

SIMPLIFIED INTER PREDICTION WITH GEOMETRIC PARTITIONING

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] Under the applicable patent law and/or rules pursuant to the Paris Convention, this application is made to timely claim the priority to and benefits of International Patent Application No. PCT/CN2019/122256, filed on November 30, 2019, International Patent Application No. PCT/CN2020/071032, filed on January 8, 2020, and International Patent Application No. PCT/CN2020/071552, filed on January 10, 2020. For all purposes under the law, the entire disclosures of the aforementioned applications are incorporated by reference as part of the disclosure of this application.

TECHNICAL FIELD

[0002] This patent document relates to video coding and decoding techniques, devices and systems.

BACKGROUND

[0003] Currently, efforts are underway to improve the performance of current video codec technologies to provide better compression ratios or provide video coding and decoding schemes that allow for lower complexity or parallelized implementations. Industry experts have recently proposed several new video coding tools and tests are currently underway for determining their effectivity.

SUMMARY

[0004] Devices, systems and methods related to digital video coding, and specifically, to management of motion vectors are described. The described methods may be applied to existing video coding standards (e.g., High Efficiency Video Coding (HEVC) or Versatile Video Coding) and future video coding standards or video codecs.

[0005] In one representative aspect, the disclosed technology may be used to provide a method for video processing. This method includes performing a conversion between a current video block of a video and a bitstream representation of the video, wherein, during the conversion, a use of a geometric partitioning mode is allowed for the current video block, and wherein parameters of the geometric partitioning mode are computed using a set of angles including a first number of angles that is less than a first threshold value and/or a set of distances including a second number of distances that is less than a second threshold value.

[0006] In another representative aspect, the disclosed technology may be used to provide another method for video processing. This method includes determining, for a conversion between a current video block of a video and a bitstream representation of the video, an applicability of a geometric partitioning mode based on a rule that depends on one or more dimensions of the current video block and/or a mathematical function of the one or more dimensions of the current video; and performing the conversion based on the determining.

[0007] In yet another representative aspect, the disclosed technology may be used to provide yet another method for video processing. This method includes determining, for a conversion between a current video block of a video and a bitstream representation of the video, an applicability of a geometric partitioning mode according to a rule that depends on coding characteristics of the current video block; and performing the conversion based on the determining.

[0008] In another representative aspect, the disclosed technology may be used to provide another method for video processing. This method includes determining that a rule of exclusion is applicable to a conversion between a current video block of a video and a bitstream representation of the video, wherein the rule of exclusion specifies that the conversion disallows using a geometric partitioning mode and a coding tool together for the current video block; and performing the conversion based on the determining.

[0009] In another representative aspect, the disclosed technology may be used to provide another method for video processing. This method includes determining, for a conversion between a video and a bitstream representation of the video, different geometric partitioning mode indices for different color components of the video according to a rule; and performing the conversion based on the determining.

[0010] In another representative aspect, the disclosed technology may be used to provide another method for video processing. This method includes determining, for a conversion between a current video block of a current picture of a video and a bitstream representation of the video, an applicability of a geometric partitioning mode according to a rule; and performing the conversion based on the determining, wherein the geometric partitioning mode includes splitting the current video block into two or more prediction sub-regions, and wherein the rule depends on a resolution of one or more reference pictures associated with the two or more prediction sub-regions.

[0011] In another representative aspect, the disclosed technology may be used to provide another method for video processing. This method includes performing a conversion between a current video block of a video and a bitstream representation of the video, wherein the bitstream representation conforms to a format rule, wherein the format rule specifies that the bitstream representation omits a syntax element related to a geometric partitioning mode in a case that the geometric partitioning mode is disabled for the current video block.

[0012] In another representative aspect, the disclosed technology may be used to provide another method for video processing. This method includes determining, for a conversion between a video unit comprising one or more video blocks of a video and a bitstream representation of the video, that multiple sets of geometric partitioning modes are allowed for coding the one or more video blocks of the video unit according to a rule; and performing the conversion based on the determining.

[0013] In another representative aspect, the disclosed technology may be used to provide another method for video processing. This method includes performing a conversion between a video including video blocks and a bitstream representation of the video, wherein, during the conversion, a first count of geometric partitioning modes are available for computing a partition angle index and/or a partition distance index, a second count of geometric partitioning modes are available for coding in the bitstream representation, and a third count of geometric partitioning modes are available for signaling in the bitstream representation of, wherein the first count and/or the second count and/or the third count are based at least on corresponding dimensions of the video blocks.

[0014] In another representative aspect, the disclosed technology may be used to provide another method for video processing. This method includes performing a conversion between a current video block of a video and a bitstream representation of the video according to a rule, wherein the rule specifies that a first geometric partitioning mode index value is signaled in the bitstream representation of the current video block and a second geometric partitioning mode index value is used for computing a partition angle index and/or a partition distance index of the current video block, and wherein the first geometric partitioning mode index value is different from the second geometric partitioning mode index value.

[0015] In another representative aspect, the disclosed technology may be used to provide another method for video processing. This method includes performing a conversion between a current video block of a video and a bitstream representation of the video, wherein a geometric partitioning mode

index of the current video block is coded in the bitstream such that a binarization of the geometric partitioning mode index is performed according to a rule.

[0016] In another representative aspect, the disclosed technology may be used to provide another method for video processing. This method includes determining, for a conversion between a chroma block of a video unit of a video and a bitstream representation of the video, blending weights and/or motion storage weights for the chroma block according to a rule; and performing the conversion based on the determining, wherein a geometric partitioning mode is applied to the chroma block, and wherein the rule depends on a chroma sample location type indicating a relative position of a certain chroma sample in the chroma block with respect to a corresponding luma sample.

[0017] Further, in a representative aspect, an apparatus in a video system comprising a processor and a non-transitory memory with instructions thereon is disclosed. The instructions upon execution by the processor, cause the processor to implement any one or more of the disclosed methods.

[0018] Also, a computer program product stored on a non-transitory computer readable media, the computer program product including program code for carrying out any one or more of the disclosed methods is disclosed.

[0019] The above and other aspects and features of the disclosed technology are described in greater detail in the drawings, the description and the claims.

BRIEF DESCRIPTION OF THE DRAWINGS

[0020] FIG. 1 shows example positions of a spatial merge candidate.

[0021] FIG. 2 shows an example of candidate pairs considered for redundancy check of spatial merge candidates.

[0022] FIG. 3 shows an example of motion vector scaling for a temporal merge candidate.

[0023] FIG. 4 shows example candidate positions for a temporal merge candidate.

[0024] FIG. 5 shows an example of a triangle partition based inter prediction mode.

[0025] FIG. 6 shows an example of uni-prediction motion vector selection for a triangle partition mode.

[0026] FIG. 7 shows examples of weights used in a blending process for a luma block and a chroma block respectively.

[0027] FIG. 8 shows an example of existing and proposed shapes in connection with a triangle prediction mode.

[0028] FIG. 9 shows an example of a geometric merge mode (GEO) split boundary description.

[0029] FIG. 10A shows an example illustration of edges supported in GEO.

[0030] FIG. 10B shows an example of geometric relations between a given pixel position and two edges.

[0031] FIG. 11 shows an example of proposed angles for GEO along with their corresponding width to height ratio.

[0032] FIG. 12 shows example locations of the top-left chroma sample as a function of a ChromaLocType variable.

[0033] FIG. 13A is a block diagram of an example of a hardware platform for implementing a visual media decoding or a visual media encoding technique described in the present document.

[0034] FIG. 13B is a block diagram of another example of a hardware platform for implementing a visual media decoding or a visual media encoding technique described in the present document.

[0035] FIG. 14 shows a flowchart of an example method for video processing.

[0036] FIG. 15 is a block diagram that illustrates a video coding system in accordance with some embodiments of the present disclosure.

[0037] FIG. 16 is a block diagram that illustrates an encoder in accordance with some embodiments of the present disclosure.

[0038] FIG. 17 is a block diagram that illustrates a decoder in accordance with some embodiments of the present disclosure.

[0039] FIGS. 18A to 18H are block diagrams of example methods for video processing based on some implementations of the disclosed technology.

[0040] FIG. 19 shows a table indicating specification of angleIdx and distanceIdx values.

[0041] FIG. 20A shows an old version of Table 36 indicating specification of angleIdx and distanceIdx values based on merge_geo_partition_idx value, which is now deleted from the relevant working draft.

[0042] FIGS. 20B to 20G show examples of Table 36 indicating specification of angleIdx and distanceIdx values based on the merge_geo_partition_idx value.

[0043] FIG. 21 shows Table 8-10 corresponding to the example embodiment No. 3 and indicating specification of angleIdx and distanceIdx values based on wedge_partition_idx' value.

[0044] FIG. 22 shows Table 8-10 corresponding to the example embodiment No. 4 and indicating specification of angleIdx and distanceIdx values based on the wedge_partition_idx' value.

[0045] FIG. 23A shows an old version of Table 8-10 which is now deleted from the relevant working draft.

[0046] FIG. 23B shows an example of Table 8-10 indicating specification of angleIdx and distanceIdx values based on the wedge_partition_idx value.

[0047] FIG. 24A shows a table indicating mapping table of geo_partition_idx' values based on geo_partition_idx value.

[0048] FIG. 24B shows an old version of Table 36 indicating specification of angleIdx and distanceIdx values based on geo_partition_idx value, which is now deleted from the relevant working draft.

[0049] FIGS. 24C to 24E show Table 36 indicating specification of angleIdx and distanceIdx values based on geo_partition_idx value.

DETAILED DESCRIPTION

[0050] Video coding in HEVC/H.265

[0051] Video coding standards have evolved primarily through the development of the well-known ITU-T and ISO/IEC standards. The ITU-T produced H.261 and H.263, ISO/IEC produced MPEG-1 and MPEG-4 Visual, and the two organizations jointly produced the H.262/MPEG-2 Video and H.264/MPEG-4 Advanced Video Coding (AVC) and H.265/HEVC standards. Since H.262, the video coding standards are based on the hybrid video coding structure wherein temporal prediction plus transform coding are utilized. To explore the future video coding technologies beyond HEVC, Joint Video Exploration Team (JVET) was founded by VCEG and MPEG jointly in 2015. Since then, many new methods have been adopted by JVET and put into the reference software named Joint Exploration Model (JEM). The JVET meeting is concurrently held once every quarter, and the new coding standard is targeting at 50% bitrate reduction as compared to HEVC. The new video coding standard was officially named as Versatile Video Coding (VVC) in the April 2018 JVET meeting, and the first version of VVC test model (VTM) was released at that time. As there are continuous effort contributing to VVC standardization, new coding techniques are being adopted to the VVC standard in every JVET meeting. The VVC working draft and test model VTM are then updated after every meeting. The VVC project is now aiming for technical completion (FDIS) at the July 2020 meeting.

[0052] 2.1. Extended merge prediction

[0053] In VTM, the merge candidate list is constructed by including the following five types of candidates in order:

- 1) Spatial MVP from spatial neighbour CUs
- 2) Temporal MVP from collocated CUs
- 3) History-based MVP from an FIFO table
- 4) Pairwise average MVP
- 5) Zero MVs.

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

The generation process of each category of merge candidates is provided in this session.

[0054] 2.1.1. Spatial candidates derivation

[0055] The derivation of spatial merge candidates in VVC is same to that in HEVC. A maximum of four merge candidates are selected among candidates located in the positions depicted in Fig. 1. The order of derivation is A_0 , B_0 , B_1 , A_1 and B_2 . Position B_2 is considered only when any CU of position A_0 , B_0 , B_1 , A_1 is not available (e.g. because it belongs to another slice or tile) or is intra coded. After candidate at position A_1 is added, the addition of the remaining candidates is subject to a redundancy check which ensures that candidates with same motion information are excluded from the list so that coding efficiency is improved. To reduce computational complexity, not all possible candidate pairs are considered in the mentioned redundancy check. Instead only the pairs linked with an arrow in Fig. 2 are considered and a candidate is only added to the list if the corresponding candidate used for redundancy check has not the same motion information.

[0056] 2.1.2. Temporal candidates derivation

[0057] In this step, only one candidate is added to the list. Particularly, in the derivation of this temporal merge candidate, a scaled motion vector is derived based on co-located CU belonging to the collocated reference picture. The reference picture list to be used for derivation of the co-located CU is explicitly signalled in the slice header. The scaled motion vector for temporal merge candidate is obtained as illustrated by the dotted line in Fig. 3, which is scaled from the motion vector of the co-

located CU using the POC distances, tb and td , where tb is defined to be the POC difference between the reference picture of the current picture and the current picture and td is defined to be the POC difference between the reference picture of the co-located picture and the co-located picture. The reference picture index of temporal merge candidate is set equal to zero.

[0058] The position for the temporal candidate is selected between candidates C_0 and C_1 , as depicted in Fig. 4. If CU at position C_0 is not available, is intra coded, or is outside of the current row of CTUs, position C_1 is used. Otherwise, position C_0 is used in the derivation of the temporal merge candidate.

