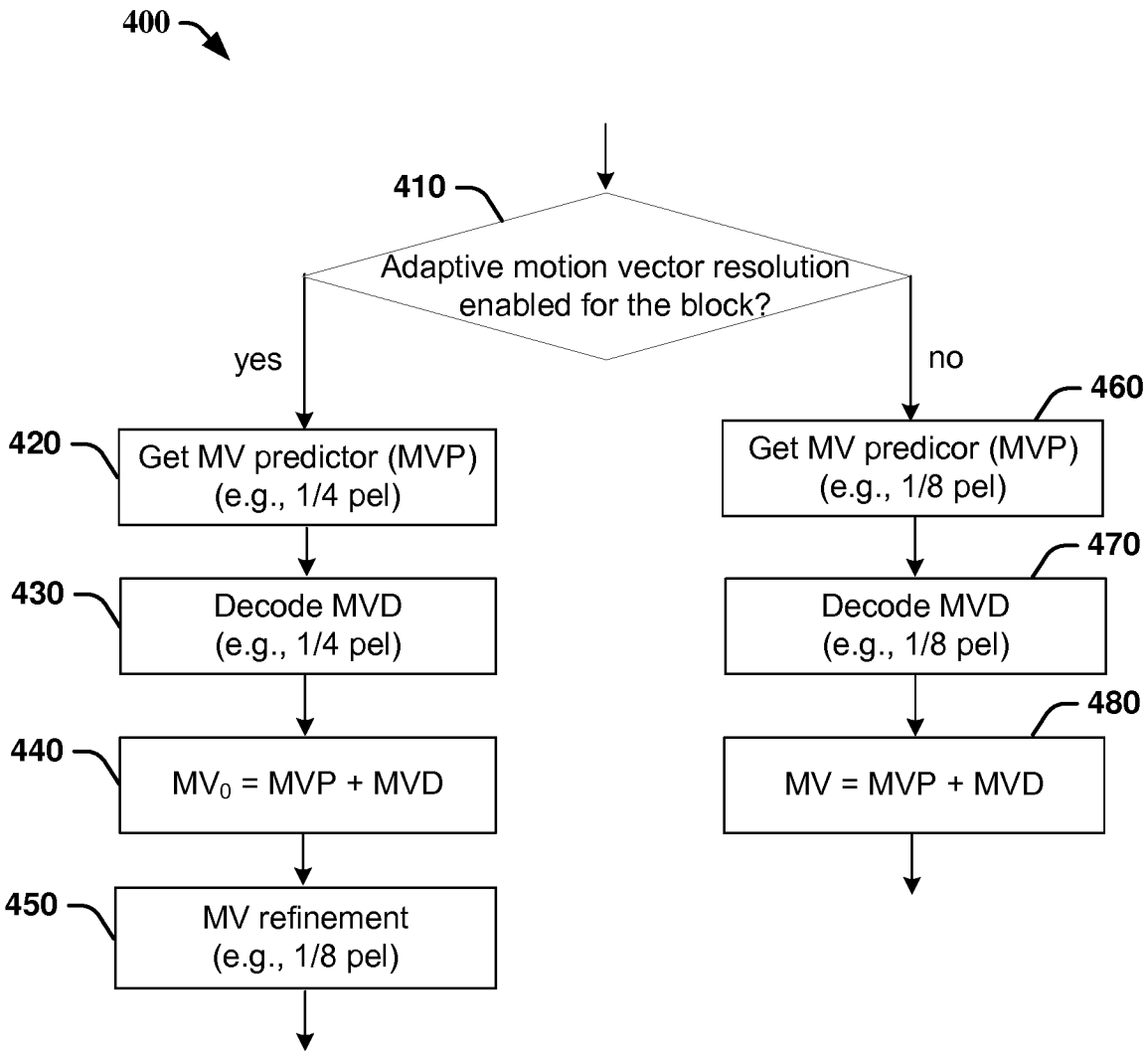
400
410
Adaptive motion vector resolution enabled for the block?
yes
no
420
Get MV predictor (MVP) (e.g., 1/4 pel)
430
Decode MVD (e.g., 1/4 pel)
440
MV0 = MVP + MVD
450
MV refinement (e.g., 1/8 pel)
460
Get MV predicor (MVP) (e.g., 1/8 pel)
470
Decode MVD (e.g., 1/8 pel)
480
MV = MVP + MVD

