

METHOD FOR SKIPPING REFINEMENT BASED ON PATCH SIMILARITY IN BILINEAR INTERPOLATION BASED DECODER-SIDE MOTION VECTOR REFINEMENT

5

CROSS-REFERENCES TO RELATED APPLICATIONS

[0001] The present invention claims priority to Indian Provisional Patent Application No. IN201831035309, filed on September 19, 2018. The Indian Provisional Patent Application is
10 hereby incorporated by reference in its entirety.

TECHNICAL FIELD

[0002] The present invention relates to motion compensation for video coding using decoder-side motion vector refinement. In particular, embodiments of the present invention relate to methods and apparatuses for verifying an alignment level between motion compensated
15 reference patches and skipping refinement when a difference between the two motion compensated reference patches is less than a coding unit size-dependent threshold value.

BACKGROUND OF THE INVENTION

[0003] In video compression, inter prediction is a process of using reconstructed samples of
20 previously decoded reference pictures by specifying motion vectors relative to a current block. These motion vectors can be coded as a prediction residual by using spatial or temporal motion vector predictors. The motion vectors can be at sub-pixel accuracy. In order to derive the sub-pixel accurate pixel values in the reference frames (pictures) from the reconstructed integer position values, an interpolation filter is applied. Bi-prediction refers to a process where the
25 prediction for the current block is derived as a weighted combination of two prediction blocks derived using two motion vectors from two reference picture areas. In this case, in addition to the motion vectors, the reference indices for the reference pictures from which the two prediction blocks are derived also need to be coded. The motion vectors for the current block can also be derived through a merge process where a spatial neighbor's motion vectors and reference indices

are inherited without coding any motion vector residuals. In addition to spatial neighbors, motion vectors of previously coded reference frames are also stored and used as temporal merge options with appropriate scaling of the motion vectors to take care of the distance to the reference frames relative to the distance to the reference frames for the current block.

5 [0004] FIG. 1 shows a template matching based decoder-side motion vector derivation, where the template of the current block is matched with the reference template in a reference picture. Referring to FIG. 1, template matching is employed to derive motion information of the current coding unit (CU) by finding the closest match between a template (top and/or left neighboring blocks of the current CU) in the current picture (denoted “Cur Pic”) and a block having the same
10 size as that of the template in a reference picture (denoted “Ref0”).

[0005] FIG. 2 shows a bilateral matching based decoder-side motion vector derivation, where a current block is predicted using two reference blocks along a motion trajectory. Referring to FIG. 2, motion information of the current block (denoted “Cur block”) is derived based on two reference pictures Ref0 and Ref1. The motion information of the current block Cur block is
15 derived by finding the best match between two blocks associated with the motion vectors MV0 and MV1 along a motion trajectory in the reference pictures Ref0 and Ref1. When the motion trajectory is a straight line, the motion vector MV0 associated with reference picture Ref0 and the motion vector MV1 associated with reference picture Ref1 are proportional to the temporal distances TD0 and TD1 between the current picture and the respective reference pictures Ref0
20 and Ref1.

[0006] Several methods have been proposed for performing a decoder-side motion vector refinement or derivation so that the motion vector residual coding bits can be further reduced. One class of methods, called template matching (TM) methods, use an L-shaped region adjoining the current block (as shown in FIG. 1) that has already been reconstructed, referred to
25 as the template, and identifies a best matching L-shaped region (using cost functions such as sum of absolute differences or mean-removed sum of absolute differences) in each reference frame using a plurality of suitably scaled spatial and temporal motion vector candidates. Then, centered on the best matching candidate, further refinement is performed within a certain refinement distance around that center. On the encoder side, rate distortion optimized cost is computed to

decide between uni-prediction (i.e., prediction using the best matching reference) and bi-prediction (i.e., prediction derived by averaging the top two best matching references).

[0007] Another class of methods, called bilateral matching (BM) methods derive motion information of the current coding unit (CU) by finding the closest match between two blocks
5 along the motion trajectory of the current CU in two different reference pictures. This is shown in FIG. 2. Under the assumption of continuous motion trajectory, the motion vectors MV0 and MV1 pointing to the two reference blocks shall be proportional to the temporal distances, i.e., TD0 and TD1, between the current picture and the two reference pictures. When the current picture is temporally between the two reference pictures and the temporal distance from the
10 current picture to the two reference pictures is the same, the bilateral matching becomes mirror based bi-directional motion vector (MV).

[0008] In the bilateral matching merge mode, bi-prediction is always applied since the motion information of a CU is derived based on the closest match between two blocks along the motion trajectory of the current CU in two different reference pictures.

15 [0009] Explicit merge mode to indicate template matching merge or bilateral matching merge can be signaled to differentiate these modes from a default merge mode that does not require any decoder-side motion vector derivation.

[0010] In the bilateral matching mode, the temporal distances are ignored and bilateral matching is performed with equal and opposite motion vectors in the past and future reference
20 frames respectively.

[0011] In some cases, no merge index is signaled while in other cases, to simplify the decoder complexity of performing multiple motion compensations, an explicit merge index is signaled.

[0012] In a variant of bilateral matching mode called decoder-side motion vector refinement (DMVR) method, a bilaterally averaged template is first created using the prediction blocks in
25 reference lists L0 and L1 obtained from explicitly signaled merge index and bilateral matching is performed against this template. The template is updated if there is any movement. Also, in some cases, the refinement is performed in one reference and the motion vector in the other reference is obtained through mirroring of this refined motion vector. The refinement alternates between

the two references until either the center position has the least error or the maximum number of iterations is reached.

[0013] In some refinement processes, a CU level refinement is first performed. Then a sub-CU level multi-candidate evaluation is performed along with the CU-level refined MVs as candidates.