[0059] 2.1.3. History-based merge candidates derivation

[0060] The history-based MVP (HMVP) merge candidates are added to merge list after the spatial MVP and TMVP. In this method, the motion information of a previously coded block is stored in a table and used as MVP for the current CU. The table with multiple HMVP candidates is maintained during the encoding/decoding process. The table is reset (emptied) when a new CTU row is encountered. Whenever there is a non-subblock inter-coded CU, the associated motion information is added to the last entry of the table as a new HMVP candidate.

[0061] In VTM the HMVP table size S is set to be 6, which indicates up to 6 History-based MVP (HMVP) candidates may be added to the table. When inserting a new motion candidate to the table, a constrained first-in-first-out (FIFO) rule is utilized wherein redundancy check is firstly applied to find whether there is an identical HMVP in the table. If found, the identical HMVP is removed from the table and all the HMVP candidates afterwards are moved forward.

[0062] HMVP candidates could be used in the merge candidate list construction process. The latest several HMVP candidates in the table are checked in order and inserted to the candidate list after the TMVP candidate. Redundancy check is applied on the HMVP candidates to the spatial or temporal merge candidate.

[0063] To reduce the number of redundancy check operations, the following simplifications are introduced:

1. Number of HMPV candidates is used for merge list generation is set as $(N \leq 4) ? M : (8 - N)$, wherein N indicates number of existing candidates in the merge list and M indicates number of available HMVP candidates in the table.
2. Once the total number of available merge candidates reaches the maximally allowed merge candidates minus 1, the merge candidate list construction process from HMVP is terminated.

[0064] 2.1.4. Pair-wise average merge candidates derivation

[0065] Pairwise average candidates are generated by averaging predefined pairs of candidates in the existing merge candidate list, and the predefined pairs are defined as $\{(0, 1), (0, 2), (1, 2), (0, 3), (1, 3), (2, 3)\}$, where the numbers denote the merge indices to the merge candidate list. The averaged motion vectors are calculated separately for each reference list. If both motion vectors are available in one list, these two motion vectors are averaged even when they point to different reference pictures; if only one motion vector is available, use the one directly; if no motion vector is available, keep this list invalid.

[0066] When the merge list is not full after pair-wise average merge candidates are added, the zero MVPs are inserted in the end until the maximum merge candidate number is encountered.

[0067] 2.2. Triangle partition for inter prediction

[0068] In VTM, a triangle partition mode (TPM) is supported for inter prediction. The triangle partition mode is only applied to CUs that are 64 samples or larger and are coded in skip or merge mode but not in a regular merge mode, or MMVD mode, or CIIP mode or subblock merge mode. A CU-level flag is used to indicate whether the triangle partition mode is applied or not.

[0069] When this mode is used, a CU is split evenly into two triangle-shaped partitions, using either the diagonal split or the anti-diagonal split (Fig. 5). Each triangle partition in the CU is inter-predicted using its own motion; only uni-prediction is allowed for each partition, that is, each partition has one motion vector and one reference index. The uni-prediction motion constraint is applied to ensure that same as the conventional bi-prediction, only two motion compensated prediction are needed for each CU. The uni-prediction motion for each partition is derived directly the merge candidate list constructed for extended merge prediction in 2.1, and the selection of a uni-prediction motion from a given merge candidate in the list is according to the procedure in 2.2.1.

[0070] If triangle partition mode is used for a current CU, then a flag indicating the direction of the triangle partition (diagonal or anti-diagonal), and two merge indices (one for each partition) are further signalled. After predicting each of the triangle partitions, the sample values along the diagonal or anti-diagonal edge are adjusted using a blending processing with adaptive weights. This is the prediction signal for the whole CU, and transform and quantization process will be applied to the whole CU as in other prediction modes. Finally, the motion field of a CU predicted using the triangle partition mode is stored in 4x4 units as in 2.2.3.

[0071] 2.2.1. Uni-prediction candidate list construction

[0072] Given a merge candidate index, the uni-prediction motion vector is derived from the merge candidate list constructed for extended merge prediction using the process in 2.1, as exemplified in Fig. 6. For a candidate in the list, its LX motion vector with X equal to the parity of the merge candidate index value, is used as the uni-prediction motion vector for triangle partition mode. These motion vectors are marked with “x” in Fig. 6. In case a corresponding LX motion vector does not exist, the L(1-X) motion vector of the same candidate in the extended merge prediction candidate list is used as the uni-prediction motion vector for triangle partition mode.

[0073] 2.2.2. Blending along the triangle partition edge

[0074] After predicting each triangle partition using its own motion, blending is applied to the two prediction signals to derive samples around the diagonal or anti-diagonal edge. The following weights are used in the blending process:

7/8, 6/8, 5/8, 4/8, 3/8, 2/8, 1/8} for luma and {6/8, 4/8, 2/8} for chroma, as shown in Fig. 7.

[0075] 2.2.3. Motion field storage

[0076] The motion vectors of a CU coded in triangle partition mode are stored in 4x4 units. Depending on the position of each 4x4 unit, either uni-prediction or bi-prediction motion vectors are stored. Denote Mv1 and Mv2 as uni-prediction motion vectors for partition 1 and partition 2, respectively. If a 4x4 unit is located in the non-weighted area shown in the example of Fig. 7, either Mv1 or Mv2 is stored for that 4x4 unit. Otherwise, if the 4x4 unit is located in the weighted area, a bi-prediction motion vector is stored. The bi-prediction motion vector is derived from Mv1 and Mv2 according to the following process:

- 1) If Mv1 and Mv2 are from different reference picture lists (one from L0 and the other from L1), then Mv1 and Mv2 are simply combined to form the bi-prediction motion vector.
- 2) Otherwise, if Mv1 and Mv2 are from the same list, and without loss of generality, assume they are both from L0. In this case,
 - 2.a) If the reference picture of either Mv2 (or Mv1) appears in L1, then that Mv2 (or Mv1) is converted to a L1 motion vector using that reference picture in L1. Then the two motion vectors are combined to form the bi-prediction motion vector;

Otherwise, instead of bi-prediction motion, only uni-prediction motion Mv1 is stored.

[0077] 2.3. Geometrical partitioning (GEO) for inter prediction

[0078] The following description is extracted from JVET-P0884, JVET-P0107, JVET-P0304 and JVET-P0264.

[0079] Geometric merge mode (GEO) was proposed in 15th Gothenburg JVET meeting as extension of the existing triangle prediction mode (TPM). In 16th Geneva JVET meeting, a simpler designed GEO mode in JVET-P0884 has been selected as a CE anchor for further study. Nowadays GEO mode is being studied as a replacement of the existing TPM in VVC.

[0080] Fig. 8 illustrates TPM in VTM-6.0 and additional shapes proposed for non-rectangular inter blocks.

[0081] The split boundary of geometric merge mode is described by angle φ_i and distance offset ρ_i as shown in Fig. 9. Angle φ_i represents a quantized angle between 0 and 360 degrees and distance offset ρ_i represents a quantized offset of the largest distance ρ_{max} . In addition, the split directions overlapped with binary tree splits and TPM splits are excluded.

[0082] In JVET-P0884, GEO is applied to block sizes not smaller than 8x8, and for each block size, there are 82 different partitioning manners, differentiated by 24 angles and 4 edges relative to the center of a CU. Fig. 10A shows that the 4 edges are distributed uniformly along the direction of normal vector within a CU, starting from Edge0 that passes through the CU center. Each partition mode (i.e., a pair of an angle index and an edge index) in GEO is assigned with a pixel-adaptive weight table to blend samples on the two partitioned parts, where the weight value of a sampled ranges from 0 to 8 and is determined by the L2 distance from the center position of a pixel to the edge. Basically, unit-gain constraint is followed when weight values are assigned, that is, when a small weight value is assigned to a GEO partition, a large complementary one is assigned to the other partition, summing up to 8.

[0083] The computation of the weight value of each pixel is two-fold: (a) computing the displacement from a pixel position to a given edge and (c) mapping the computed displacement to a weight value through a pre-defined look-up table. The way to compute the displacement from a pixel position (x, y) to a given edge Edge_i is actually the same as computing the displacement from (x, y) to Edge0 and subtract this displacement by the distance ρ between Edge0 and Edge_i. Fig. 10B illustrates the geometric relations among (x, y) and edges. Specifically, the displacement from (x, y) to Edge_i can be formulated as follows:

$$\overline{PB} = \overline{PA} - \rho \quad (1)$$

$$= (\overline{PC} + \overline{CD}) * \sin(\alpha) - \rho \quad (2)$$

$$= \left(\left(\frac{H}{2} - y \right) + \left(x - \frac{W}{2} \right) * \cot(\alpha) \right) * \sin(\alpha) - \rho \quad (3)$$

$$= \left(x - \frac{W}{2} \right) * \cos(\alpha) - \left(y - \frac{H}{2} \right) * \sin(\alpha) - \rho \quad (4)$$

$$= x * \cos(\alpha) - y * \sin(\alpha) - \left(\rho + \frac{W}{2} * \cos(\alpha) - \frac{H}{2} * \sin(\alpha) \right) \quad (5)$$

$$= x * \cos(\alpha) + y * \cos\left(\alpha + \frac{\pi}{2}\right) - \left(\rho + \frac{W}{2} * \cos(\alpha) + \frac{H}{2} * \cos\left(\alpha + \frac{\pi}{2}\right) \right). \quad (6)$$

[0084] The value of ρ is a function of the maximum length (denoted by ρ_{max}) of the normal vector and edge index i , that is:

$$\rho = i * (\rho_{max} - 1) / N \quad (7)$$

$$= i * \left(\left(\frac{H}{2} * \tan(\alpha) + \frac{W}{2} \right) * \cos(\alpha) - 1 \right) / N, \quad (8)$$

where N is the number of edges supported by GEO and the “1” is to prevent the last edge EdgeN-1 from falling too close to a CU corner for some angle indices. Substituting Eq. (8) from (6), we can compute the displacement from each pixel (x,y) to a given Edge i . In short, we denote $\overline{PB}$ as $wIdx(x,y)$. The computation of ρ is needed once per CU, and the computation of $wIdx(x,y)$ is needed once per sample, in which multiplications are involved.

[0085] 2.3.1. JVET-P0884

[0086] JVET-P0884 jointed the proposal jointed the proposed simplification of JVET-P0107 slope based version 2, JVET-P0304 and JVET-P0264 test 1 on top of CE4-1.14 of the 16th Geneva JVET meeting.

- a) In the jointed contribution, the geo angles are defined slope (tangle power of 2) same as in JVET-P0107 and JVET-P0264. The slope used in this proposal is (1, 1/2, 1/4, 4, 2). In this case, the multiplications are replaced by shift operation if the blending mask is calculated on the fly.
- b) The rho calculation is replaced by offset X and offset Y as described in JVET-P304. In this case, only the 24 blending masks need to be stored in case of not calculate the blending mask on the fly

[0087] 2.3.2. JVET-P0107 slope based version 2

[0088] Based on the slope based GEO version 2, The Dis[.] look up table is illustrated in Table 1

Table 1 2 bits Dis[.] look up table for slope based GEO

idx	0	1	2	4	6	7	8	9	10	12	14	15
Dis[idx]	4	4	4	4	2	1	0	-1	-2	-4	-4	-4
idx	16	17	18	20	22	23	24	25	26	28	30	31
Dis[idx]	-4	-4	-4	-4	-2	-1	0	1	2	4	4	4

[0089] With the slope based GEO version 2, the computation complexity of geo blending mask derivation is considered as multiplication (up to 2 bits shift) and addition. There is no different partitions compared to TPM. Furthermore, the rounding operation of distFromLine is removed in order to easier store the blending mask. This bugfix guarantees that sample weights are repeated in each row or column in a shifted fashion.

[0090] For example:

TPM:

...
The variable wIdx is derived as follows:
– If cIdx is equal to 0 and triangleDir is equal to 0, the following applies: $wIdx = (nCbw > nCbH) ? (Clip3(0, 8, \underline{(x / nCbR - y) + 4})) \quad (8-842)$ $: (Clip3(0, 8, \underline{(x - y / nCbR) + 4}))s$ – Otherwise, if cIdx is equal to 0 and triangleDir is equal to 1, the following applies: $wIdx = (nCbw > nCbH) ? (Clip3(0, 8, \underline{(nCbH - 1 - x / nCbR - y) + 4})) \quad (8-843)$ $(Clip3(0, 8, \underline{(nCbw - 1 - x - y / nCbR) + 4}))$ – Otherwise, if cIdx is greater than 0 and triangleDir is equal to 0, the following applies: $wIdx = (nCbw > nCbH) ? (Clip3(\underline{0, 4}, \underline{(x / nCbR - y) + 2})) \quad (8-844)$ $: (Clip3(\underline{0, 4}, \underline{(x - y / nCbR) + 2}))$ – Otherwise (if cIdx is greater than 0 and triangleDir is equal to 1), the following applies: $wIdx = (nCbw > nCbH) ? (Clip3(0, 4, \underline{(nCbH - 1 - x / nCbR - y) + 2})) \quad (8-845)$ $(Clip3(0, 4, \underline{(nCbw - 1 - x - y / nCbR) + 2}))$
...