FIG. 4

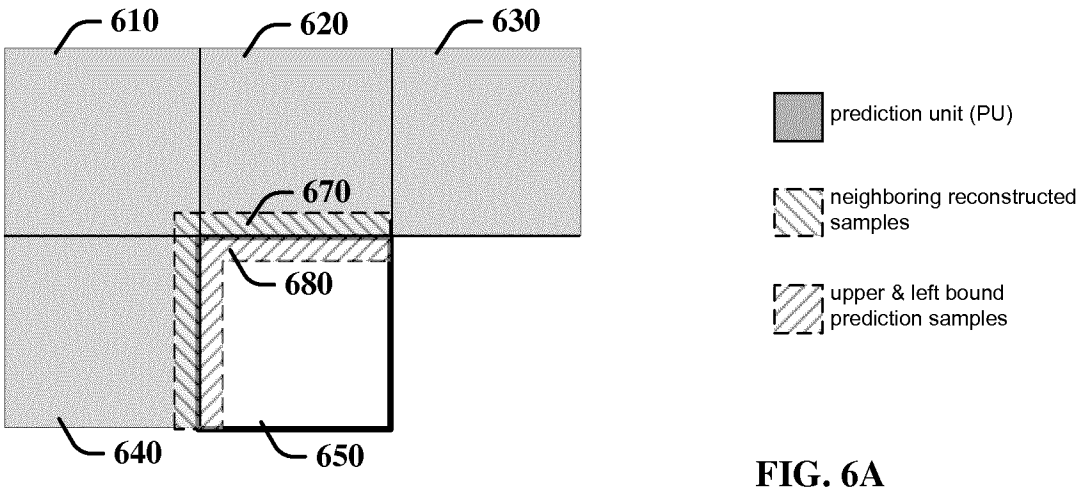

FIG. 6A

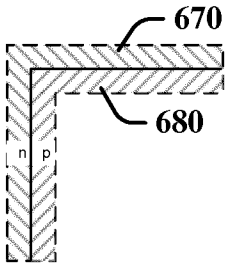

FIG. 6B

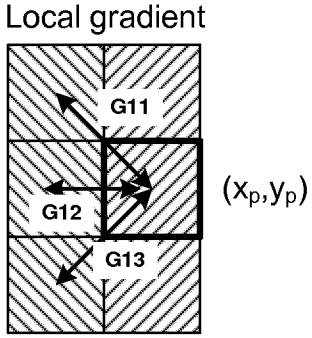

FIG. 7A

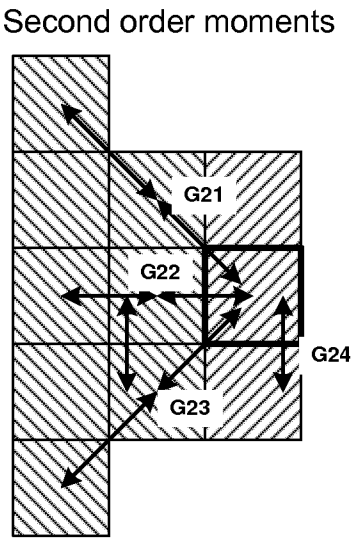

FIG. 7B

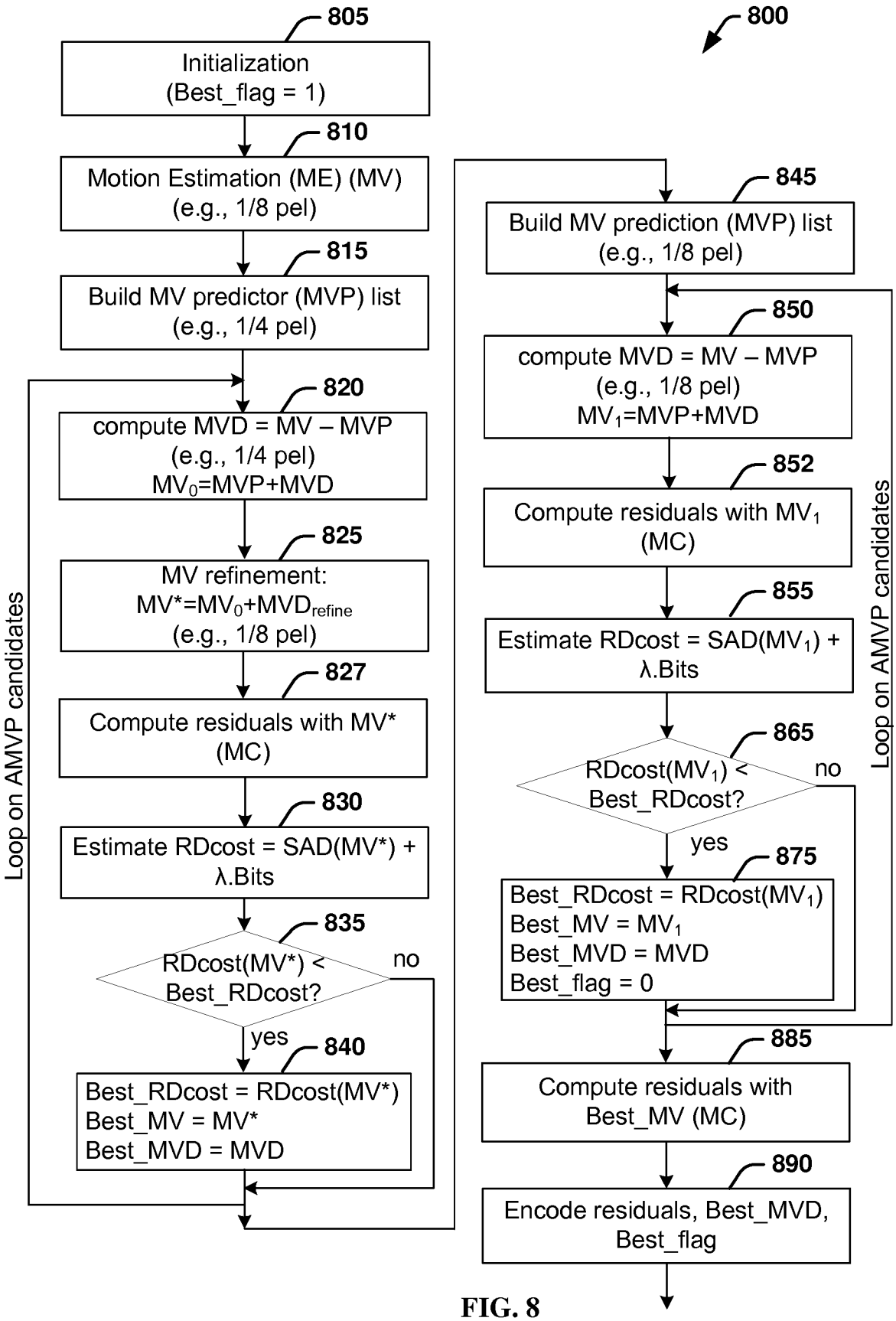
800
805
Initialization
(Best_flag = 1)
810
Motion Estimation (ME) (MV)
(e.g., 1/8 pel)
815
Build MV predictor (MVP) list
(e.g., 1/4 pel)
820
compute MVD = MV – MVP
(e.g., 1/4 pel)
MV0=MVP+MVD
825
MV refinement:
MV*=MV0+MVDrefine
(e.g., 1/8 pel)
827
Compute residuals with MV*
(MC)
830
Estimate RDcost = SAD(MV*) + λ.Bits
835
RDcost(MV*) < Best_RDcost?
no
yes
840
Best_RDcost = RDcost(MV*)
Best_MV = MV*
Best_MVD = MVD
Loop on AMVP candidates
845
Build MV prediction (MVP) list
(e.g., 1/8 pel)
850
compute MVD = MV – MVP
(e.g., 1/8 pel)
MV1=MVP+MVD
852
Compute residuals with MV1
(MC)
855
Estimate RDcost = SAD(MV1) + λ.Bits
865
RDcost(MV1) < Best_RDcost?
no
yes
875
Best_RDcost = RDcost(MV1)
Best_MV = MV1
Best_MVD = MVD
Best_flag = 0
Loop on AMVP candidates
885
Compute residuals with Best_MV (MC)
890
Encode residuals, Best_MVD, Best_flag