5 In other refinement processes, each sub-CU can perform its own refinement with respect to the best matching candidate.

[0014] Given the implicit decoder-side derivation or refinement process, the encoder needs to perform these steps in exactly the same manner as the decoder in order for the encoder-side reconstruction to match with the decoder-side reconstruction.

10 [0015] Only luma samples are typically used during the decoder side motion vector refinement or derivation process. However, chrominance is also motion compensated using the final refinement motion vectors (suitably scaled to account for any chroma downsampling) as used for luma motion compensation.

[0016] Since the merge motion vectors are sub-pixel accurate, the refinement is typically
15 performed initially at integer distance from the respective sub-pixel accurate merge motion vectors in each reference. Since the normative motion compensation is computationally expensive, a bilinear interpolation method is commonly used to create the interpolated sample values required for performing the integer distance refinement. Other prior art techniques made
20 proposal of using cost function values evaluated at integer distance positions and the cost function value at the position with the lowest cost when the refinement ends to obtain a parametric error surface using which sub-pixel delta motion vector corrections are estimated. Once the final integer distance plus sub-pixel distance based motion vector corrections in each reference is obtained, final normative motion compensation is performed.

25 BRIEF SUMMARY OF THE INVENTION

[0017] Embodiments of the present invention relate to methods and apparatuses for determining the level of alignment between motion compensated reference patches against a predetermined coding unit size dependent threshold value. By determining the alignment level between motion compensated reference patches against the predetermined coding unit size

dependent threshold value, computational operations of the decoder-side motion vector refinement iterations can be saved, thereby reducing power consumption in a video decoder.

[0018] Methods and apparatuses of determining an alignment level between motion compensated reference patches for reducing motion vector refinement steps are provided.

- 5 According to one method, a video decoder determines the alignment level by rounding merge motion vectors, calculates the sum of absolute differences (SAD) between the two motion compensated reference patches by performing simple motion compensation using the rounded motion vectors, and determines whether the SAD is less than a CU size-dependent threshold value. When the SAD is less than the CU size-dependent threshold value, the decoder skips
- 10 remaining decoder-side motion vector process steps, and performs final motion compensation using the unrounded merge motion vectors. In one embodiment, the unrounded merge motion vectors may be rounded to a nearest integer sample position in the horizontal direction and in the vertical direction, and the final motion compensation includes performing an averaging of the integer position samples. In another embodiment, the unrounded merge motion vectors may be
- 15 rounded to a nearest half-pixel sample position in the horizontal direction and in the vertical direction to obtain half-pixel position samples, and the final motion compensation includes performing an averaging of the half-pixel position samples. The method is advantageous in that whenever the SAD is less than a predetermined threshold value, the video decoder determines that the alignment level between motion compensated reference patches is acceptable and the
- 20 video decoder can skip the motion vector refinement processes such as bilinear interpolation, mean value determination, refinement cost function evaluations, and other process steps to save clock cycles and reduce power consumption. In another method, the video decoder calculates the SAD between the two motion compensated reference patches using a subset of samples from a bilinear motion compensated interpolation using sub-pixel accurate merge motion vectors. The
- 25 SAD is then compared with a coding unit (CU) size-dependent threshold value. The bit depth of the coding unit size-dependent threshold value can be adjusted according to the bit depth of the interpolated samples. By computing only the SAD using a subset of interpolated samples, the method advantageously requires fewer operations (clock cycles). In yet another method, the decoder calculates a mean-reduced SAD (MR-SAD) between motion compensated interpolated
- 30 samples at center positions of respective motion compensated reference patches and determines whether the MR-SAD is less than a CU size-dependent threshold value. In this method, the MR-

SAD of the center positions of the two motion compensated reference patches is computed where the refinement begins. When the MR-SAD for the center positions is less than the CU size-dependent threshold value, the video decoder skips the remaining of the decoder-side motion vector refinement process. An inter prediction method, comprising: obtaining a first motion compensated reference patch and a second motion compensated reference patch based on initial motion vectors of Merge Mode; computing a patch difference value between the first motion compensated reference patch and the second motion compensated reference patch; obtaining a prediction value of a current block based on the initial motion vectors in the event that the patch difference value is less than a threshold, wherein the threshold is determined based on a size of the current block; and refining the initial motion vectors to obtain the prediction value of the current block in the event that the patch difference value is greater than or equal to the threshold. In a feasible implementation, the initial motion vectors are derived from motion vectors of a neighboring block of the current block. In a feasible implementation, the initial motion vectors comprises a first motion vector and a second motion vector, wherein the first motion compensated reference patch is obtained according to the first motion vector, and the second motion compensated reference patch is obtained according to the second motion vector. In a feasible implementation, the patch difference value is a SAD value or a mean-reduced SAD value. In a feasible implementation, computing the patch difference value between the first motion compensated reference patch and the second motion compensated reference patch, comprising: computing the patch difference value between a subset samples of the first motion compensated reference patch and a corresponding subset samples of the second motion compensated reference patch. In a feasible implementation, after computing the patch difference value between the first motion compensated reference patch and the second motion compensated reference patch, further comprising: performing a comparison based on the patch difference value and the threshold. In a feasible implementation, before performing the comparison based on the patch difference value and the threshold, further comprising: adjusting the threshold after determining the threshold according to the size of the current block. In a feasible implementation, in the event that the patch difference value is less than the threshold, before obtaining the prediction value of the current block based on the initial motion vectors, further comprising: determining decoder-side motion vector refinement (DMVR) is skipped. In a feasible implementation, in the event that the patch difference value is less than the threshold, obtaining

the prediction value of the current block using motion vectors that are the same as the initial motion vectors. An inter prediction apparatus, comprising: a non-transitory memory having processor-executable instructions stored thereon; and a processor, coupled to the memory, configured to execute the processor-executable instructions to facilitate the method according to any one of feasible implementation of the above inter prediction method. These novel techniques save substantial iterations of the motion vector refinement, thereby reducing the number of clock cycles and power consumption of the video decoder.