[0091] GEO

Angle idx is 0: $\frac{((x < 1) + 1) * (4 < 1) - ((y < 1) + 1) * (1 < 1) - \rho}{2}$

Angle idx is 4: $\frac{((x < 1) + 1) * (4 < 1) - ((y < 1) + 1) * (4 < 1) - \rho}{2}$

Angle idx is 6: $\frac{((x < 1) + 1) * (2 < 1) - ((y < 1) + 1) * (4 < 1) - \rho}{2}$

[0092] 2.3.3. JVET-P0264 Test 1

[0093] In JVET-P0264, the angles in GEO is replaced with the angles which have powers of 2 as tangent. Since the tangent of the proposed angles is a power-of-2 number, most of multiplications can be replaced by bit-shifting. Besides, the weight values for these angles can be implemented by repeating row-by-row or column-by-column with phase shift. With the proposed angles, one row or column is needed to store per block size and per partition mode. Fig. 11 shows proposed angles for GEO along with their corresponding width:height ratio.

[0094] 2.3.4. JVET-P0304

[0095] In JVET-P0304, it is proposed to derive the weights and the mask for motion field storage for all blocks and partition modes from two sets of pre-defined masks, one for the blending weights derivation and the other for the masks of motion field storage. There are totally 16 masks in each set. Each mask per angle is calculated using the same equations in GEO with block width and block height set to 256 and displacement set to 0. For a block of size W×H with angle φ and distance ρ , the blending weights for the luma samples are directly cropped from the pre-defined masks with offsets calculated as follows:

- Variables offsetX and offsetY are calculated as follows:

$$\begin{aligned} & offsetX \\ &= \begin{cases} (M - W) \gg 1 & \varphi \% 16 = 8 \text{ or } (\varphi \% 16 \neq 0 \text{ and } H \geq W) \\ ((M - W) \gg 1) + \varphi < 16 ? (\rho \times W) \gg 3 : -((\rho \times W) \gg 3) & otherwise \end{cases} \\ & offsetY \\ &= \begin{cases} ((M - H) \gg 1) + \varphi < 16 ? (\rho \times H) \gg 3 : -((\rho \times H) \gg 3) & \varphi \% 16 = 8 \text{ or } (\varphi \% 16 \neq 0 \text{ and } H \geq W) \\ (M - H) \gg 1 & otherwise \end{cases} \end{aligned}$$

- $sampleWeight_L[x][y] = g_sampleWeight_L[\varphi \% 16][x + offsetX][y + offsetY]$.

, where $g_sampleWeight_L[]$ is the pre-defined masks for the blending weights.

[0096] 2.4. Specification of GEO in JVET-P0884

[0097] The following specification is extracted from the provided working draft in JVET-P0884 which is on top of JVET-O2001-vE. In the below spec, GEO is also called merge wedge mode.

Merge data syntax

merge_data(x0, y0, cbWidth, cbHeight, chType) {	Descriptor
if (CuPredMode[chType][x0][y0] == MODE_IBC) {	
if(MaxNumIbcMergeCand > 1)	
merge_idx[x0][y0]	ae(v)
} else {	
if(MaxNumSubblockMergeCand > 0 && cbWidth >= 8 && cbHeight >= 8)	
merge_subblock_flag[x0][y0]	ae(v)
if(merge_subblock_flag[x0][y0] == 1) {	
if(MaxNumSubblockMergeCand > 1)	
merge_subblock_idx[x0][y0]	ae(v)
} else {	
if((cbWidth * cbHeight) >= 64 && (sps_ciip_enabled_flag && cu_skip_flag[x0][y0] == 0 && cbWidth < 128 && cbHeight < 128) (sps_wedge_enabled_flag && MaxNumWedgeMergeCand > 1 && cbWidth >= 8 && cbHeight >= 8 && slice_type == B)))	
regular_merge_flag[x0][y0]	ae(v)
if (regular_merge_flag[x0][y0] == 1) {	
if(sps_mmvd_enabled_flag)	
mmvd_merge_flag[x0][y0]	ae(v)
if(mmvd_merge_flag[x0][y0] == 1) {	
if(MaxNumMergeCand > 1)	
mmvd_cand_flag[x0][y0]	ae(v)
mmvd_distance_idx[x0][y0]	ae(v)
mmvd_direction_idx[x0][y0]	ae(v)
} else {	
if(MaxNumMergeCand > 1)	
merge_idx[x0][y0]	ae(v)
}	
} else {	
if(sps_ciip_enabled_flag && sps_wedge_enabled_flag && MaxNumWedgeMergeCand > 1 && slice_type == B && cu_skip_flag[x0][y0] == 0 && cbWidth >= 8 && cbHeight >= 8 && cbWidth < 128 && cbHeight < 128) {	

ciip_flag[x0][y0]	ae(v)
if(ciip_flag[x0][y0] && MaxNumMergeCand > 1)	
merge_idx[x0][y0]	ae(v)
<u>if(!ciip_flag[x0][y0] && MaxNumWedgeMergeCand > 1) {</u>	
<u>wedge_partition_idx[x0][y0]</u>	ae(v)
<u>merge_wedge_idx0[x0][y0]</u>	ae(v)
<u>merge_wedge_idx1[x0][y0]</u>	ae(v)
 }	
 }	
 }	
 }	
 }	

[0098] The variable **wedge_merge_mode[x0][y0]**, which specifies whether non-rectangular based motion compensation is used to generate the prediction samples of the current coding unit, when decoding a B slice, is derived as follows:

- If all the following conditions are true, **wedge_merge_mode[x0][y0]** is set equal to 1:
 - **sps_wedge_enabled_flag** is equal to 1.
 - **slice_type** is equal to B.
 - **general_merge_flag[x0][y0]** is equal to 1.
 - **MaxNumWedgeMergeCand** is greater than or equal to 2.
 - **cbWidth** is greater than 8 and **cbHeight** is greater than 8.
 - **regular_merge_flag[x0][y0]** is equal to 0.
 - **merge_subblock_flag[x0][y0]** is equal to 0.
 - **ciip_flag[x0][y0]** is equal to 0.
- Otherwise, **wedge_merge_mode[x0][y0]** is set equal to 0.

wedge_partition_idx[x0][y0] specifies the geometric splitting type of the merge geometric mode. The array indices **x0, y0** specify the location (**x0, y0**) of the top-left luma sample of the considered coding block relative to the top-left luma sample of the picture.

merge_merge_wedge_idx0[x0][y0] specifies the first merging candidate index of the non-rectangular shape based motion compensation candidate list where **x0, y0** specify the location

(x_0, y_0) of the top-left luma sample of the considered coding block relative to the top-left luma sample of the picture.

When $\text{wedge_partition_idx0}[x_0][y_0]$ is not present, it is inferred to be equal to 0.

merge_wedge_idx1[x_0][y_0] specifies the second merging candidate index of the wedge shape based motion compensation candidate list where x_0, y_0 specify the location (x_0, y_0) of the top-left luma sample of the considered coding block relative to the top-left luma sample of the picture.

When $\text{merge_wedge_idx1}[x_0][y_0]$ is not present, it is inferred to be equal to 0.

Decoding process for wedge inter blocks

General

This process is invoked when decoding a coding unit with $\text{wedge_merge_mode}[x_{Cb}][y_{Cb}]$ equal to 1.

Inputs to this process are:

- a luma location (x_{Cb}, y_{Cb}) specifying the top-left sample of the current coding block relative to the top-left luma sample of the current picture,
- a variable $cbWidth$ specifying the width of the current coding block in luma samples,
- a variable $cbHeight$ specifying the height of the current coding block in luma samples,
- the luma motion vectors in 1/16 fractional-sample accuracy mvA and mvB ,
- the chroma motion vectors $mvCA$ and $mvCB$,
- the reference indices $refIdxA$ and $refIdxB$,
- the prediction list flags $predListFlagA$ and $predListFlagB$.

Outputs of this process are:

- an $(cbWidth) \times (cbHeight)$ array predSamples_L of luma prediction samples,
- an $(cbWidth / \text{SubWidthC}) \times (cbHeight / \text{SubHeightC})$ array predSamples_{Cb} of chroma prediction samples for the component Cb ,
- an $(cbWidth / \text{SubWidthC}) \times (cbHeight / \text{SubHeightC})$ array predSamples_{Cr} of chroma prediction samples for the component Cr .

Let predSamplesLA_L and predSamplesLB_L be $(cbWidth) \times (cbHeight)$ arrays of predicted luma sample values and, $\text{predSamplesLA}_{Cb}$, $\text{predSamplesLB}_{Cb}$, $\text{predSamplesLA}_{Cr}$ and $\text{predSamplesLB}_{Cr}$ be $(cbWidth / \text{SubWidthC}) \times (cbHeight / \text{SubHeightC})$ arrays of predicted chroma sample values.

The predSamples_L , predSamples_{Cb} and predSamples_{Cr} are derived by the following ordered steps:

1. For N being each of A and B, the following applies:
 - The reference picture including an ordered two-dimensional array refPicLN_L of luma samples and two ordered two-dimensional arrays refPicLN_{Cb} and refPicLN_{Cr} of chroma samples is derived by invoking the process specified in clause 8.5.6.2 with X set equal to predListFlag_N and refIdxX set equal to refIdx_N as input.
 - The array predSamplesLN_L is derived by invoking the fractional sample interpolation process specified in clause 8.5.6.3 with the luma location (x_{Cb} , y_{Cb}), the luma coding block width sbWidth set equal to cbWidth , the luma coding block height sbHeight set equal to cbHeight , the motion vector offset mvOffset set equal to (0, 0), the motion vector mvLX set equal to mv_N and the reference array refPicLX_L set equal to refPicLN_L , the variable bdofFlag set equal to FALSE, and the variable cIdx is set equal to 0 as inputs.
 - The array $\text{predSamplesLN}_{Cb}$ is derived by invoking the fractional sample interpolation process specified in clause 8.5.6.3 with the luma location (x_{Cb} , y_{Cb}), the coding block width sbWidth set equal to $\text{cbWidth} / \text{SubWidth}_C$, the coding block height sbHeight set equal to $\text{cbHeight} / \text{SubHeight}_C$, the motion vector offset mvOffset set equal to (0, 0), the motion vector mvLX set equal to mv_{CN} , and the reference array refPicLX_{Cb} set equal to refPicLN_{Cb} , the variable bdofFlag set equal to FALSE, and the variable cIdx is set equal to 1 as inputs.
 - The array $\text{predSamplesLN}_{Cr}$ is derived by invoking the fractional sample interpolation process specified in clause 8.5.6.3 with the luma location (x_{Cb} , y_{Cb}), the coding block width sbWidth set equal to $\text{cbWidth} / \text{SubWidth}_C$, the coding block height sbHeight set equal to $\text{cbHeight} / \text{SubHeight}_C$, the motion vector offset mvOffset set equal to (0, 0), the motion vector mvLX set equal to mv_{CN} , and the reference array refPicLX_{Cr} set equal to refPicLN_{Cr} , the variable bdofFlag set equal to FALSE, and the variable cIdx is set equal to 2 as inputs.
2. The partition angle and distance of the wedge merge mode angleIdx and distanceIdx are set according to the value of $\text{wedge_partition_idx}[x_{Cb}][y_{Cb}]$ as specified in Table 8-10
3. The prediction samples inside the current luma coding block, $\text{predSamples}_L[x_L][y_L]$ with $x_L = 0..\text{cbWidth} - 1$ and $y_L = 0..\text{cbHeight} - 1$, are derived by invoking the weighted sample prediction process for wedge merge mode specified in clause 8.5.7.2 with the coding block

width $nCbW$ set equal to $cbWidth$, the coding block height $nCbH$ set equal to $cbHeight$, the sample arrays $predSamplesLA_L$ and $predSamplesLB_L$, and the variables $angleIdx$ and $distanceIdx$, and $cIdx$ equal to 0 as inputs.