FIG. 8

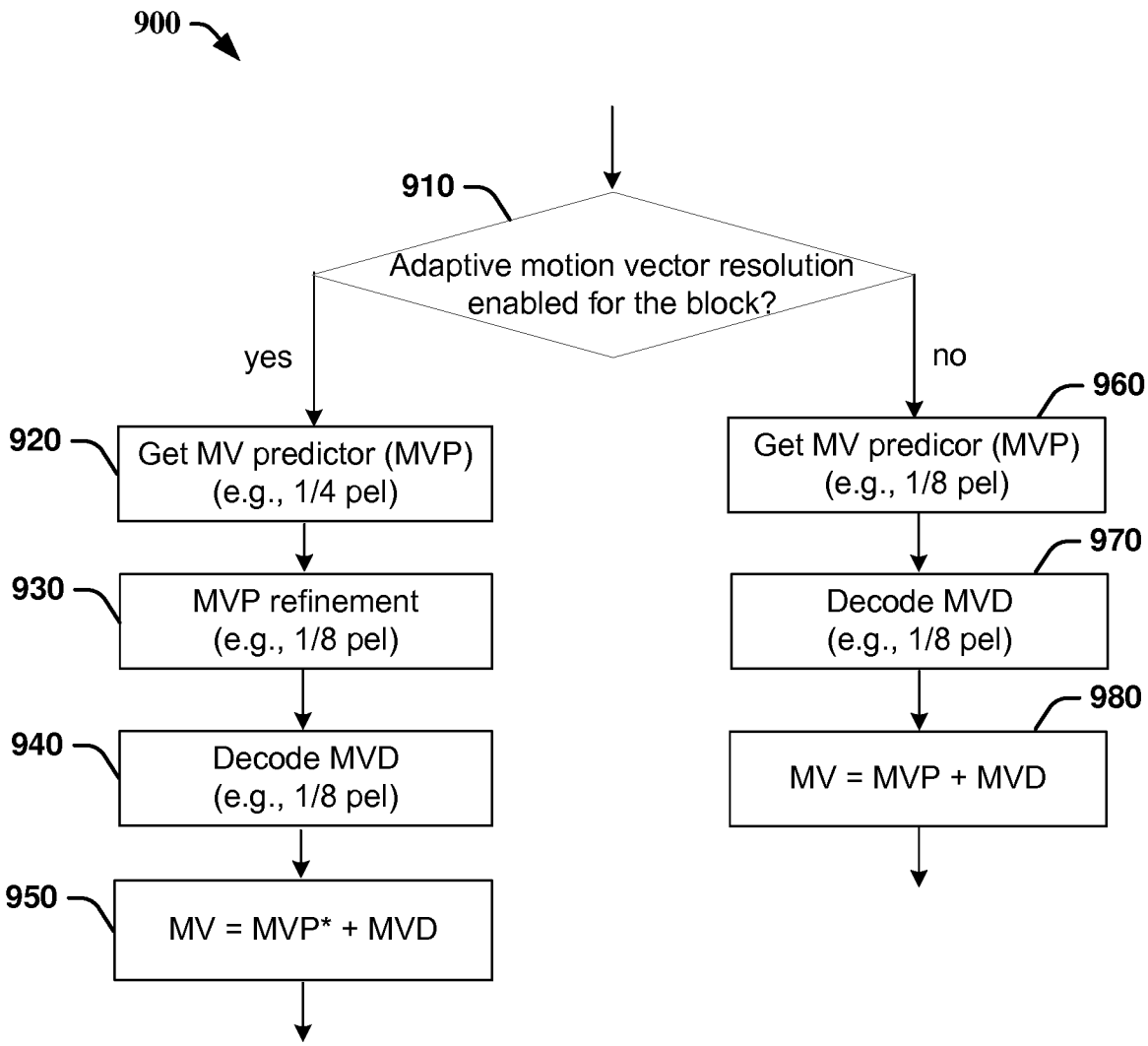
900
910
Adaptive motion vector resolution enabled for the block?
yes
no
920
Get MV predictor (MVP) (e.g., 1/4 pel)
930
MVP refinement (e.g., 1/8 pel)
940
Decode MVD (e.g., 1/8 pel)
950
MV = MVP* + MVD
960
Get MV predicor (MVP) (e.g., 1/8 pel)
970
Decode MVD (e.g., 1/8 pel)
980
MV = MVP + MVD