BRIEF DESCRIPTION OF THE DRAWINGS

10 [0019] FIG. 1 shows a template matching based decoder-side motion vector derivation, where the template of the current block is matched with the reference template in a reference picture.

[0020] FIG. 2 shows a bilateral matching based decoder-side motion vector derivation, where a current block is predicted using two reference blocks along a motion trajectory.

15 [0021] FIG. 3 shows an example of a bilateral template matching based decoder-side motion vector refinement (DMVR), where a template is generated by using the bi-prediction from the prediction blocks referred by the initial motion vectors MV0 and MV1 and the bilateral template matching to find the best matched blocks referred by the updated motion vectors MV0' and MV1'.

20 [0022] FIG. 4 is a simplified flowchart illustrating a method of determining an alignment level between motion compensated reference patches according to an embodiment of the present invention, where the conditional skipping of the refinement process steps is based on a rounded integer position-based cost function.

25 [0023] FIG. 5 shows a simplified flowchart of a method using decoder-side verification, checking or determination of an alignment level between motion compensated patches according to an embodiment of the present invention, where the conditional skipping of the refinement process steps is based on a sum of absolute differences between two motion compensated reference patches using a subset of motion compensated interpolated samples.

[0024] FIG. 6 shows a simplified flowchart of a method using decoder-side verification, checking or determination of an alignment level between motion compensated patches according

to an embodiment of the present invention, where the conditional skipping of the refinement process steps is based on a mean-reduced sum of absolute differences between motion compensated interpolated samples at center positions of respective motion compensated reference patches.

5 [0025] FIG. 7 is a block diagram showing an example structure of a content supply system 3100 which realizes a content delivery service.

[0026] FIG. 8 is a block diagram showing a structure of an example of a terminal device.

10

DETAILED DESCRIPTION OF THE INVENTION

[0027] The present invention relates to versatile video coding standardization which was earlier pursued as a Joint Exploratory Model (JEM) within Joint Video Exploration Team which is a joint work between Q16 of VCEG and MPEG (SC29/WG11). Document JVET-G1001 and other Huawei prior art relating to decoder-side motion vector refinement and decoder-side motion vector derivation can be used to get a list of contribution documents and patents related to this invention.

[0028] As explained above, the bilateral matching process tries to improve the level of alignment between the motion compensated blocks in the two references involved in bi-prediction. In order to avoid coding an extra flag, the preferred option is for all merge mode inter-coded coding units to be eligible for decoder-side motion vector refinement. However, it is noticed that quite a good number of coding units have very good alignment already at the merge mode motion vectors and do not have much room for improvement through refinement. Since refinement involves performing cost function evaluations at multiple positions within the refinement range, avoiding these operations when there is no possible improvement that can result in considerable reduction of average decoder complexity which can translate to attributes such as better battery life or lower consumption of energy. Some prior art methods have proposed skipping the refinement if the level of alignment between the two normative motion compensated patches obtained from the merge motion vectors in bi-predictive merge mode coding units is less than a pre-determined threshold for each coding-unit size. However, given the fact that bilinear interpolation is commonly employed to create the samples used for

performing refinement, prior art would still require a normative motion compensation for checking whether refinement can be skipped, bilinear interpolation for performing the refinement, and final normative motion compensation with the sub-pixel accurate refinement motion vectors. Since the normative 8-tap interpolation filters are quite computational expensive, performing the interpolation filtering twice makes the worst-case complexity higher than without the test for skipping the refinement. Also, from a hardware timing point of view, the sum of absolute difference evaluation adds a dependent stage that reduces the clocks available for refinement.

[0029] Hence there is a need to harmonize the concept of checking to skip the refinement iterations without increasing the interpolation cycles (when compared to the bilinear interpolation based refinement with final exact motion compensation using refined motion vectors) and without worsening the worst-case timing constraints for hardware implementations.

[0030] Motion vector (MV) refinement is a pattern based MV search with the criterion of bilateral matching cost or template matching cost. In the current development, two search patterns are supported – an unrestricted center-biased diamond search (UCBDS) and an adaptive cross search for MV refinement at the CU level and sub-CU level, respectively. For both CU and sub-CU level MV refinement, the MV is directly searched at quarter luma sample MV accuracy, and this is followed by one-eighth luma sample MV refinement. The search range of MV refinement for the CU and sub-CU step are set equal to 8 luma samples.

[0031] In bi-prediction operation, for the prediction of one block region, two prediction blocks, formed using a MV of list0 and a MV of list1, respectively, are combined to form a single prediction signal. In the decoder-side motion vector refinement (DMVR) method, the two motion vectors of the bi-prediction are further refined by a bilateral template matching process. The bilateral template matching applied in the decoder to perform a distortion-based search between a bilateral template and the reconstruction samples in the reference pictures in order to obtain a refined MV without transmission of additional motion information.

[0032] FIG. 3 shows an example of a bilateral template matching based decoder-side motion vector refinement (DMVR), where a template is generated by using the bi-prediction from the prediction blocks referred by the initial MV0 and MV1 and the bilateral template matching to find the best matched blocks referred by the updated MV0' and MV1'. First, the best match MV is set for the current block in a merge mode to be the candidate MV0 in list0. Similarly, the best

match MV is set for the current block (in the merge mode) to be the candidate MV in list1. The reference pixels are then averaged to form a template. Next, using the template, the surrounding regions of the candidate MVs of the first and second reference pictures are searched, and the MV with the lowest cost is determined to be the final MV. The cost value may be calculated using a sum of differences between each pixel in the template and each pixel in the regions searched. Referring to FIG. 3, in step 1, a bilateral template is generated from the prediction blocks referred to by the initial motion vectors MV0 and MV1 in the reference blocks in list0 and list1, respectively.