4. The prediction samples inside the current chroma component Cb coding block, $predSamples_{Cb}[x_C][y_C]$ with $x_C = 0..cbWidth / SubWidthC - 1$ and $y_C = 0..cbHeight / SubHeightC - 1$, are derived by invoking the weighted sample prediction process for wedge merge mode specified in clause 8.5.7.2 with the coding block width $nCbW$ set equal to $cbWidth / SubWidthC$, the coding block height $nCbH$ set equal to $cbHeight / SubHeightC$, the sample arrays $predSamplesLA_{Cb}$ and $predSamplesLB_{Cb}$, and the variables $angleIdx$ and $distanceIdx$, and $cIdx$ equal to 1 as inputs.
5. The prediction samples inside the current chroma component Cr coding block, $predSamples_{Cr}[x_C][y_C]$ with $x_C = 0..cbWidth / SubWidthC - 1$ and $y_C = 0..cbHeight / SubHeightC - 1$, are derived by invoking the weighted sample prediction process for wedge merge mode specified in clause 8.5.7.2 with the coding block width $nCbW$ set equal to $cbWidth / SubWidthC$, the coding block height $nCbH$ set equal to $cbHeight / SubHeightC$, the sample arrays $predSamplesLA_{Cr}$ and $predSamplesLB_{Cr}$, and the variables $angleIdx$ and $distanceIdx$, and $cIdx$ equal to 2 as inputs.
6. The motion vector storing process for merge wedge mode specified in clause 8.5.7.3 is invoked with the luma coding block location (xCb, yCb), the luma coding block width $cbWidth$, the luma coding block height $cbHeight$, the partition direction $angleIdx$ and $distanceIdx$, the luma motion vectors mvA and mvB , the reference indices $refIdxA$ and $refIdxB$, and the prediction list flags $predListFlagA$ and $predListFlagB$ as inputs. The specification of the $angleIdx$ and $distanceIdx$ values is shown in Table 8-10 in FIG. 19.

Weighted sample prediction process for wedge merge mode

Inputs to this process are:

- two variables $nCbW$ and $nCbH$ specifying the width and the height of the current coding block,
- two $(nCbW) \times (nCbH)$ arrays $predSamplesLA$ and $predSamplesLB$,
- a variable $angleIdx$ specifying the angle index of the wedge partition ,
- a variable $distanceIdx$ specifying the distance idx of the wedge partition,
- a variable $cIdx$ specifying colour component index.

Output of this process is the $(nCbW) \times (nCbH)$ array pbSamples of prediction sample values.

The variable bitDepth is derived as follows:

- If cIdx is equal to 0, bitDepth is set equal to BitDepth_Y.
- If cIdx is equal to 0, nW and nH are set equal to nCbW and nCbH respectively, otherwise (cIdx is not equal to 0) nW and nH are set equal to nCbW $\times$ SubWidthC and nCbH $\times$ SubHeightC respectively.
- If cIdx is equal to 0, subW and subH are both set 1, otherwise (cIdx is not equal to 0) subW and subH are set equal to SubWidthC and SubHeightC respectively.
- Otherwise, bitDepth is set equal to BitDepth_C.

Variables shift1 and offset1 are derived as follows:

- The variable shift1 is set equal to $\text{Max}(5, 17 - \text{bitDepth})$.
- The variable offset1 is set equal to $1 \ll (\text{shift1} - 1)$.

The values of the following variables are set:

- hwRatio is set to nH / nW
- displacementX is set to angleIdx
- displacementY is set to $(\text{displacementX} + 6) \% 24$
- If angleIdx ≥ 10 && angleIdx ≤ 20 , PART1 and PART2 are set equal to A and B respectively, otherwise PART1 and PART2 are set equal to B and A respectively.
- rho is set to the following value using the look-up tables denoted as Dis, specified in Table 8-12:

$$\rho = (\text{Dis}[\text{displacementX}] \ll 8) + (\text{Dis}[\text{displacementY}] \ll 8)$$

If one of the following conditions is true, variable shiftHor is set equal to 0:

- angleIdx % 12 is equal to 6
- angleIdx % 12 is not equal to 0 and hwRatio ≥ 1

Otherwise, shiftHor is set equal to 1.

If shiftHor is equal to 0, offsetX and offsetY are derived as follows:

$$\text{offsetX} = (256 - nW) \gg 1$$

$$\text{offsetY} = (256 - nH) \gg 1 + \text{angleIdx} < 12 ? (\text{distanceIdx} * nH) \gg 3 : -((\text{distanceIdx} * nH) \gg 3)$$

Otherwise, if shiftHor is equal to 1, offsetX and offsetY are derived as follows:

$$\text{offsetX} = (256 - \text{nW}) \gg 1 + \text{angleIdx} < 12 ? (\text{distanceIdx} * \text{nW}) \gg 3 : -((\text{distanceIdx} * \text{nW}) \gg 3)$$

$$\text{offsetY} = (256 - \text{nH}) \gg 1$$

The prediction sample values $\text{pbSamples}[x][y]$ with $x = 0..n_{CbW} - 1$ and $y = 0..n_{CbH} - 1$ is set according the following ordered steps:

- The variable weightIdx and weightIdxAbs are calculated using the look-up table Table 8-12 as follows:

$$\begin{aligned} \text{weightIdx} &= (((x * \text{subW} + \text{offsetX}) \ll 1) + 1) * \text{Dis}[\text{displacementX}] \\ &\quad + (((y * \text{subH} + \text{offsetY}) \ll 1) + 1) * \text{Dis}[\text{displacementY}] - \text{rho}. \\ \text{weightIdxAbs} &= \text{Clip3}(0, 26, \text{abs}(\text{weightIdx})). \end{aligned}$$

- The value of sampleWeight is derived according to according to Table 8-13 as follows:

$$\text{sampleWeight} = \text{weightIdx} \leq 0 ? \text{WedgeFilter}[\text{weightIdxAbs}] : 8 - \text{WedgeFilter}[\text{weightIdxAbs}]$$

NOTE – The value of sample $\text{sampleWeight}_L[x][y]$ can also be derived from $\text{sampleWeight}_L[x - \text{shiftX}][y - \text{shiftY}]$. If the angleIdx is larger than 4 and smaller than 12, or angleIdx is larger than 20 and smaller than 24, shiftX is the tangent of the split angle and shiftY is 1, otherwise shiftX is 1 of the split angle and shiftY is cotangent of the split angle. If tangent (resp. cotangent) value is infinity, shiftX is 1 (resp. 0) or shiftY is 0 (reps. 1).

- The prediction sample value $\text{pbSamples}[x][y]$ is derived as follows :

$$\begin{aligned} \text{pbSamples}[x][y] &= \text{Clip3}(0, (1 \ll \text{bitDepth}) - 1, (\text{predSamplesLPART1} \\ &\quad [x][y] * (8 - \text{sampleWeight}) + \\ &\quad \text{predSamplesLPART2}[x][y] * \text{sampleWeight} + \text{offset1}) \gg \text{shift1}) \end{aligned}$$

Table 8-12 - Look-up table Dis for derivation of wedgemetric partitioning distance.

idx	0	1	2	3	4	5	6	7	8	9	10	11
Dis[idx]	8	8	8	8	4	2	0	-2	-4	-8	-8	-8
idx	12	13	14	15	16	17	18	19	20	21	22	23
Dis[idx]	-8	-8	-8	-8	-4	-2	0	2	4	8	8	8

Table 8-13 - Filter weight look-up table WedgeFilter for derivation of wedge partitioning filter weights.

idx	0	1	2	3	4	5	6	7	8	9	10	11	12	13
WedgeFilter[idx]	4	4	4	4	5	5	5	5	5	5	5	6	6	6
idx	14	15	16	17	18	19	20	21	22	23	24	25	26	
WedgeFilter[idx]	6	6	6	6	7	7	7	7	7	7	7	7	8	

Motion vector storing process for wedge merge mode

This process is invoked when decoding a coding unit with MergeWedgeFlag[xCb][yCb] equal to 1.

Inputs to this process are:

- a luma location (xCb, yCb) specifying the top-left sample of the current coding block relative to the top-left luma sample of the current picture,
- a variable cbWidth specifying the width of the current coding block in luma samples,
- a variable cbHeight specifying the height of the current coding block in luma samples,
- the luma motion vectors in 1/16 fractional-sample accuracy mvA and mvB,
- the reference indices refIdxA and refIdxB,
- the prediction list flags predListFlagA and predListFlagB.

The variables numSbX and numSbY specifying the number of 4x4 blocks in the current coding block in horizontal and vertical direction are set equal to numSbX = cbWidth >> 2 and numSbY = cbHeight >> 2.

The value of the following variables are set:

- displacementX is set to angleIdx, displacementY is set to (displacementX + 6)%24
- hwRatio is set equal to nCbH / nCbW

If one of the following conditions is true, variable shiftHor is set equal to 0:

angleIdx % 12 is equal to 8

angleIdx % 12 is not equal to 0 and hwRatio ≥ 1

Otherwise, shiftHor is set equal to 1.

partIdx is set to $\text{angleIdx} \geq 10 \ \&\& \ \text{angleIdx} \leq 20 ? 1 : 0$.

If shiftHor is equal to 0, offsetX and offsetY are derived as follows:

- $\text{offsetX} = (64 - \text{numSbX}) \gg 1$
- $\text{offsetY} = (64 - \text{numSbY}) \gg 1 + \text{angleIdx} < 12 ? (\text{distanceIdx} * \text{nCbH}) \gg 5 : -((\text{distanceIdx} * \text{nCbH}) \gg 5)$

Otherwise, if shiftHor is equal to 1, offsetX and offsetY are derived as follows:

- $\text{offsetX} = (64 - \text{numSbX}) \gg 1 + \text{angleIdx} < 12 ? (\text{distanceIdx} * \text{nCbW}) \gg 5 : -((\text{distanceIdx} * \text{nCbW}) \gg 5)$
- $\text{offsetY} = (64 - \text{numSbY}) \gg 1$

The value of the variable rho is derived according to the equation below and according to the Dis lookup table specified in Table 8-12:

- $\text{rho} = (\text{Dis}[\text{displacementX}] < 8) + (\text{Dis}[\text{displacementY}] < 8)$.

motionOffset is set equal to the following value using the look-up tables denoted as Dis, specified in Table 8-11 and Table 8-12:

- $\text{motionOffset} = 3 * \text{Dis}[\text{displacementX}] + 3 * \text{Dis}[\text{displacementY}]$.

For each 4x4 subblock at subblock index (xSbIdx, ySbIdx) with $\text{xSbIdx} = 0.. \text{numSbX} - 1$, and $\text{ySbIdx} = 0.. \text{numSbY} - 1$, the following applies:

The variable motionIdx is calculated using the look-up table Table 8-12 as following:

- $\text{motionIdx} = (((\text{xSbIdx} + \text{offsetX}) < 3) + 1) * \text{Dis}[\text{displacementX}] + (((\text{ySbIdx} + \text{offsetY}) < 3) + 1) * \text{Dis}[\text{displacementY}] - \text{rho} + \text{motionOffset}$

The variable sType is derived as follows:

- $\text{sType} = \text{abs}(\text{motionIdx}) < 32 ? 2 : \text{motionIdx} \leq 0 ? \text{partIdx} : 1 - \text{partIdx}$
- Depending on the value of sType, the following assignments are made:
 - If sType is equal to 0, the following applies:

$$\text{predFlagL0} = (\text{predListFlagA} == 0) ? 1 : 0 \quad (8-853)$$

$$\text{predFlagL1} = (\text{predListFlagA} == 0) ? 0 : 1 \quad (8-854)$$

$$\text{refIdxL0} = (\text{predListFlagA} == 0) ? \text{refIdxA} : -1 \quad (8-855)$$

$$\text{refIdxL1} = (\text{predListFlagA} == 0) ? -1 : \text{refIdxA} \quad (8-856)$$

$$mvL0[0] = (predListFlagA == 0) ? mvA[0] : 0 \quad (8-857)$$

$$mvL0[1] = (predListFlagA == 0) ? mvA[1] : 0 \quad (8-858)$$

$$mvL1[0] = (predListFlagA == 0) ? 0 : mvA[0] \quad (8-859)$$

$$mvL1[1] = (predListFlagA == 0) ? 0 : mvA[1] \quad (8-860)$$

- Otherwise, if sType is equal to 1 or (sType is equal to 2 and predListFlagA + predListFlagB is not equal to 1), the following applies:

$$predFlagL0 = (predListFlagB == 0) ? 1 : 0 \quad (8-861)$$

$$predFlagL1 = (predListFlagB == 0) ? 0 : 1 \quad (8-862)$$

$$refIdxL0 = (predListFlagB == 0) ? refIdxB : -1 \quad (8-863)$$

$$refIdxL1 = (predListFlagB == 0) ? -1 : refIdxB \quad (8-864)$$

$$mvL0[0] = (predListFlagB == 0) ? mvB[0] : 0 \quad (8-865)$$

$$mvL0[1] = (predListFlagB == 0) ? mvB[1] : 0 \quad (8-866)$$

$$mvL1[0] = (predListFlagB == 0) ? 0 : mvB[0] \quad (8-867)$$

$$mvL1[1] = (predListFlagB == 0) ? 0 : mvB[1] \quad (8-868)$$

- Otherwise (sType is equal to 2 and predListFlagA + predListFlagB is equal to 1), the following applies:

$$predFlagL0 = 1 \quad (8-869)$$

$$predFlagL1 = 1 \quad (8-870)$$

$$refIdxL0 = (predListFlagA == 0) ? refIdxA : refIdxB \quad (8-871)$$

$$refIdxL1 = (predListFlagA == 0) ? refIdxB : refIdxA \quad (8-872)$$

$$mvL0[0] = (predListFlagA == 0) ? mvA[0] : mvB[0] \quad (8-873)$$

$$mvL0[1] = (predListFlagA == 0) ? mvA[1] : mvB[1] \quad (8-874)$$

$$mvL1[0] = (predListFlagA == 0) ? mvB[0] : mvA[0] \quad (8-875)$$

$$mvL1[1] = (predListFlagA == 0) ? mvB[1] : mvA[1] \quad (8-876)$$

- The following assignments are made for x = 0..3 and y = 0..3:

$$MvL0[(xSbIdx << 2) + x][(ySbIdx << 2) + y] = mvL0 \quad (8-877)$$

$$MvL1[(xSbIdx << 2) + x][(ySbIdx << 2) + y] = mvL1 \quad (8-878)$$

$$RefIdxL0[(xSbIdx << 2) + x][(ySbIdx << 2) + y] = refIdxL0 \quad (8-879)$$

$$\text{RedIdxL1}[(\text{xSbIdx} \ll 2) + x][(\text{ySbIdx} \ll 2) + y] = \text{refIdxL1} \quad (8-880)$$

$$\text{PredFlagL0}[(\text{xSbIdx} \ll 2) + x][(\text{ySbIdx} \ll 2) + y] = \text{predFlagL0} \quad (8-881)$$

$$\text{PredFlagL1}[(\text{xSbIdx} \ll 2) + x][(\text{ySbIdx} \ll 2) + y] = \text{predFlagL1} \quad (8-882)$$