FIG. 9

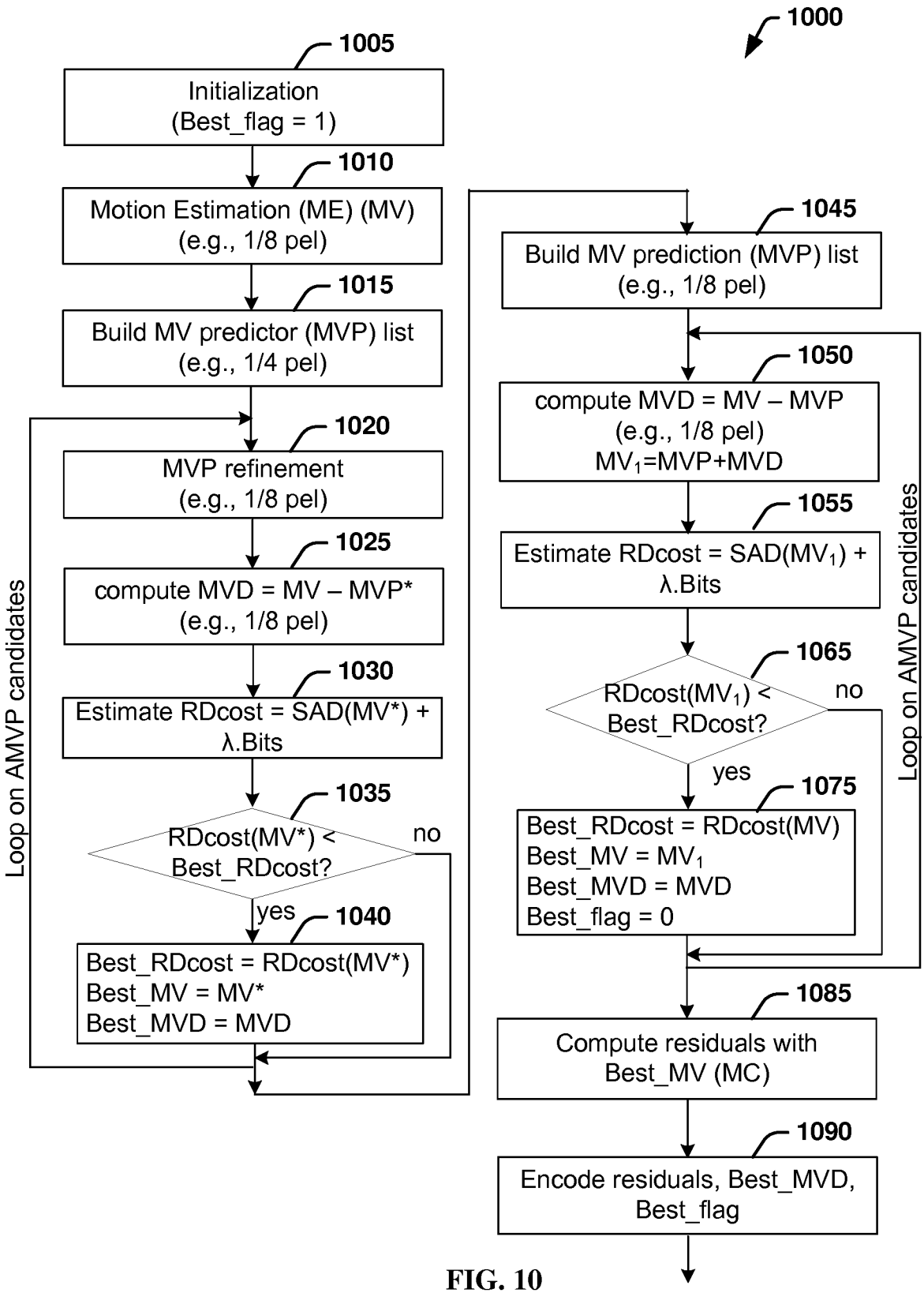
1000
1005
Initialization
(Best_flag = 1)
1010
Motion Estimation (ME) (MV)
(e.g., 1/8 pel)
1015
Build MV predictor (MVP) list
(e.g., 1/4 pel)
1020
MVP refinement
(e.g., 1/8 pel)
1025
compute MVD = MV – MVP*
(e.g., 1/8 pel)
1030
Estimate RDcost = SAD(MV*) + λ.Bits
1035
RDcost(MV*) < Best_RDcost?
no
yes
1040
Best_RDcost = RDcost(MV*)
Best_MV = MV*
Best_MVD = MVD
Loop on AMVP candidates
1045
Build MV prediction (MVP) list
(e.g., 1/8 pel)
1050
compute MVD = MV – MVP
(e.g., 1/8 pel)
MV1=MVP+MVD
1055
Estimate RDcost = SAD(MV1) + λ.Bits
1065
RDcost(MV1) < Best_RDcost?
no
yes
1075
Best_RDcost = RDcost(MV)
Best_MV = MV1
Best_MVD = MVD
Best_flag = 0
Loop on AMVP candidates
1085
Compute residuals with Best_MV (MC)
1090
Encode residuals, Best_MVD, Best_flag