[0033] In step 2, a bilateral matching is to find the best matched blocks referred by the updated motion vectors MV0' and MV1'. The template matching operation includes calculating cost measures between the generated template and the sample region (around the initial prediction block) in the reference picture. For each of the two reference pictures, the MV that yields the minimum template cost is considered as the updated MV of that list to replace the original one. In the current development, nine MV candidates are searched for each list. The nine MV candidates include the original MV and 8 surrounding MVs with one luma sample offset to the original MV in either the horizontal or vertical direction, or both. Finally, the two new MVs, i.e., MV0' and MV1', are used for generating the final bi-prediction results. A sum of absolute differences (SAD) is used as the cost measure. DMVR is applied for the merge mode of bi-prediction with one MV from a reference picture in the past and another from a reference picture in the future, without the transmission of additional syntax elements.

[0034] In various embodiments, the difference between two motion compensated reference patches is compared or measured against a pre-determined coding unit size dependent threshold value, and the refinement is skipped based on the comparison or measurement result. The difference between two motion compensated reference patches is performed either using pre-interpolation samples or using a subset of the bilinear interpolated samples. Also, in some embodiments, a mean-removed sum of absolute differences (MR-SAD) of the patches may be used as the error metric for refinement, simple sum of absolute differences of the patches may be used as the metric on which the skipping check is performed in order to allow saving of the mean computations on the average and also to help with the worst-case timing requirements. The pre-

determined threshold value is adapted to the lower bit-depths that may be employed during or after interpolation to reduce the buffer size and processing requirements.

[0035] Given that decoder-side motion vector refinement/derivation is a normative aspect of a coding system, the encoder will also have to perform the same error surface technique in order to not have any drift between the encoder's reconstruction and the decoder's reconstruction. Hence, all aspects of all embodiments are applicable to both encoding and decoding systems.

Embodiments provided below are merely some examples for understanding the present disclosure and thus should not be understood as being intended to be limiting.

[0036] Embodiment 1

[0037] In this embodiment, the merge motion vectors are rounded to the nearest integer position. Using these rounded integer grid motion vectors as the offset with respect to the current coding unit position in the two references used for bi-prediction, a sum of absolute differences (SAD) between the blocks of coding unit size samples is computed. This sum of absolute differences is compared against a coding unit (CU) size-dependent threshold value and the rest or remaining of the decoder-side motion vector refinement process steps is normatively skipped when the sum of absolute differences is less than the threshold value.

[0038] The appropriate pre-determined CU size-dependent threshold value is determined based on the desired average computational savings or energy saving traded-off against the drop in compression gains with a high threshold.

[0039] FIG. 4 is a simplified flowchart of a method using decoder-side verification or determination of an alignment level between motion compensated patches according to an embodiment of the present invention, where the conditional skipping of the remaining refinement process steps is based on a rounded integer position-based cost function. The steps shown in the flowchart may be implemented as program codes or instructions executable on one or more processors at the encoder side or the decoder side. The steps shown in the flowchart may be implemented using electronic components, digital and/or analog circuitry, logic elements, hardware, software, firmware, or combinations thereof. The method may include, at step 401, a video decoder (video decoder and decoder are interchangeably used herein) obtains unrounded merge motion vectors between two motion compensated reference patches. A patch has a

predetermined size of a current block of $M \times N$ pixels, M and N each are positive integers. In some embodiments, M may be equal to N . In other embodiments, M and N may be different. A patch may be a prediction unit or a coding unit and is referred to as a block of samples within a picture (frame). The unrounded merge motion vectors can be obtained using bilateral matching, template matching, or other techniques. Unrounded merge motion vectors can be merge motion vectors having an integer pixel precision or a fractional pixel precision. The decoder rounds the unrounded merge motion vectors to obtain the rounded motion vectors at step 402. Rounding operation may include converting an unrounded motion vector to an integer pixel accuracy motion vector or reducing the pixel precision of the motion vector. For example, a fractional precision motion vector can be rounded to an integer pixel precision motion vector. The rounding operation may include right shifting of the vector value by one or more bits. In one embodiment, the video decoder may round the unrounded merge motion vectors to the nearest integer sample positions. In another embodiment, the video decoder may round the unrounded merge motion vectors to the nearest half-pixel sample positions.

15 **[0040]** At step 403, the video decoder computes (calculates) a sum of absolute differences (SAD) between the two motion compensated reference patches by performing simple motion compensation using the rounded motion vectors. The sum of differences SAD (first patch, second patch) in the respective reference pictures Ref0, Ref1 is a cost function for determining the best template match in the respective search spaces. At step 404, the decoder compares the SAD with a coding unit (CU) size-dependent threshold value to determine whether the SAD is less than or not less than (i.e., equal to or greater than) the CU size-dependent threshold value. When the SAD is determined to be less than the CU size-dependent threshold value ("yes" in step 405), the decoder skips the decoder-side motion vector refinement (DMVR) process steps (step 406) and performs the final motion compensation using the unrounded merge motion vectors (step 408).

25 When the SAD is determined to be not less than the CU size-dependent threshold value ("no" in step 405), the decoder performs the decoder-side motion vector refinement (DMVR) process steps (step 407), and then performs the final motion compensation (step 408). The decoder may refine the motion vector candidates based on a reference block in a search window that closely matches the current block, i.e., the decoder may determine a new interpolated motion vector (e.g., using bilinear interpolation) for the current block when the SAD is not less than the CU size-dependent threshold value, thereafter, the decoder then performs the final motion compensation

30

for the current block. In one embodiment, performing the final motion compensation may include rounding the unrounded merge motion vectors to the nearest integer sample position in the horizontal direction and in the vertical direction to obtain integer position samples, and performing an averaging operation on the integer position samples. In another embodiment, performing the final motion compensation may include rounding the unrounded merge motion vectors to the nearest half-pixel sample position in the horizontal direction and in the vertical direction to obtain half-pixel position samples, and performing an averaging operation on the half-pixel position samples.