Decoding process for the residual signal of coding blocks coded in inter prediction mode

Inputs to this process are:

- a sample location (xTb0, yTb0) specifying the top-left sample of the current transform block relative to the top-left sample of the current picture,
- a variable nTbW specifying the width of the current transform block,
- a variable nTbH specifying the height of the current transform block,
- a variable cIdx specifying the colour component of the current block.

Output of this process is an (nTbW)x(nTbH) array resSamples.

The maximum transform block width maxTbWidth and height maxTbHeight are derived as follows:

$$\text{maxTbWidth} = (\text{cIdx} == 0) ? \text{MaxTbSizeY} : \text{MaxTbSizeY} / \text{SubWidthC} \quad (8-883)$$

$$\text{maxTbHeight} = (\text{cIdx} == 0) ? \text{MaxTbSizeY} : \text{MaxTbSizeY} / \text{SubHeightC} \quad (8-884)$$

The luma sample location is derived as follows:

$$(\text{xTbY}, \text{yTbY}) = (\text{cIdx} == 0) ? (\text{xTb0}, \text{yTb0}) : (\text{xTb0} * \text{SubWidthC}, \text{yTb0} * \text{SubHeightC}) \quad (8-885)$$

Depending on maxTbSize, the following applies:

- If nTbW is greater than maxTbWidth or nTbH is greater than maxTbHeight, the following ordered steps apply.
 1. The variables newTbW and newTbH are derived as follows:

$$\text{newTbW} = (\text{nTbW} > \text{maxTbWidth}) ? (\text{nTbW} / 2) : \text{nTbW} \quad (8-886)$$

$$\text{newTbH} = (\text{nTbH} > \text{maxTbHeight}) ? (\text{nTbH} / 2) : \text{nTbH} \quad (8-887)$$
 2. The decoding process for the residual signal of coding units coded in inter prediction mode as specified in this clause is invoked with the location (xTb0, yTb0), the transform block width nTbW set equal to newTbW and the height nTbH set equal to newTbH, and the variable cIdx as inputs, and the output is a modified reconstructed picture before in-loop filtering.
 3. When nTbW is greater than maxTbWidth, the decoding process for the residual signal of coding units coded in inter prediction mode as specified in this clause is invoked

with the location (xTb0, yTb0) set equal to (xTb0 + newTbW, yTb0), the transform block width nTbW set equal to newTbW and the height nTbH set equal to newTbH, and the variable cIdx as inputs, and the output is a modified reconstructed picture .

4. When nTbH is greater than maxTbHeight, the decoding process for the residual signal of coding units coded in inter prediction mode as specified in this clause is invoked with the location (xTb0, yTb0) set equal to (xTb0, yTb0 + newTbH), the transform block width nTbW set equal to newTbW and the height nTbH set equal to newTbH, and the variable cIdx as inputs, and the output is a modified reconstructed picture before in-loop filtering.
 5. When nTbW is greater than maxTbWidth and nTbH is greater than maxTbHeight, the decoding process for the residual signal of coding units coded in inter prediction mode as specified in this clause is invoked with the location (xTb0, yTb0) set equal to (xTb0 + newTbW, yTb0 + newTbH), the transform block width nTbW set equal to newTbW and height nTbH set equal to newTbH, and the variable cIdx as inputs, and the output is a modified reconstructed picture before in-loop filtering.
- Otherwise, if cu_sbt_flag is equal to 1, the following applies:
 - The variables sbtMinNumFourths, wPartIdx and hPartIdx are derived as follows:

$$\text{sbtMinNumFourths} = \text{cu_sbt_quad_flag} ? 1 : 2 \quad (8-888)$$

$$\text{wPartIdx} = \text{cu_sbt_horizontal_flag} ? 4 : \text{sbtMinNumFourths} \quad (8-889)$$

$$\text{hPartIdx} = !\text{cu_sbt_horizontal_flag} ? 4 : \text{sbtMinNumFourths} \quad (8-890)$$
 - The variables xPartIdx and yPartIdx are derived as follows:
 - If cu_sbt_pos_flag is equal to 0, xPartIdx and yPartIdx are set equal to 0.
 - Otherwise (cu_sbt_pos_flag is equal to 1), the variables xPartIdx and yPartIdx are derived as follows:

$$\text{xPartIdx} = \text{cu_sbt_horizontal_flag} ? 0 : (4 - \text{sbtMinNumFourths}) \quad (8-891)$$

$$\text{yPartIdx} = !\text{cu_sbt_horizontal_flag} ? 0 : (4 - \text{sbtMinNumFourths}) \quad (8-892)$$
 - The variables xTbYSub, yTbYSub, xTb0Sub, yTb0Sub, nTbWSub and nTbHSub are derived as follows:

$$\text{xTbYSub} = \text{xTbY} + ((\text{nTbW} * ((\text{cIdx} == 0) ? 1 : \text{SubWidthC}) * \text{xPartIdx} / 4) \quad (8-893)$$

$$yTbYSub = yTbY + ((nTbH * ((cIdx == 0) ? 1 : SubHeightC) * yPartIdx / 4)$$

(8-894)

$$xTb0Sub = xTb0 + (nTbW * xPartIdx / 4)$$

(8-895)

$$yTb0Sub = yTb0 + (nTbH * yPartIdx / 4)$$

(8-896)

$$nTbWSub = nTbW * wPartIdx / 4$$

(8-897)

$$nTbHSub = nTbH * hPartIdx / 4$$

(8-898)

- The scaling and transformation process as specified in clause 8.7.2 is invoked with the luma location (xTbYSub , yTbYSub), the variable cIdx, nTbWSub and nTbHSub as inputs, and the output is an (nTbWSub)x(nTbHSub) array resSamplesTb.
- The residual samples resSamples[x][y] with x = 0..nTbW – 1, y = 0..nTbH – 1 are set equal to 0.
- The residual samples resSamples[x][y] with x = xTb0Sub..xTb0Sub + nTbWSub – 1, y = yTb0Sub..yTb0Sub + nTbHSub – 1 are derived as follows:

$$resSamples[x][y] = resSamplesTb[x - xTb0Sub][y - yTb0Sub]$$

(8-899)
- Otherwise, the scaling and transformation process as specified in clause 8.7.2 is invoked with the luma location (xTbY, yTbY), the variable cIdx, the transform width nTbW and the transform height nTbH as inputs, and the output is an (nTbW)x(nTbH) array resSamples.

Table 9-77 – Syntax elements and associated binarizations

Syntax structure	Syntax element	Binarization	
		Processes	Input parameters
merge_data()	regular_merge_flag[][]	FL	cMax = 1
	mmvd_merge_flag[][]	FL	cMax = 1
	mmvd_cand_flag[][]	FL	cMax = 1
	mmvd_distance_idx[][]	TR	cMax = 7, cRiceParam = 0
	mmvd_direction_idx[][]	FL	cMax = 3
	ciip_flag[][]	FL	cMax = 1
	merge_subblock_flag[][]	FL	cMax = 1
	merge_subblock_idx[][]	TR	cMax = MaxNumSubblockMergeCand - 1, cRiceParam = 0
	<u>wedge_partition_idx[][]</u>	<u>TB</u>	<u>cMax = 82</u>
	<u>merge_wedge_idx0[][]</u>	<u>TR</u>	<u>cMax = MaxNumWedgeMergeCand - 1, cRiceParam = 0</u>
	<u>merge_wedge_idx1[][]</u>	<u>TR</u>	<u>cMax = MaxNumWedgeMergeCand - 2, cRiceParam = 0</u>
	merge_idx[][]	TR	cMax = MaxNumMergeCand - 1, cRiceParam = 0

[0099] 2.5. Chroma sample location type

This paragraph of the definition of chroma sample location type is extracted from JVET-P2007-v3. Fig. 12 illustrates the indicated relative position of the top-left chroma sample when chroma_format_idc is equal to 1 (4:2:0 chroma format), and chroma_sample_loc_type_top_field or chroma_sample_loc_type_bottom_field is equal to the value of a variable ChromaLocType. The region represented by the top-left 4:2:0 chroma sample (depicted as a large red square with a large red dot at its centre) is shown relative to the region represented by the top-left luma sample (depicted as a small black square with a small black dot at its centre). The regions represented by neighbouring luma samples are depicted as small grey squares with small grey dots at their centres.

[0100] Drawbacks of existing implementations

[0101] There are several potential issues in the current design of GEO, which are described below.

- (1) In the CE anchor of JVET-P0884, the total number of GEO modes for hardware validation is 1558, which is calculated from multiplying 19 PU shapes by 82 GEO modes. It is expressed by

experts that the 1558 validation cases for the GEO coding tool is too much. It is desirable to reduce the total cases of GEO.

- (2) In the CE anchor of JVET-P0884, the GEO mode is applied to block sizes no smaller than 8×8 , that is $W \geq 8$ and $H \geq 8$.
- a) GEO mode may be not that necessary for large block sizes. A better tradeoff between the coding gain and complexity may be considered by reducing the allowable block sizes for GEO.
 - b) The $4 \times N$ and $N \times 4$ block sizes may be beneficial for the coding gain.

[0102] Example techniques and embodiments

[0103] The detailed embodiments described below should be considered as examples to explain general concepts. These embodiments should not be interpreted narrowly way. Furthermore, these embodiments can be combined in any manner.

[0104] The term ‘GEO’ may represent a coding method that split one block into two or more sub-regions wherein at least one sub-region couldn’t be generated by any of existing partitioning structure (e.g., QT/BT/TT). The term ‘GEO’ may indicate the triangle prediction mode (TPM), and/or geometric merge mode (GEO), and/or wedge prediction mode.

[0105] The term ‘block’ may represent a coding block of CU and/or PU and/or TU.

[0106] In some embodiments, the ‘GEO mode index (or GEO mode)’ may be the signaled GEO mode index in the coded bitstream. In some embodiments, the GEO mode index (or GEO mode) is used to derive the GEO angle index and GEO distance index in the decoding process of wedge inter blocks. In some embodiments, the GEO mode index (or GEO mode) for deriving the angle/distance index in the decoding process may also be obtained by table mapping. If not specified, the GEO mode index can mean the wedge_partition_idx that is used to derive the angle/distance index in the decoding process, such as which is defined in the Table 8-10 in the working draft of JVET-P0884-v8.

Block size restriction for GEO mode

Denote the block width as W and the block height as H .

1. Whether GEO is allowed or not may be dependent on the block width and/or block height.
 - a) Whether GEO is allowed or not may be dependent on the block size (such as $W \times H$) and/or the aspect ratio of the block.