FIG. 10

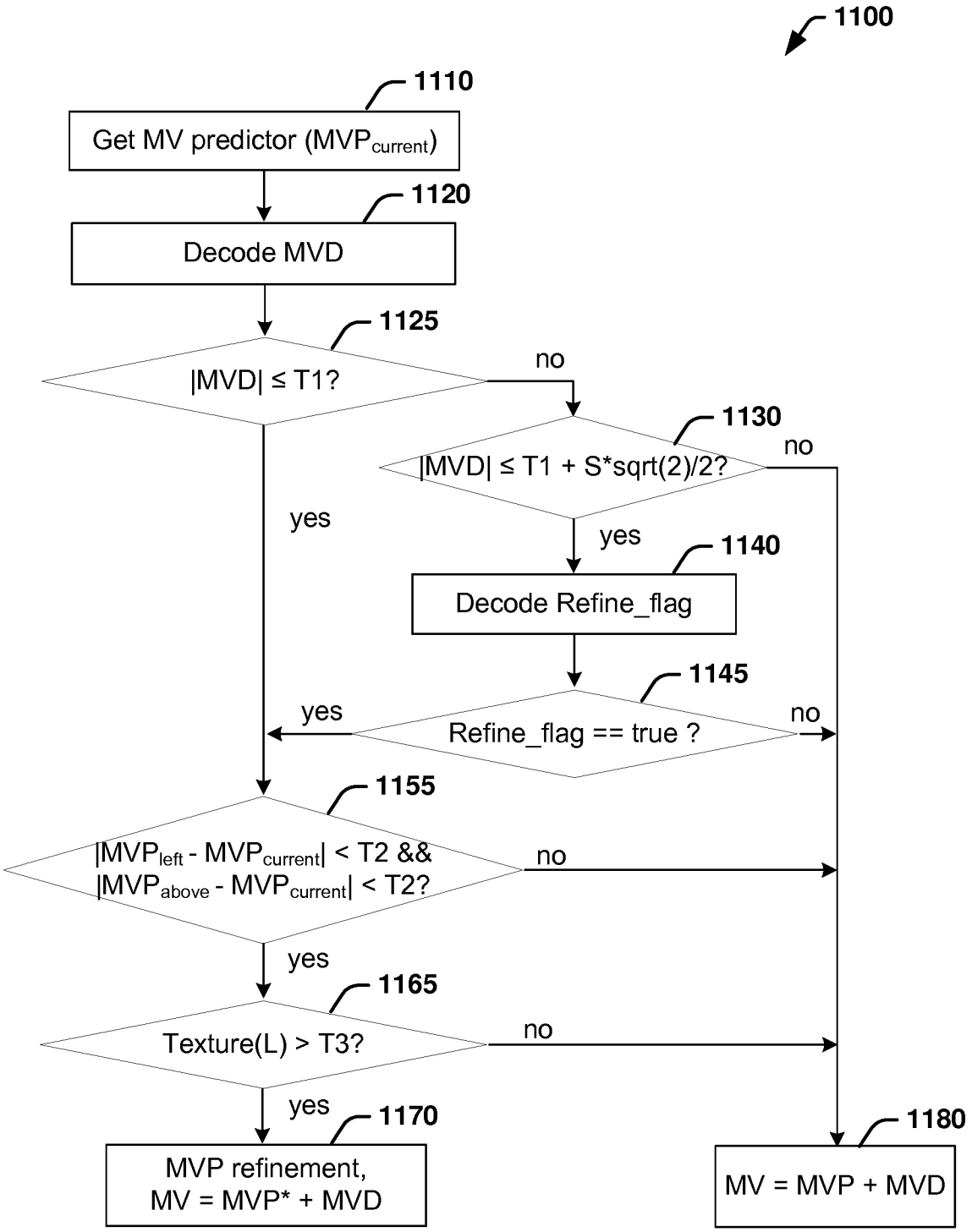
1100
1110
Get MV predictor (MVP$_{current}$)
1120
Decode MVD
1125
|MVD| ≤ T1?
no
yes
1130
|MVD| ≤ T1 + S*sqrt(2)/2?
no
yes
1140
Decode Refine_flag
1145
yes
Refine_flag == true ?
no
1155
|MVP$_{left}$ - MVP$_{current}$| < T2 &&
|MVP$_{above}$ - MVP$_{current}$| < T2?
no
yes
1165
Texture(L) > T3?
no
yes
1170
MVP refinement,
MV = MVP* + MVD
1180
MV = MVP + MVD