[0041] This embodiment is advantageous, among other advantages, in that whenever the metric evaluated is less than the threshold value, all others aspects of decoder-side motion vector refinement such as bilinear interpolation, mean value determination, and refinement cost function evaluations at different positions, can be skipped and the final normative motion compensation can be performed. It should be noted that in hardware implementations, it is possible in some embodiments to speculatively perform bilinear interpolation in parallel with the evaluation of the metric for skipping the refinement to improve the timing constraint (i.e., obtain more time or clock cycles to perform the refinement tasks).

[0042] Embodiment 2

[0043] In this embodiment, the decoder computes the sum of absolute differences between the motion compensated patches in the two references involved in bi-prediction using a subset of samples from the bilinear motion compensated interpolation performed using the sub-pixel accurate merge motion vectors. This sum of absolute differences is then compared against a pre-determined coding unit size dependent threshold value and the rest of the refinement steps are normatively skipped when the decoder determines that the sum of absolute differences is less than the threshold value.

[0044] In one exemplary implementation, the bilinear motion compensated interpolation is performed in parallel (concurrently) on both the reference patches. In another embodiment, the production of the interpolated samples across the two references are interleaved. In both these cases, the sum of absolute difference calculation is advantageously computed at the earliest possible time rather than waiting for all interpolated samples to be produced. This process also avoids loading the interpolated samples again. This reduces the hardware latency and improves

the time available to perform the remaining refinement computations required for the worst-case path.

[0045] In some implementations, a block of samples with the dimensions (size) equal to the coding unit dimensions are used for computing the sum of absolute differences. In such
5 embodiments, the computation of interpolated samples required for the refinement positions other than the central position can be skipped when the sum of absolute differences is determined to be less than the threshold value. In other embodiments, only a subset of interpolated samples are utilized for computing the sum of absolute differences. In these cases, the pre-determined threshold value is modified to be a function of the number of samples used for the sum of
10 absolute differences. When the decoder determines that the sum of absolute differences is less than the threshold value, the remaining interpolation also can be skipped. In one embodiment, the subset of interpolated samples can be $(\text{coding_unit_width} - 2) * (\text{coding_unit_height} - 2)$ samples in one embodiment. In another embodiment, the subset of interpolated samples can be $(\text{coding_unit_width} * \text{coding_unit_height} - (\text{coding_unit_width} - 2) * (\text{coding_unit_height} - 2))$
15 samples in another embodiment, where coding_unit_width is a width of the coding unit, and $\text{coding_unit_height}$ is a height of the coding unit.

[0046] When bilinear interpolation constrains the bit depth of the interpolated samples, the pre-determined threshold is also adjusted accordingly by shifting down the threshold value derived at unconstrained bit-depth of the interpolated samples.

20 [0047] FIG. 5 shows a simplified flowchart of a method using decoder-side verification, checking or determination of an alignment level between motion compensated patches according to an embodiment of the present invention, where the conditional skipping of the decoder-side motion vector refinement process steps is based on a subset of the bilinear motion compensated interpolated samples. The steps shown in the flowchart may be implemented as program codes or
25 instructions executable on one or more processors at the encoder side or the decoder side. The steps shown in the flowchart may also be implemented using electronic components, digital and/or analog circuitry, logic elements, hardware, software, firmware, or combinations thereof. The method may include, at step 501, a video decode obtains unrounded merge motion vectors between two motion compensated reference patches. A patch has a predetermined size of a
30 current block of $M \times N$ pixels, M and N each are positive integers. In some embodiments, M and

N may be the same. In other embodiments, M and N may be different. The unrounded merge motion vectors can be obtained using bilateral matching, template matching, or other techniques. The decoder also obtains motion compensated interpolated samples based on sub-pixel accurate merge motion vectors at step 502.

- 5 **[0048]** At step 503, the decoder computes (calculates) a sum of absolute difference (SAD) between the two motion compensated reference patches using a subset of the motion compensated interpolated samples. In some embodiments, the SAD can be computed in parallel or concurrently alongside the interpolation process. At step 504, the decoder compares the SAD with a coding unit (CU) size-dependent threshold value to determine whether the SAD is less
10 than or not less than the CU size-dependent threshold value. In one embodiment, the CU size-dependent threshold value is a function of the number of samples in the subset of the interpolated samples. When the SAD is determined to be less than the CU size-dependent threshold value (“yes” in step 505), the decoder skips the decoder-side motion vector refinement (DMVR) process steps (step 506) and performs the final motion compensation using the unrounded merge
15 motion vectors (step 508). When the SAD is determined to be not less than (i.e., equal to or greater than) the CU size-dependent threshold value (“no” in step 505), the decoder performs the decoder-side motion vector refinement (DMVR) process steps using the unrounded merge motion vectors (step 507), and then performs the final motion compensation using the unrounded merge motion vectors (step 508). The decoder may refine the motion vector candidates based on
20 a reference block in a search window that closely matches the current block, i.e., the decoder may determine a new interpolated motion vector (e.g., using bilinear interpolation) for the current block when the SAD is not less than the CU size-dependent threshold value, thereafter, the decoder then performs the final motion compensation for the current block using the unrounded merge motion vectors.
- 25 **[0049]** The second embodiment (embodiment 2) allows less coding gain decrease as a conservative threshold can still allow many coding units to skip the remaining steps of the refinement process. From a timing point of view, the sum of absolute difference operations required for the determination or verification of the SAD against the threshold can be substantially hidden against the bilinear interpolation step and thus will not impact the worst-
30 case timing constraint.

[0050] Embodiment 3

[0051] In this embodiment, the early exit check (determination) is performed using the mean-removed sum of absolute differences (MR-SAD) computed for the center position where the refinement starts. In some embodiments, the center position's MR-SAD is evaluated first and compared against a coding unit size dependent threshold. If the MR-SAD for the center position is less than the threshold value, then the rest of the refinement process is skipped normatively.