- i. For example, for a $W \times H$ block, GEO may only be enabled if $W \geq T1$ and/or $H \geq T2$ and/or $W * H < T3$ and/or $W * H > T4$, wherein $T1$, $T2$, $T3$ and $T4$ are constant values.
- ii. For another example, for a $W \times H$ block, GEO may only be enabled if $W \geq T1$ and/or $H \geq T2$ and/or $W * H \leq T3$ and/or $W * H \geq T4$, wherein $T1$, $T2$, $T3$ and $T4$ are constant values.
- iii. In one example, for a $W \times H$ block, GEO may only be enabled if $W * H < T1 \parallel (W * H \leq T2 \&\& W/H \leq T3 \&\& H/W \leq T4)$.
 - 1) In one example, $T1$, $T2$, $T3$ and $T4$ may refer to luma blocks.
 - 2) In one example, $T1 = 512$, $T2 = 2048$, $T3 = 2$, $T4 = 2$.
- iv. In one example, for a $W \times H$ block, GEO may only be enabled if $W * H < T1 \parallel (W * H \leq T2 \&\& \text{abs}(\log W - \log H) \leq T3)$.
 - 1) In one example, $T1$, $T2$, $T3$ and $T4$ may refer to luma blocks.
 - 2) In one example, $T1 = 512$, $T2 = 2048$, $T3 = 1$.
- v. In one example, for a $W \times H$ block, GEO may only be enabled if $W * H \leq T1 \&\& W/H \leq T2 \&\& H/W \leq T3$.
 - 1) In one example, $T1$, $T2$, $T3$ and $T4$ may refer to luma blocks.
 - 2) In one example, $T1 = 2048$, $T2 = 2$, $T3 = 4$.
- vi. In one example, for a $W \times H$ block, GEO may only be enabled if $W \geq T_x$ and $H \geq T_y$ and/or one of the above 1.a.i to 1.a.v are satisfied.
 - 1) In one example, T_x and T_y may refer to luma blocks.
 - 2) In one example, $T_x = 8$, $T_y = 8$.
- vii. GEO may be not allowed for blocks with block width larger than N or/and block height larger than M .
 - 1) In one example, N and M may refer to luma blocks.
 - 2) In one example, $N = M = 64$.
 - 3) In one example, $N = M = 32$.
- viii. GEO may be not allowed for blocks with block width equal to N or/and block height equal to M .
 - 1) In one example, $N = M = 4$.

ix. For example, for a $W \times H$ block, GEO may be not allowed if one and/or more than one conditions in below (1.a) to (1.f) are satisfied, where T_i ($i=1 \dots 17$) are constant values.

- 1) The conditions (1.a) to (1.f) may be as follows.
 - a) $W < T_1$ and/or $W > T_2$ and/or $W = T_3$
 - b) $H < T_4$ and/or $H > T_5$ and/or $H = T_6$
 - c) $W * H < T_7$ and/or $W * H > T_8$ and/or $W * H = T_8$
 - d) $W/H < T_9$ and/or $W/H > T_{10}$ and/or $W/H = T_{11}$
 - e) $H/W < T_{12}$ and/or $H/W > T_{13}$ and/or $H/W = T_{14}$
 - f) $\text{Abs}(\log W - \log H) > T_{15}$ and/or $\text{Abs}(\log W - \log H) < T_{16}$ and/or $\text{Abs}(\log W - \log H) = T_{17}$
- 2) Alternatively, GEO can only be allowed if one or more than one conditions in above (1.a) to (1.f) are satisfied.
- 3) For example, for a $W \times H$ block, GEO may be not allowed if $W < T_1$ or $H < T_2$ or $W * H > T_3$ or ($W * H \geq T_4$ and $\text{Abs}(\log W - \log H) > T_5$).
 - a) Alternatively, for a $W \times H$ block, GEO may only be allowed if $W \geq T_1$ and $H \geq T_2$ and ($W * H < T_4$ or ($W * H \leq T_3$ and $\text{Abs}(\log W - \log H) \leq T_5$)).
 - b) In one example, T_i ($i=1 \dots 5$) may refer to luma blocks.
 - c) In one example, $T_1 = 8$, $T_2 = 8$, $T_3 = 2048$, $T_4 = 512$, $T_5 = 1$
- 4) For example, for a $W \times H$ block, GEO may be not allowed if $W < T_1$ or $H < T_2$ or $W * H > T_3$ or ($W * H \geq T_4$ and ($W/H > T_5$ or $H/W > T_5$)).
 - a) Alternatively, for a $W \times H$ block, GEO may only be allowed if $W \geq T_1$ and $H \geq T_2$ and ($W * H < T_4$ or ($W * H \leq T_3$ and $W/H \leq T_5$ and $H/W \leq T_5$)).
 - b) In one example, T_i ($i=1 \dots 5$) may refer to luma blocks.
 - c) In one example, $T_1 = 8$, $T_2 = 8$, $T_3 = 2048$, $T_4 = 512$, $T_5 = 2$.
- 5) For example, for a $W \times H$ block, GEO may be not allowed if $W < T_1$ or $H < T_2$ or $W * H > T_3$ or $H/W > T_4$ or $W/H > T_5$.
 - a) Alternatively, for a $W \times H$ block, GEO may only be allowed if $W \geq T_1$ and $H \geq T_2$ and $W * H \leq T_3$ and $H/W \leq T_4$ and $W/H \leq T_5$.
 - b) In one example, T_i ($i=1 \dots 5$) may refer to luma blocks.
 - c) In one example, $T_1 = 8$, $T_2 = 8$, $T_3 = 2048$, $T_4 = 4$, $T_5 = 2$.

- b) Whether to enable or disable GEO may depend on a function of block width and height.
- i. For example, the function may depend on the ratios of block width and/or height. For example, the function may be $\max(H, W) / \min(H, W)$.
 - ii. For example, the function may be the differences and/or ratios between block width and height, e.g., $\text{Abs}(\text{Log}_2(\text{cbWidth}) - \text{Log}_2(\text{cbHeight}))$ wherein $\text{Abs}(x)$ returns the absolute value of x and $\text{Log}_2(x)$ returns the log base 2 of a number x .
 - iii. For example, the function may be the sum or difference of block width and block height, e.g., $\text{cbWidth} + \text{cbHeight}$, and/or $\text{cbWidth} - \text{cbHeight}$, and/or $\text{cbHeight} - \text{cbWidth}$.
 - 1) In one example, GEO may be disabled when $\text{cbWidth} + \text{cbHeight}$ is equal to (or less than, or greater than) a number X .
 - 2) In one example, GEO may be disabled when $\text{cbWidth} - \text{cbHeight}$ is equal to (or less than, or greater than) a number X .
 - 3) In one example, GEO may be disabled when $\text{cbHeight} - \text{cbWidth}$ is equal to (or less than, or greater than) a number X .
 - 4) The function may be the absolute value of the difference between the block width and block height.
 - iv. For example, the function may be as follows, where $A, B, \dots, F$ represent constant numbers.
 - 1) $A * \text{cbWidth} < \text{cbHeight}$, and/or $B * \text{cbHeight} < \text{cbWidth}$ and/or $C * \text{cbWidth} > \text{cbHeight}$, and/or $D * \text{cbHeight} > \text{cbWidth}$, and/or $\log_2(\text{cbWidth}) - \log_2(\text{cbHeight}) < E$, and/or $\log_2(\text{cbWidth}) - \log_2(\text{cbHeight}) < F$, and/or $(\text{cbWidth} < S) < \text{cbHeight}$, and/or $(\text{cbHeight} < S) < \text{cbWidth}$ and/or $(\text{cbWidth} < S) > \text{cbHeight}$, and/or $(\text{cbHeight} < S) > \text{cbWidth}$ and/or $\text{abs}(\log_2(\text{cbWidth}) - \log_2(\text{cbHeight})) < F$.
 - a) In one example, $A/B/C/D$ is equal to 8.
 - b) In one example, E is equal to 4.
 - c) S may be equal to 2, or 3, or 4.
 - d) ">" above may be replaced by ">=".
 - e) "<" above may be replaced by "<=".

- c) GEO may be not allowed for blocks with width to height ratio or height to width ratio greater than (in another example, no less than) X (e.g., $X = 2$).
 - i. In one example, for a $W \times H$ block, GEO may be disabled if $W/H > X$ (e.g., $X=2$).
 - ii. In one example, for a $W \times H$ block, GEO may be disabled if $H/W > X$ (e.g., $X=2$).
- d) In one example, GEO may be enabled for one color component (e.g., luma block) but disabled for another color component (e.g., chroma block) in the same coding unit/prediction unit/block.
- e) In one example, for one coding unit/prediction unit/block, whether to allow/disallow GEO may be dependent on the luma block's dimension.
 - i. In one example, when GEO is disallowed for the luma block, it is also disabled for a chroma block.
 - ii. In one example, when GEO is allowed for the luma block, it is also allowed for a chroma block.
- f) Whether GEO is enabled or not may be dependent on the block width and/or block height and/or block width-to-height ratio and/or block height-to-width ratio.
 - i. For example, for a $W \times H$ block, GEO may be only allowed when $W \geq T1$ and $H \geq T2$ and $W \leq T3$ and $H \leq T4$ and $W/H \leq T5$ and $H/W \leq T6$.
 - 1) In one example, $T1=T2=8$, $T3=T4=64$, $T5=2$, $T6=4$.
 - 2) In one example, $T1=T2=8$, $T3=T4=64$, $T5=T6=4$.
 - 3) In one example, $T1=T2=8$, $T3=T4=32$, $T5=2$, $T6=4$.
 - 4) In one example, $T1=T2=8$, $T3=T4=32$, $T5=T6=4$.
 - ii. For example, for a $W \times H$ block, GEO may be only allowed when $W \geq T1$ and $H \geq T2$ and $W \leq T3$ and $H \leq T4$.
 - 1) In one example, $T1=T2=8$, $T3=T4=64$.
 - 2) In one example, $T1=T2=8$, $T3=T4=32$.
 - iii. Alternatively, for a $W \times H$ block, GEO may be disabled when $W < T1$ or $H < T2$ or $W > T3$ or $H > T4$ or $W/H > T5$ or $H/W > T6$.
 - 1) In one example, $T1=T2=8$, $T3=T4=64$, $T5=2$, $T6=4$.
 - 2) In one example, $T1=T2=8$, $T3=T4=64$, $T5=T6=4$.
 - 3) In one example, $T1=T2=8$, $T3=T4=32$, $T5=2$, $T6=4$.

- 4) In one example, $T1=T2=8$, $T3=T4=32$, $T5=T6=4$.
- iv. Alternatively, for a $W \times H$ block, GEO may be disabled when $W < T1$ or $H < T2$ or $W > T3$ or $H > T4$.
 - 1) In one example, $T1=T2=8$, $T3=T4=64$.
 - 2) In one example, $T1=T2=8$, $T3=T4=32$.
- 2. Whether GEO is allowed or not for a block may be dependent on the maximum transform size.
 - a) In one example, GEO may be not allowed for a block with width or/and height larger than the maximum transform size.
- 3. Whether GEO is allowed or not for a block may be dependent on the maximum allowed CU size.
 - a) In one example, GEO may be not allowed for a block with block width or/and height equal to the maximum CU size.
- 4. GEO may be not allowed for a certain chroma format.
 - a) In one example, GEO may be not allowed for 4:0:0 chroma format.
 - b) In one example, GEO may be not allowed for 4:4:4 chroma format.
 - c) In one example, GEO may be not allowed for 4:2:2 chroma format.
 - d) In one example, GEO may be not allowed for a certain color component (such as Cb or Cr) with a certain chroma format.
- 5. GEO and coding tool X may be mutually exclusive.
 - a) In one example, if GEO is applied to the block, the coding tool X is disabled.
 - i. Alternatively, furthermore, signaling of indication of usage of the coding tool X and/or side information of the coding tool X is skipped when GEO is applied.
 - ii. Alternatively, when coding tool X is applied to a block, GEO is not applied.
 - 1) Alternatively, furthermore, signaling of indication of usage of GEO and/or side information of GEO is skipped when X is applied.
 - b) In one example, X may refer to as adaptive color transform.
 - c) In one example, X may refer to as dual tree coding mode.
 - d) In one example, X may refer to as transform skip mode.
 - e) In one example, X may refer to as BDPCM coding mode.
 - f) In one example, X may be Sub-Block Transform (SBT).

- i. In one example, whether to disable SBT may depend on the GEO angle indices and/or GEO distance indices and/or GEO mode indices and/or sub-partitions due to GEO.
 - 1) In one example, if the edge of sub-partitions due to GEO and the edge of sub-partitions due to SBT are crossed, then SBT may be disabled for a GEO coded block.
 - 2) Alternatively, if the edge of sub-partitions due to GEO and the edge of sub-partitions due to SBT are NOT crossed, then SBT may be enabled for a GEO coded block
 - ii. In one example, which kinds of SBT (e.g., horizontal SBT, vertical SBT, and etc.) are used to GEO-coded block may depend on the GEO angle indices and/or GEO distance indices and/or GEO mode indices and/or sub-partitions due to GEO.
 - 1) In one example, if the edge of sub-partitions due to GEO and the edge of sub-partitions due to SBT are crossed, then horizontal SBT and/or vertical SBT may be disabled for a GEO coded block.
 - 2) Alternatively, if the edge of sub-partitions due to GEO and the edge of sub-partitions due to SBT are NOT crossed, then horizontal SBT and/or vertical SBT may be enabled for a GEO coded block.
6. Different color components may have different GEO mode index.
- a) In one example, chroma components may have a different GEO index of luma component.
 - b) In one example, GEO may be not applied to chroma components.
 - c) Alternatively, furthermore, different GEO mode indices may be signaled for different color components.
 - i. For example, a mode index may be signaled for the luma component and a mode index may be signaled for the chroma components.
 - ii. Alternatively, furthermore, mode index of a first color component may be predicted from mode index of a second color component.
7. GEO may be not allowed if the resolutions of reference pictures associated with different sub-regions in GEO are different.
- a) Alternatively, GEO may be not allowed if the resolution of one reference pictures used in GEO is different to the resolution of the current picture.