FIG. 11

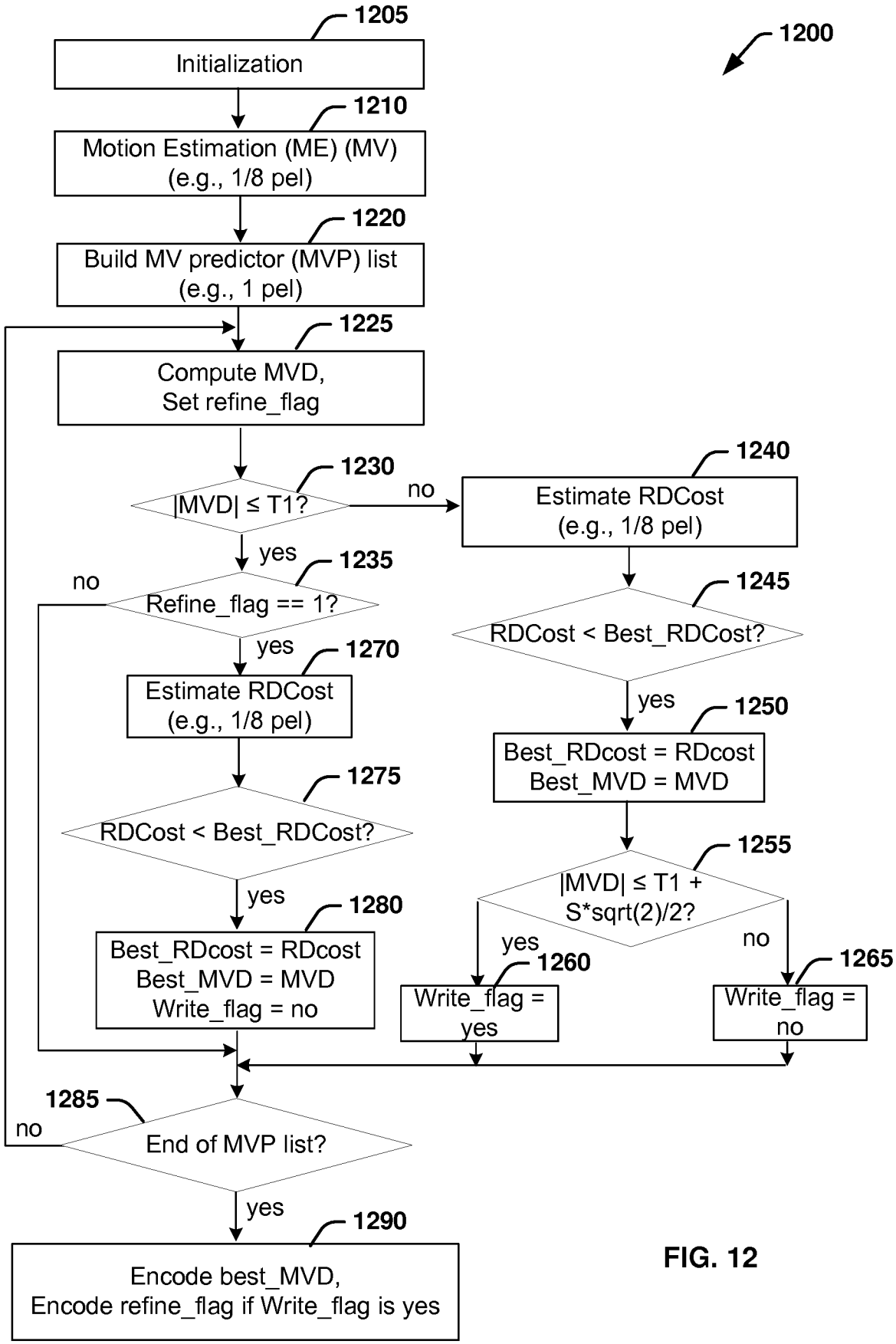
1200
1205
Initialization
1210
Motion Estimation (ME) (MV)
(e.g., 1/8 pel)
1220
Build MV predictor (MVP) list
(e.g., 1 pel)
1225
Compute MVD,
Set refine_flag
1230
|MVD| ≤ T1?
no
yes
1235
Refine_flag == 1?
no
yes
1240
Estimate RDCost
(e.g., 1/8 pel)
1245
RDCost < Best_RDCost?
yes
1250
Best_RDcost = RDcost
Best_MVD = MVD
1255
|MVD| ≤ T1 +
S*sqrt(2)/2?
yes
no
1260
Write_flag =
yes
1265
Write_flag =
no
1270
Estimate RDCost
(e.g., 1/8 pel)
1275
RDCost < Best_RDCost?
yes
1280
Best_RDcost = RDcost
Best_MVD = MVD
Write_flag = no
1285
End of MVP list?
no
yes
1290
Encode best_MVD,
Encode refine_flag if Write_flag is yes