[0052] In some embodiments, it is possible that MR-SAD evaluations at positions other than the center position (e.g. certain positions within the first iteration refinement points) may be evaluated alongside the MR-SAD evaluation of the center position. However, these evaluated MR-SADs will have no effect when the center position's MR-SAD is less than the pre-determined threshold value.

[0053] When interpolation constrains the bit depth of the interpolated samples, the pre-determined threshold value is also adjusted accordingly by shifting down the threshold value derived at unconstrained bit-depth of the interpolated samples.

[0054] FIG. 6 shows a simplified flowchart of a method using decoder-side verification, checking or determination of an alignment level between motion compensated patches according to an embodiment of the present invention, where the conditional skipping of the decoder-side motion vector refinement process steps is based on a mean-removed sum of absolute differences of the refinement start position. The steps shown in the flowchart may be implemented as program codes or instructions executable on one or more processors at the encoder side or the decoder side. The steps shown in the flowchart may also be implemented using electronic components, digital and/or analog circuitry, logic elements, hardware, software, firmware, or combinations thereof. The method may begin, at step 602, a video decoder obtains motion compensated interpolated samples based on sub-pixel accurate merge motion vectors between two motion compensated reference patches. At step 603, the decoder computes (calculates) a mean value of each of the patches for the center position at which the refinement starts. The decoder also computes (at 603) a mean-removed sum of absolute differences (MR-SAD) for the center position.

[0055] At 604, the decoder compares the MR-SAD against a coding unit (CU) size-dependent threshold value to determine whether the MR-SAD is less than or not less than (equal to or greater than) the CU size-dependent threshold value. In one embodiment, the CU size-dependent threshold value is a function of the bit depth of the sub-pixel accurate merge motion vectors.

- 5 When the MR-SAD is determined to be less than the CU size-dependent threshold value (“yes” in step 605), the decoder skips the decoder-side motion vector refinement (DMVR) process steps (step 606) and performs the final motion compensation using the merge motion vectors (step 608). When the MR-SAD is determined to be not less than the CU size-dependent threshold value (“no” in step 605), the decoder performs the decoder-side motion vector refinement
- 10 (DMVR) process steps using the merge motion vectors (step 607), and then performs the final motion compensation using the merge motion vectors (step 608). The decoder may refine the motion vector candidates based on a reference block in a search window that closely matches the current block, i.e., the decoder may determine a new interpolated motion vector (e.g., using bilinear interpolation) for the current block when the MR-SAD is not less than the CU size-
- 15 dependent threshold value, thereafter, the decoder then performs the final motion compensation for the current block using the merge motion vectors.

[0056] In some embodiments, computing the mean value of a patch may include averaging the luma intensity of all of the pixels within the corresponding reference patch. The mean-removed sum of absolute differences (MR-SAD) may be computed using the following expression:

20
$$\text{MR-SAD} = \text{SAD}(\text{first_sample} - \text{mean}(\text{first_patch}), \text{second_sample} - \text{mean}(\text{second_patch}))$$

- [0057] Where first_sample represents the first motion compensated interpolated sample at the center of the first motion compensated reference patch, the second motion compensated interpolated sample at the center of the second motion compensated reference patch, mean(first_block) represents an averaging operation of sample values of the first_block,
- 25 mean(second_block) represents an averaging operation of sample values of the second_block, the (first, second) patch may have a width and height of the (first, second) block in terms of samples, and SAD is a summing operation of absolute differences of the values of samples of the first patch and the samples of the second patch.

- [0058] This embodiment can ensure that no additional computations are incurred for
- 30 performing the conditional skipping of refinement check over what is required without such a

check. Statistically, the computations of the refinement iterations are saved which can result in power savings and average cycles reduction in software.

[0059] The flowcharts shown and described above are intended to illustrate exemplary embodiments of determination, checking or verification of the alignment between motion compensated patches according to the present invention. As one skilled in the art knows, the alignment checking and determination steps described herein may be re-arranged, modified, or combined to practice the present invention without departing from the scope of the present invention.

[0060] Embodiments in accordance with the present disclosure may be implemented in hardware circuitry, software program codes, computer instructions executable by one or more processors or CPUs, or combination thereof. For example, embodiments of the present invention can be one or more integrated circuits or devices that perform the steps described herein. The program codes can be executed on a digital signal processor (DSP). Embodiments can also be implemented by one or microprocessors, one or more field programmable gate array (FPGAs).

[0061] Embodiments in accordance with the present disclosure provide an apparatus for determining the level of alignment between motion compensated reference patches (coding units, blocks) against a predetermined CU size-dependent threshold value. The apparatus may include circuitry configured to obtain unrounded merge motion vectors between two motion compensated reference patches; round the unrounded merge motion vectors to the nearest integer sample position to obtain rounded motion vectors; compute the sum of absolute differences (SAD) between the two motion compensated reference patches using the rounded motion vectors; and determine whether the SAD is less than a CU size-dependent threshold value. In the event that the SAD is less than the CU size-dependent threshold value, the circuitry of the apparatus skips remaining steps of a decoder-side motion vector refinement process and perform the final motion compensation using the unrounded merge motion vectors. In the event that the SAD is not less than the CU size-dependent threshold value, the circuitry of the apparatus performs the remaining steps of a decoder-side motion vector refinement process, and then perform the final motion compensation using the unrounded merge motion vectors.

[0062] In one embodiment, the circuitry of the apparatus derives the SAD using a subset of motion compensated interpolated samples instead of using the rounded motion vectors.

[0063] In one embodiment, the circuitry of the apparatus computes a mean-removed sum of absolute differences (MR-SAD) for the integer pixel motion search and compare the MR-SAD against the CU size-dependent threshold value.

[0064] In some embodiments, the circuitry of the apparatus may include an integrated semiconductor device or chip. In other embodiments, the circuitry may include hardware components or device or a combination of hardware and software, e.g., one or more processors, programmable devices, or DSPs operable with instructions or program codes.