- b) Alternatively, GEO may be allowed even when the resolution of reference picture and the current picture are different.
 - c) The resolution of a picture may refer to the width/height of the picture, or it may refer to a window in the picture, such as a conformance window or a scaling window in the picture.
8. When GEO is disabled or not allowed, the GEO syntax elements (such as **wedge_partition_idx**, **merge_wedge_idx0**, and **merge_wedge_idx1** in the syntax table of merge data signalling) may be not signaled.
- a) When a syntax element is not signaled, it may be inferred to be a default value such as 0.
 - b) When GEO is disabled or not allowed, the GEO related semantic variables (such as **wedge_merge_mode**) may be inferred to be a default value such as 0.

Block size dependent GEO mode selection

9. One or more syntax elements (e.g., flag) may be signaled in sequence/picture/slice/tile/brick/subpicture/other video processing unit (e.g., a VPDU) level to specify how many GEO modes are allowed for the video unit (e.g., sequence/group of pictures/picture/subpicture/slice/tile/VPDU/CTU row/CTU/CU/PU/TU).
- a) In one example, they may be signaled in SPS/VPS/APS/PPS/PH/SH/picture/subpicture/slice/tile level.
 - i. Alternatively, furthermore, the syntax elements may be conditionally signaled, such as whether the GEO mode is enabled for a video processing unit (such as whether **sps_geo_enabled_flag** is equal to 1); and/or whether current picture type is non-Intra or B picture; and/or whether current slice type is B slice.
 - b) In one example, the syntax element may indicate whether the number of allowed GEO modes in the video processing unit is equal to X (such as X=16 or 32 or 30) or not.
 - i. In one example, one syntax element (e.g., one SPS flag, or one PPS flag, or one flag in picture header) may be signaled for indicating whether X (such as X=16 or 32 or 30) GEO modes are allowed for all blocks in the video unit.
 - 1) Alternatively, the one flag may be signaled for indicating whether X (such as X=16 or 32 or 30) GEO modes are allowed for selective blocks, such as for those with condition C satisfied.
 - a) C may be illustrated as: blocks with $H/W \leq T$ (e.g., $T = 1$ or 2 or 4 or 8).

- b) C may be illustrated as: blocks with $H/W > T$ (e.g., $T = 1$ or 2 or 4 or 8).
- ii. In one example, multiple syntax elements (e.g., two SPS flags) may be signaled to indicate the allowed GEO modes for each category of blocks wherein blocks are classified to multiple categories, such as according to block dimension.
 - 1) In one example, one is for indicating whether X (such as $X=16$ or 32 or 30) GEO modes are allowed for blocks with condition C. The other is for indicating whether Y (such as $Y=16$ or 32 or 30) GEO modes are allowed for blocks with condition D.
 - a) Alternatively, furthermore, C may be blocks with $H/W \leq T$ (e.g., $T = 1$ or 2 or 4 or 8), while D may be blocks with $H/W > T$ (e.g., $T = 1$ or 2 or 4 or 8).
- c) In one example, how to signal the GEO mode index for a block may be dependent on the afore-mentioned syntax elements (e.g., flag).
 - i. In one example, the binarization and/or entropy coding of the GEO mode index for a block may be dependent on the syntax element and/or the block dimensions.
 - 1) In one example, if the number of allowed GEO modes for a block derived by the syntax element is equal to X (such as $X=16$ or 32 or 30), then the value of cMax for GEO mode index coding may be equal to X.
 - 2) In one example, if the number of allowed GEO modes for a block derived by syntax element is equal to X (such as $X=16$ or 32 or 30), and the block dimensions satisfy the condition C, then the value of cMax for GEO mode index coding may be equal to X.
 - a) C may be illustrated as: blocks with $H/W \leq T$ (e.g., $T = 1$ or 2 or 4 or 8).
 - b) C may be illustrated as: blocks with $H/W > T$ (e.g., $T = 1$ or 2 or 4 or 8).
 - ii. In one example, the binarization methods for GEO mode index coding may be different according to the block dimensions and/or the syntax element.
- d) In one example, the value of maximum value of **merge_geo_partition_idx** (e.g., **wedge_partition_idx**) may be dependent on the afore-mentioned syntax elements (e.g., flag) and/or the block dimensions.
 - i. In one example, a bitstream constraint may be added to constrain the value the **merge_geo_partition_idx** (e.g., **wedge_partition_idx**) should be less than the maximum allowed GEO mode in the bitstream.

- ii. In one example, a bitstream constraint may be added to constrain the value the **merge_geo_partition_idx** (e.g., **wedge_partition_idx**) should be less than the maximum allowed GEO mode for blocks with block dimensions satisfy the condition C.
 - a) C may be illustrated as: blocks with $H/W \leq T$ (e.g., $T = 1$ or 2 or 4 or 8).
 - b) C may be illustrated as: blocks with $H/W > T$ (e.g., $T = 1$ or 2 or 4 or 8).
- e) In one example, one or more constraint flags may be signaled in a video processing unit level to specify whether to constrain the usage of the X (such as $X=16$ or 32 or 30) modes GEO method for the video unit.
 - 1) In one example, a constraint flag may be signaled to constrain whether X modes GEO method is used for all blocks in a sequence.
 - 2) In one example, how to constrain X modes GEO method may be dependent on the block dimensions.
 - a) In one example, a constraint flag may be signaled in SPS level to constrain whether X modes GEO method is used for blocks with condition C.
 - i. C may be illustrated as: blocks with $H/W \leq T$ (e.g., $T = 1$ or 2 or 4 or 8).
 - ii. C may be illustrated as: blocks with $H/W > T$ (e.g., $T = 1$ or 2 or 4 or 8).
 - b) In one example, two constraint flags may be signaled in SPS level. One is to constrain whether X modes GEO method is used for blocks with condition C. The other is to constrain whether Y modes GEO method is used for blocks with condition D.
 - i. C may be blocks with $H/W \leq T$ (e.g., $T = 1$ or 2 or 4 or 8), while D may be blocks with $H/W > T$ (e.g., $T = 1$ or 2 or 4 or 8).
- f) In one example, which GEO modes are allowed for a block may be dependent on the aforementioned syntax elements (e.g., flag).
 - i. In one example, whether a subset of GEO modes or a full set of GEO modes are allowed for a block may be dependent may be on the afore-mentioned syntax elements (e.g., flag).
 - ii. In one example, whether a subset of GEO angles or a full set of GEO angles are allowed for a block may be dependent may be on the afore-mentioned syntax elements (e.g., flag).

- iii. In one example, whether a subset of GEO displacements or a full set of GEO displacements are allowed for a block may be dependent on the afore-mentioned syntax elements (e.g., flag).
 - 1) In one example, whether the GEO modes with non-zero displacement indexes are used or not may be dependent on the afore-mentioned syntax elements (e.g., flag).
- 10. Multiple sets of allowed GEO modes may be utilized for processing a video unit (e.g., a picture/slice/tile/brick/CTU row/CTU).
 - a) In one example, selection of a set from the multiple sets may be dependent on decoded information (e.g., block dimension/block shape of a block).
 - b) In one example, at least two sets among the multiple sets are with different numbers of allowed GEO modes.
 - c) In one example, T (such as T=2) sets among the multiple sets may be with the same number of allowed GEO modes, however, at least one GEO mode included in one set is excluded in another set.
 - d) In one example, T (such as T=2) sets among the multiple sets may be with the same GEO modes, however, at least one GEO mode is arranged in a different position for any two of the T sets.
 - e) In one example, how to signal a GEO mode index may depend on the corresponding set of allowed GEO modes, such as the number of allowed GEO modes in the set.
 - f) In one example, the decoded GEO mode index may be corresponding to different GEO mode (e.g., different angles or different distances).
 - i. In one example, how to map the decoded GEO mode index to a GEO mode may depend on the corresponding set of a block.
 - g) In one example, the number of GEO modes that can be used for a block in the bitstream may be defined as a number (denoted as B) which may be smaller than A (e.g., A = 81 as in the decoding process of the working draft of JVET-P0884-v8).
 - a. For example, B may be a constant value for any GEO block regardless of block dimensions.
 - b. For example, B may be a variable that may be changed from different blocks depending on the block dimensions.

- h) In one example, the number of GEO modes that can be signaled for a block in the bitstream may be defined as a number (denoted as C) which may be smaller than A.
 - c. For example, C may be a constant value for any GEO block regardless of block dimensions.
 - d. For example, C may be a variable that may be changed from different blocks depending on the block dimensions.
 - e. For example, B may be equal to C.
 - f. For example, B or C may be equal to 30 or 40 or 45 or 50.
- i) In one example, B or C may be signaled from the encoder to the decoder.
- j) In one example, two sets (e.g., set A and set B) of allowed GEO modes may be defined for processing GEO-coded blocks.
 - i. In one example, at least one GEO mode included in set A may be excluded in set B.
 - 1) In one example, at least one GEO angle derived from the GEO modes in set A may be excluded in GEO angles derived from the GEO modes in set B.
 - ii. In one example, set A and set B may be with same number of allowed GEO modes, e.g., X (such as X=16 or 32 or 30) modes are used for either set.
 - 1) In one example, set A and set B may be with same number of allowed GEO angles, e.g., Y (such as Y < 24) angles are used for either set.
 - 1) In one example, set A and set B may be with different number of allowed GEO modes, e.g., X1 (such as X1=16) modes used for set A, while X2 (such as X2=32) modes used for set B. In one example, set A and set B may be with different number of allowed GEO angles, e.g., Y1 angles used for set A, while Y2 angles used for set B, such as $Y1 \neq Y2$, $Y1 < 24$, $Y2 < 24$.
 - iii. In one example, whether a block uses GEO modes/angles/distances from set A or set B may be dependent on block dimensions, such as whether the block dimension satisfies condition C.
 - 1) C may be illustrated as: blocks with $H/W \leq T$ (e.g., $T = 1$ or 2 or 4 or 8).
 - 2) C may be illustrated as: blocks with $H/W > T$ (e.g., $T = 1$ or 2 or 4 or 8).
 - iv. In one example, how to signal the GEO mode index for a block may be dependent on the block dimensions.

- 1) In one example, the cMax value for TR coding for GEO mode index may be equal to X (such as X=16 or 32 or 30), given that the $H/W \leq T$ (e.g., $T = 1$ or 2 or 4 or 8).
- v. Suppose I denotes the total number of GEO mode sets, Set_i ($i = 0 \dots I-1$) denotes the GEO mode set used for a block, L_i ($i = 0 \dots I-1$) denotes the length of Set_i . In one example, the GEO-coded block may be classified into multiple block categories, according to decoded information (e.g., related syntax elements, block dimensions).
 - 1) In one example, which GEO mode set is used for a block may be dependent on the block categories, and/or syntax elements (such as the flag described in bullet 9).
 - 2) In one example, how many GEO modes allowed for a block may be dependent on the block categories, and/or syntax elements (such as the flag described in bullet 9).
 - 3) Suppose the corresponding GEO mode set for a block is denoted as GEO mode set i (e.g., Set_i)
 - a) In one example, the number of allowable GEO modes for this block may be less than the length of Set_i , that is, less than L_i .
 - b) In one example, the number of allowable GEO modes for this block may be equal to the length of Set_i , that is, equal to L_i .
 - c) In one example, all allowable GEO modes for this block may be from the corresponding GEO mode set i (e.g., Set_i).
 - d) In one example, a part of GEO modes allowed for this block may be from the corresponding GEO mode set i (e.g., Set_i).
 - e) In one example, the allowable GEO modes for this block may include at least N (such as $N < L_i$) modes in the corresponding GEO mode set (e.g., Set_i).
 - i. In one example, the first N (such as N=16 or 14) modes in the corresponding GEO mode set may be used.
 - ii. In one example, the last N (such as N=16 or 14) modes in the corresponding GEO mode set may be used.
 - iii. In one example, one out of every M (such as M=2) modes in the corresponding GEO mode set may be used.