FIG. 12

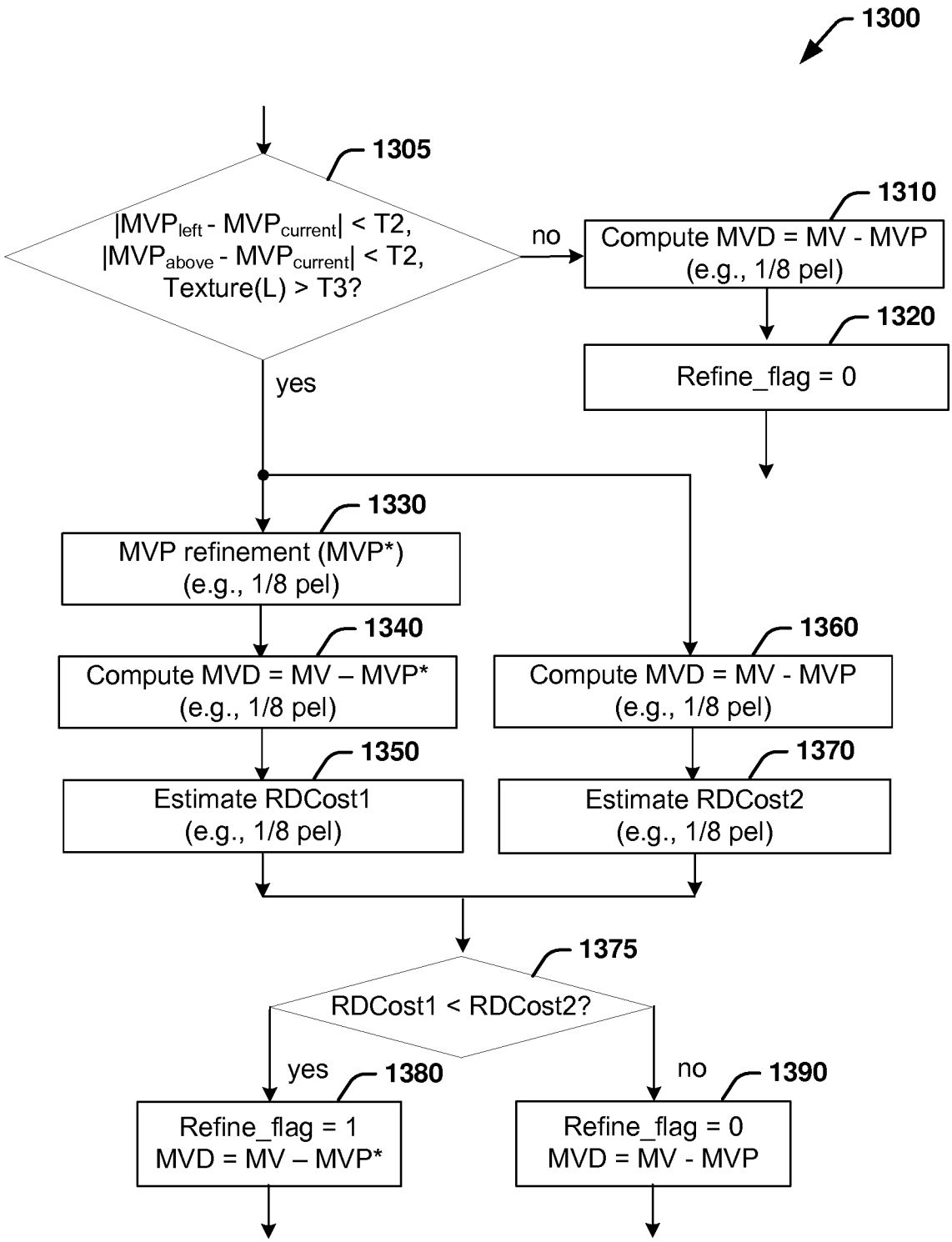
1300
1305
|MVP_left - MVP_current| < T2,
|MVP_above - MVP_current| < T2,
Texture(L) > T3?
no
1310
Compute MVD = MV - MVP
(e.g., 1/8 pel)
1320
Refine_flag = 0
yes
1330
MVP refinement (MVP*)
(e.g., 1/8 pel)
1340
Compute MVD = MV – MVP*
(e.g., 1/8 pel)
1350
Estimate RDCost1
(e.g., 1/8 pel)
1360
Compute MVD = MV - MVP
(e.g., 1/8 pel)
1370
Estimate RDCost2
(e.g., 1/8 pel)
1375
RDCost1 < RDCost2?
yes
1380
Refine_flag = 1
MVD = MV – MVP*
no
1390
Refine_flag = 0
MVD = MV - MVP