[0065] Following is an explanation of the applications of the encoding method as well as the decoding method as shown in the above-mentioned embodiments, and a system using them.

[0066] FIG. 7 is a block diagram showing a content supply system 3100 for realizing content distribution service. This content supply system 3100 includes capture device 3102, terminal device 3106, and optionally includes display 3126. The capture device 3102 communicates with the terminal device 3106 over communication link 3104. The communication link may include the communication channel 13 described above. The communication link 3104 includes but not limited to WIFI, Ethernet, Cable, wireless (3G/4G/5G), USB, or any kind of combination thereof, or the like.

[0067] The capture device 3102 generates data, and may encode the data by the encoding method as shown in the above embodiments. Alternatively, the capture device 3102 may distribute the data to a streaming server (not shown in the Figures), and the server encodes the data and transmits the encoded data to the terminal device 3106. The capture device 3102 includes but not limited to camera, smart phone or Pad, computer or laptop, video conference system, PDA, vehicle mounted device, or a combination of any of them, or the like. For example, the capture device 3102 may include the source device 12 as described above. When the data includes video, the video encoder 20 included in the capture device 3102 may actually perform video encoding processing. When the data includes audio (i.e., voice), an audio encoder included in the capture device 3102 may actually perform audio encoding processing. For some practical scenarios, the capture device 3102 distributes the encoded video and audio data by multiplexing them together. For other practical scenarios, for example in the video conference system, the encoded audio data and the encoded video data are not multiplexed. Capture device 3102

distributes the encoded audio data and the encoded video data to the terminal device 3106 separately.

[0068] In the content supply system 3100, the terminal device 310 receives and reproduces the encoded data. The terminal device 3106 could be a device with data receiving and recovering capability, such as smart phone or Pad 3108, computer or laptop 3110, network video recorder (NVR)/ digital video recorder (DVR) 3112, TV 3114, set top box (STB) 3116, video conference system 3118, video surveillance system 3120, personal digital assistant (PDA) 3122, vehicle mounted device 3124, or a combination of any of them, or the like capable of decoding the above-mentioned encoded data. For example, the terminal device 3106 may include the destination device 14 as described above. When the encoded data includes video, the video decoder 30 included in the terminal device is prioritized to perform video decoding. When the encoded data includes audio, an audio decoder included in the terminal device is prioritized to perform audio decoding processing.

[0069] For a terminal device with its display, for example, smart phone or Pad 3108, computer or laptop 3110, network video recorder (NVR)/ digital video recorder (DVR) 3112, TV 3114, personal digital assistant (PDA) 3122, or vehicle mounted device 3124, the terminal device can feed the decoded data to its display. For a terminal device equipped with no display, for example, STB 3116, video conference system 3118, or video surveillance system 3120, an external display 3126 is contacted therein to receive and show the decoded data.

[0070] When each device in this system performs encoding or decoding, the picture encoding device or the picture decoding device, as shown in the above-mentioned embodiments, can be used.

[0071] FIG. 8 is a diagram showing a structure of an example of the terminal device 3106. After the terminal device 3106 receives stream from the capture device 3102, the protocol proceeding unit 3202 analyzes the transmission protocol of the stream. The protocol includes but not limited to Real Time Streaming Protocol (RTSP), Hyper Text Transfer Protocol (HTTP), HTTP Live streaming protocol (HLS), MPEG-DASH, Real-time Transport protocol (RTP), Real Time Messaging Protocol (RTMP), or any kind of combination thereof, or the like.

[0072] After the protocol proceeding unit 3202 processes the stream, stream file is generated. The file is outputted to a demultiplexing unit 3204. The demultiplexing unit 3204 can separate the multiplexed data into the encoded audio data and the encoded video data. As described above, for some practical scenarios, for example in the video conference system, the encoded audio data and the encoded video data are not multiplexed. In this situation, the encoded data is transmitted
5 to video decoder 3206 and audio decoder 3208 without through the demultiplexing unit 3204.

[0073] Via the demultiplexing processing, video elementary stream (ES), audio ES, and optionally subtitle are generated. The video decoder 3206, which includes the video decoder 30 as explained in the above mentioned embodiments, decodes the video ES by the decoding
10 method as shown in the above-mentioned embodiments to generate video frame, and feeds this data to the synchronous unit 3212. The audio decoder 3208, decodes the audio ES to generate audio frame, and feeds this data to the synchronous unit 3212. Alternatively, the video frame may store in a buffer (not shown in FIG. Y) before feeding it to the synchronous unit 3212. Similarly, the audio frame may store in a buffer (not shown in FIG. Y) before feeding it to the
15 synchronous unit 3212.

[0074] The synchronous unit 3212 synchronizes the video frame and the audio frame, and supplies the video/audio to a video/audio display 3214. For example, the synchronous unit 3212 synchronizes the presentation of the video and audio information. Information may code in the syntax using time stamps concerning the presentation of coded audio and visual data and time
20 stamps concerning the delivery of the data stream itself.

[0075] If subtitle is included in the stream, the subtitle decoder 3210 decodes the subtitle, and synchronizes it with the video frame and the audio frame, and supplies the video/audio/subtitle to a video/audio/subtitle display 3216.

[0076] The present invention is not limited to the above-mentioned system, and either the
25 picture encoding device or the picture decoding device in the above-mentioned embodiments can be incorporated into other system, for example, a car system.