FIG. 13

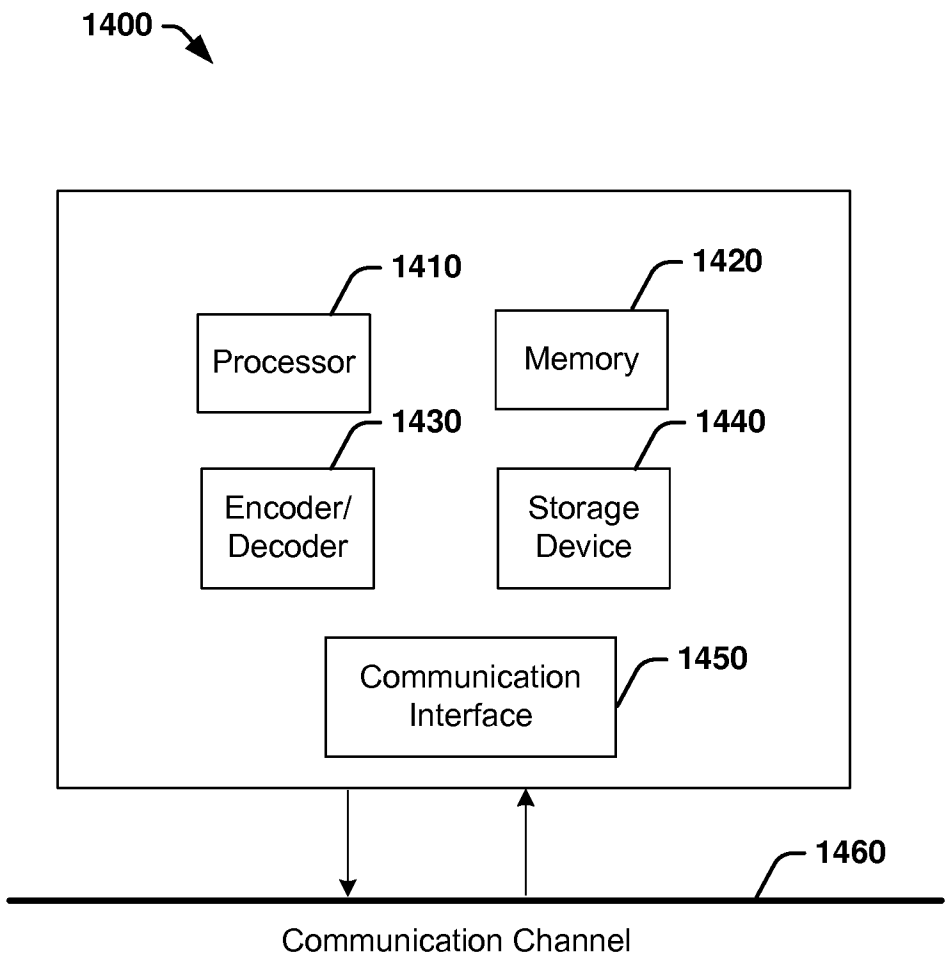
1400
1410
Processor
1420
Memory
1430
Encoder/
Decoder
1440
Storage
Device
Communication
Interface
1450
1460
Communication Channel

FIG. 14

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2017/065809

A. CLASSIFICATION OF SUBJECT MATTER
INV. H04N19/52 H04N19/139 H04N19/176 H04N19/46 H04N19/14 H04N19/513
ADD.
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)
EPO-Internal, COMPENDEX, INSPEC

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WO 2011/047994 A1 (THOMSON LICENSING [FR]; FRANCOIS EDOUARD [FR]; THOREAU DOMINIQUE [FR];) 28 April 2011 (2011-04-28)	1-6, 8-13,15, 16
Y	abstract page 7, line 4 - page 8, line 28; figures 3,4 page 13, line 24 - page 15, line 31; figure 7 claims 3-6	7,14

-/--

[X] Further documents are listed in the continuation of Box C.

[X] 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
22 August 2017	29/08/2017

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Colesanti, Carlo

1

Form PCT/ISA/210 (second sheet) (April 2005)

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	LAKSHMAN H ET AL: "Conditional motion vector refinement for improved prediction", 2012 PICTURE CODING SYMPOSIUM (PCS 2012) : KRAKOW, POLAND, 7 - 9 MAY 2012 ; [PROCEEDINGS], IEEE, PISCATAWAY, NJ, 7 May 2012 (2012-05-07), pages 497-500, XP032449810, DOI: 10.1109/PCS.2012.6213263 ISBN: 978-1-4577-2047-5 cited in the application abstract Sections I, II.B and III	7,14
A	JIANLE CHEN QUALCOMM INC ET AL: "Algorithm Description of Joint Exploration Test Model 1", 1. JVET MEETING; 19-10-2015 - 21-10-2015; GENEVA; (THE JOINT VIDEO EXPLORATION TEAM OF ISO/IEC JTC1/SC29/WG11 AND ITU-T SG.16); URL: HTTP://PHENIX.INT-EVRY.FR/JVET/,, no. JVET-A1001, 24 February 2016 (2016-02-24), XP030150000, Section 2.4.2	1-16
A	W-J CHIEN (CE COORDINATOR): "Description of Core Experiment 12: Adaptive Motion Vector Resolution", 94. MPEG MEETING; 11-10-2010 - 15-10-2010; GUANGZHOU; (MOTION PICTURE EXPERT GROUP OR ISO/IEC JTC1/SC29/WG11),, no. M18605, 30 October 2010 (2010-10-30), XP030047195, the whole document	1-16

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/EP2017/065809

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 2011047994 A1	28-04-2011	BR 112012009042 A2	19-04-2016
		CN 102668562 A	12-09-2012
		EP 2491717 A1	29-08-2012
		JP 5669278 B2	12-02-2015
		JP 2013509063 A	07-03-2013
		KR 20120096471 A	30-08-2012
		TW 201116070 A	01-05-2011
		US 2013101040 A1	25-04-2013
		WO 2011047994 A1	28-04-2011