WHAT IS CLAIMED IS:

1. A method of motion compensation for a block of a frame in a video with a decoder-side motion vector refinement (DMVR) process comprising:
 - obtaining, by a decoder, motion compensated interpolated samples in two reference frames of the block based on sub-pixel accurate merge motion vectors of the block
 - 5 for the two reference frames via a bilinear motion compensated interpolation;
 - computing, by the decoder, a sum of absolute differences (SAD) between two motion compensated reference patches using a subset of the motion compensated interpolated samples;
 - determining, by the decoder, whether the SAD is less than a coding unit (CU)
 - 10 size-dependent threshold value, the CU threshold value depending on a size of the CU;
 - when the SAD is less than the CU size-dependent threshold value:
 - skipping remaining steps of the DMVR process for the block; and
 - performing final motion compensation using the sub-pixel accurate merge motion vectors of the block to obtain reconstructed samples for the block; and
 - 15 when the SAD is not less than the CU size-dependent threshold value:
 - performing the remaining steps of the DMVR process to determine refined motion vectors for the block; and
 - performing the final motion compensation using the refined motion vectors to obtain the reconstructed samples for the block.
2. The method of claim 1, wherein the subset of the motion compensated interpolated samples are generated based on unrounded merge motion vectors associated with the two motion compensated reference patches.
3. The method of claim 1, wherein the bilinear motion compensated interpolation for generating the motion compensated interpolated samples in the two reference frames are performed in parallel .

4. The method of claim 1, wherein the motion compensated interpolated samples are generated in an interleaved manner.

5. The method of claim 1, wherein the subset of the motion compensated interpolated samples comprises a block of samples having a dimension equal to a dimension of a coding unit (CU).

6. The method of claim 5, wherein the subset of the motion compensated interpolated samples comprises $(CU_width - 2) * (CU_height - 2)$ samples, wherein CU_width is a width of the coding unit and CU_height is a height of the coding unit.

7. The method of claim 5, wherein the subset of the motion compensated interpolated samples comprises $(CU_width * CU_height - (CU_width - 2) * (CU_height - 2))$ samples, wherein CU_width is a width of the coding unit and CU_height is a height of the coding unit.

8. The method of claim 1, wherein the CU size-dependent threshold value is a function of a number of samples in the subset of motion compensated interpolated samples.

9. A decoder, performing motion compensation for a block of a frame in a video with a decoder-side motion vector refinement (DMVR) process, the decoder comprising:

one or more processors; and

a non-transitory computer-readable storage medium coupled to the processors and storing programming for execution by the processors, the programming, when executed by the processors, configuring the decoder to:

obtain motion compensated interpolated samples in two reference frames of the block based on sub-pixel accurate merge motion vectors of the block for the two reference frames via a bilinear motion compensated interpolation;

compute a sum of absolute differences (SAD) between two motion compensated reference patches using a subset of the motion compensated interpolated samples;

determine whether the SAD is less than a coding unit (CU) size-dependent threshold value, the CU threshold value depending on a size of the CU;

when the SAD is less than the CU size-dependent threshold value:

skip remaining steps of the DMVR process for the block; and

perform final motion compensation using the sub-pixel accurate merge motion vectors of the block to obtain reconstructed samples for the block; and

when the SAD is not less than the CU size-dependent threshold value:

perform the remaining steps of the DMVR process to determine refined motion vectors for the block; and

perform the final motion compensation using the refined motion vectors to obtain the reconstructed samples for the block.

10. The decoder of claim 9, wherein the subset of the motion compensated interpolated samples are generated based on unrounded merge motion vectors associated with the two motion compensated reference patches.

11. The decoder of claim 9, wherein the bilinear motion compensated interpolation for generating the motion compensated interpolated samples in the two reference frames are performed in parallel.

12. The decoder of claim 9, wherein the motion compensated interpolated samples are generated in an interleaved manner.

13. The decoder of claim 9, wherein the subset of the motion compensated interpolated samples comprises a block of samples having a dimension equal to a dimension of a coding unit (CU).

14. The decoder of claim 13, wherein the subset of the motion compensated interpolated samples comprises $(\text{CU_width} - 2) * (\text{CU_height} - 2)$ samples, wherein CU_width is a width of the coding unit and CU-height is a height of the coding unit.

15. The decoder of claim 13, wherein the subset of the motion compensated interpolated samples comprises $(\text{CU_width} * \text{CU_height} - (\text{CU_width} - 2) * (\text{CU_height} - 2))$ samples, wherein CU_width is a width of the coding unit and CU-height is a height of the coding unit.

16. The decoder of claim 9, wherein the CU size-dependent threshold value is a function of a number of samples in the subset of motion compensated interpolated samples.

17. A non-transitory computer-readable storage medium comprising instructions which, when the instructions are executed by a computer, cause the computer to carry out motion compensation for a block of a frame in a video with a decoder-side motion vector refinement (DMVR) process comprising:

obtaining motion compensated interpolated samples in two reference frames of the block based on sub-pixel accurate merge motion vectors of the block for the two reference frames via a bilinear motion compensated interpolation;

computing a sum of absolute differences (SAD) between two motion compensated reference patches using a subset of the motion compensated interpolated samples;

determining whether the SAD is less than a coding unit (CU) size-dependent threshold value, the CU threshold value depending on a size of the CU;

when the SAD is less than the CU size-dependent threshold value:

skipping remaining steps of the DMVR process for the block; and

performing final motion compensation using the sub-pixel accurate merge motion vectors of the block to obtain reconstructed samples for the block; and

when the SAD is not less than the CU size-dependent threshold value:

performing the remaining steps of the DMVR process to determine refined motion vectors for the block; and

performing the final motion compensation using the refined motion vectors to obtain the reconstructed samples for the block.

18. The non-transitory computer-readable storage medium of claim 17, wherein the subset of the motion compensated interpolated samples are generated based on unrounded merge motion vectors associated with the two motion compensated reference patches.

19. The non-transitory computer-readable storage medium of claim 17, wherein the bilinear motion compensated interpolation for generating the motion compensated interpolated samples in the two reference frames are performed in parallel.

20. The non-transitory computer-readable storage medium of claim 17, wherein the motion compensated interpolated samples are generated in an interleaved manner.