- f) In one example, the allowable GEO modes for this block may consist of some modes in the corresponding GEO mode set and some other predefined GEO modes (such as the GEO modes with zero displacement, e.g., distance index is equal to 0).
11. How to map the GEO mode index to angle/distance index may be dependent on the decoded information (e.g., related syntax elements, block dimensions).
- a) In one example, how to map the GEO mode index to angle/distance index may be dependent on whether the block dimension satisfies condition C.
 - i. C may be illustrated as: blocks with $H/W \leq T$ (e.g., $T = 1$ or 2 or 4 or 8).
 - ii. C may be illustrated as: blocks with $H/W > T$ (e.g., $T = 1$ or 2 or 4 or 8).
 - b) Suppose I denotes the total number of allowed GEO modes for a block, J denotes the total number of allowed GEO angles for a block, K denotes the total number of allowed GEO distances for a block, M_i ($i = 0 \dots I-1$) denotes the coded/signaled GEO mode index for a block, A_j ($j = 0 \dots J-1$) denotes the mapped angle index for a block, D_k ($k = 0 \dots K-1$) denotes the distance index for a block.
 - i. In one example, the mapped angle index A_j may not rise with the value of GEO mode index M_i .
 - 1) In one example, for multiple consecutive coded/signaled GEO mode indexes M_i , the corresponding angle indexes A_j may be not consecutive numbers, and/or not in descending order, and/or not in ascending order, and/or out-of-order.
 - 2) Alternatively, for multiple consecutive coded/signaled GEO mode indexes, the corresponding angle indexes A_j may be consecutive numbers, and/or in descending order, and/or in ascending order.
 - ii. In one example, the mapped distance index D_k may not rise with the value of GEO mode index M_i .
 - 1) In one example, for multiple consecutive coded/signaled GEO mode indexes M_i , the corresponding distance indexes D_k may be not consecutive numbers, and/or not in descending order, and/or not in ascending order, and/or out-of-order.

- 2) Alternatively, for multiple consecutive coded/signaled GEO mode indexes, the corresponding distance indexes D_k may be consecutive numbers, and/or in descending order, and/or in ascending order.
- iii. In one example, if the coded/signaled GEO mode index is mapped to another set of mapped GEO mode index, the mapped GEO mode index may not rise with the coded/signaled GEO mode index.
 - 1) In one example, for multiple consecutive coded/signaled GEO mode indexes, the corresponding mapped GEO mode indexes may be not consecutive numbers, and/or not in descending order, and/or not in ascending order, and/or out-of-order.
 - 2) Alternatively, for multiple consecutive coded/signaled GEO mode indexes, the corresponding mapped GEO mode indexes may be consecutive numbers, and/or in descending order, and/or in ascending order.
- 12. The number of allowed modes/angles/distances for a GEO block may be different from the number of probable GEO modes/angles/distances for a video unit.
 - a) In one example, the maximum GEO mode index signaled for a block may be smaller than the total number of allowed GEO modes for a sequence.
 - b) In one example, the number of GEO angles allowed for a block may be smaller than the total number of allowed GEO angles defined for a sequence.
 - c) In one example, how many numbers of GEO modes/angles/distances allowed for a block may be dependent on the block dimensions (such as W or H or W/H or H/W).
- 13. The number of the GEO modes for different block dimensions (such as block height and/or block width) that can be signaled or used in the bitstream may be different. The total number of the GEO modes that may be used for deriving the angle/distance indexes as defined in JVET-P0884-v8 in the decoding process is denoted as A. The number of GEO modes that can be used for a block in the bitstream may be defined as a number, denoted as B. The number of GEO modes that can be signaled for a block in the bitstream may be defined as a number, denoted as C.
 - a) In one example, B or C may be different from A.
 - i. For example, B may be equal to C.
 - ii. For example, B and C may be smaller than A.

- b) In one example, B or C may be defined to be different for different block categories.
- 1) In one example, the block category may be classified by the ratios of block width and height.
 - a. In one example, for a $W \times H$ block, B or C may be smaller than A if $W/H = 1$ and/or 2 and/or 3 and/or 4 and/or 8.
 - b. In one example, for a $W \times H$ block, B or C may be smaller than A if $H/W = 1$ and/or 2 and/or 3 and/or 4 and/or 8.
 - 2) In one example, the block category may be classified by a function of block width and height, such as the block size, equal to $W * H$.
 - a. In one example, for a $W \times H$ block, B or C may be smaller than A if $W * H > T$ (such as $T = 512/1024/2048/4096$).
 - b. In one example, for a $W \times H$ block, B or C may be smaller than A if $W * H \leq T$ (such as $T = 512/1024/2048/4096$).
 - 3) In one example, the block category may be classified by the block dimensions, e.g., W and/or H.
 - a. In one example, for a $W \times H$ block, B or C may be smaller than A if $W = T1$ and/or $H = T2$ (wherein $T1$ and $T2$ are constant values).
 - b. In one example, for a $W \times H$ block, B or C may be smaller than A if $W > T1$ and/or $H > T2$ (wherein $T1$ and $T2$ are constant values).
 - c. In one example, for a $W \times H$ block, B or C may be smaller than A if $W < T1$ and/or $H < T2$ (wherein $T1$ and $T2$ are constant values).
 - 4) In one example, a set of fixed numbers B_i ($i = 0 \dots N-1$ wherein N denotes the number of block categories as defined in the above bullets) may be defined for B or C for a block category i .
 - a. In one example, B_0 is equal to 40 or 45 or 50 for block Category 0 with block width smaller than 32 and block height smaller than 32. B_1 is equal to 20 or 30 or 40 block Category 1 with block width equal to 32 and block height equal to 32. B_2 is equal to 20 or 30 for block Category 2 with block width larger than 32 and block height larger than 32.
 - 5) B or C for each block category may be signaled from the encoder to the decoder.

- a. Alternatively, B or C for each block category may be predefined for the encoder and the decoder.
- 6) In one example, the width and height of luma block may be used for deriving B or/and C.
- 14. The value of the GEO modes index signaled in the bitstream may be different from the value of the GEO mode index that is used for deriving the angle/distance index in the decoding process.
 - a) In one example, a subset of GEO modes/angles/distances regarding the full set of GEO modes/angles/distances (e.g., the full set of GEO modes is defined in Table 8-10 in the working draft of JVET-P0884) may be used for a certain block category, wherein the block category may be classified by block width and/or block height, as elaborated in the above bullets.
 - b) In one example, a mapping table (e.g. look-up-table) may be used to define the corresponding relationship between the signaled GEO mode index and the mapped GEO mode index (e.g., the mapped GEO mode may be used for deriving the angle index and distance indexes, such as the **wedge_partition_idx** in the Table 8-10 of the decoding process provided by the working draft of JVET-P0884).
 - c) In one example, N mapping tables ($N > 1$) may be defined, depending on the GEO block categories. For example, N is a constant which may be smaller than 19.
 - a. In one example, the number of mapping tables may be dependent on the number of block categories.
 - b. The length of those mapping tables may be different for different block categories, according to the number of GEO modes allowed for different block categories.
 - d) One or more mapping tables as defined above may be signaled from the encoder to the decoder.
 - a. Alternatively, the mapping tables may be predefined for the encoder and the decoder.
- 15. The binarization for the signalled GEO mode index may be dependent on the decoded information (e.g., block dimensions/categories).
 - a) In one example, the value of maximum value (denoted as cMax) during the binarization of the signalled wedge mode index may be dependent on the block dimensions (such as the block width and/or the block height), or the block categories (as elaborated in the above bullets).

- b) In one example, if the block size satisfies the condition C, then the value of cMax for GEO mode index coding may be equal to X (such as X=16 or 32 or 30).
 - i. C may be illustrated as: blocks with $H/W \leq T$ (e.g., $T = 1$ or 2 or 4 or 8).
 - ii. C may be illustrated as: blocks with $H/W > T$ (e.g., $T = 1$ or 2 or 4 or 8).
- 16. The GEO mode index may be coded with truncated rice, or truncated binary, or truncated unary, or fixed-length, or k-th order Exp-Golomb, or limited k-th order exp-golomb binarization.
 - a) Truncated binary code may be used for the binarization for the signalled GEO mode index.
 - i. In one example, the signalled GEO mode index in the bitstream may be different from the derived GEO mode index that is used for deriving the angle/distance indexes as defined in JVET-P0884-v8 in the decoding process.
 - b) K^{th} -EG coding may be used for the binarization for the signalled GEO mode index.
 - i. In one example, $K = 0$ or 1 or 2 or 3.
- 17. Context coding may be used to code the GEO mode index.
 - a) In one example, the first X (such as X=1) bins of the GEO mode index may be coded by context coding. And other bins may be coded by by-pass coding without context modeling.

Blending weights and motion storage weights generation

- 18. The blending weights and/or motion storage weights for chroma components in TPM and/or GEO mode may depend on the chroma sample location type (e.g., ChromaLocType in FIG. 12.)
 - a) The type of downsampling filter used for blending weights derivation for chroma samples may be signalled at video unit level (such as SPS/VPS/PPS/Picture header/Subpicture/Slice/Slice header/Tile/Brick/CTU/VPDU level).
 - a. In one example, a high-level flag may be signaled to switch between different chroma location types of content.
 - i. In one example, a high-level flag may be signaled to switch between chroma location type 0 and chroma location type 2.
 - ii. In one example, a flag may be signaled for specifying whether the top-left downsampled luma weights in TPM/GEO prediction mode is collocated with the top-left luma weights (i.e., chroma sample location type 0).
 - iii. In one example, a flag may be signaled for specifying whether the top-left downsampled luma sample in TPM/GEO prediction mode is horizontally co-

sited with the top-left luma sample but vertically shifted by 0.5 units of luma samples relatively to the top-left luma sample (i.e., chroma sample location type 2).

- b. In one example, the type of downsampling filter may be signaled for 4:2:0 chroma format and/or 4:2:2 chroma format.
- c. In one example, a flag may be signaled for specifying the type of chroma downsampling filter used for TPM/GEO prediction.
 - i. In one example, the flag may be signaled for whether to use downsampling filter A or downsampling filter B for the chroma weights derivation in TPM/GEO prediction mode.
- b) The type of downsampling filter used for blending weights derivation for chroma samples may be derived at video unit level (such as SPS/VPS/PPS/Picture header/Subpicture/Slice/Slice header/Tile/Brick/CTU/VPDU level).
 - a. In one example, a look up table may be defined to specify the correspondence relationship between the chroma subsampling filter type and the chroma format types of content.
- c) A specified downsampling filter may be used for TPM/GEO prediction mode in case of different chroma location types.
 - a. In one example, chroma weights of TPM/GEO may be subsampled from the collocated top-left luma weights in case of a certain chroma sample location type (e.g., chroma sample location type 0).
 - b. In one example, in case of a certain chroma sample location type (e.g., chroma sample location type 0 or 2), a specified X-tap filter (X is a constant such as X = 6 or 5) may be used for chroma weights subsampling in TPM/GEO prediction mode.

Reduced GEO angles/distances

19. The number and/or candidates of allowed GEO modes/angles/displacements/partitions/dividing patterns for a block may be dependent on decoded information (e.g., block dimensions, and/or syntax elements, and etc.), and/or derived information (e.g., GEO angles).
 - a) In one example, blocks may be classified into blocks types (categories). The number and/or candidates of allowed GEO modes/angles/displacements/partitions/dividing patterns for a

block may be dependent on the type of the block. The type of a block may be dependent on the dimensions of the block.

- b) In one example, the number and/or candidates of allowed displacements for a block may be dependent on the block dimensions.
 - i. In one example, the number of allowed displacements for a type of blocks may be equal to N (such as $N = 1$ or 2 or 3) for blocks with block width equal to X (such as $X = 128$ or 64) and/or block height equal to Y (such as $Y = 128$ or 64)
 - ii. In one example, the number of allowed displacements for a type of blocks may be equal to N (such as $N = 1$ or 2 or 3) for blocks with the ratio of block width and height equal to T (such as $T = 1/8$, and/or 8 , and/or $1/4$, and/or 4) or in a range (such as $T \leq 1/4$, $T \geq 4$, $T \leq 1/8$, $T \geq 8$).
- c) In one example, the number and/or candidates of allowed displacements for a block may be dependent on the derived GEO angles.
 - i. In one example, the number of allowed displacements may be different for different GEO angles.
 - 1) For example, suppose there are N (such as $N > 1$) subsets of angles supported in GEO mode, the number of allowed displacements may be equal to X (such as $X = 1$ or 2 or 3) for a subset of angles, while the number of allowed displacements may be equal to Y (such as $Y = 4$) for another subset of angles.
- d) In one example, the number of GEO modes X for a block may be derived by syntax elements (e.g., a syntax flag, and/or syntax parameter, and etc.)
 - i. The syntax element(s) may be signaled in VPS/DPS/SPS/PPS/sequence header/picture header/slice header/CTU/CU.
 - ii. Whether to use X or Y numbers of GEO modes may be dependent on a syntax flag.
 - 1) For example, the number X may be smaller than the number Y .
 - 2) For example, $X = 2$ or 4 or 6 or 8 or 12 or 14 or 16 .
 - 3) For example, $Y = 16$ or 32 or 48 or 54 or 64 or 82 .
 - iii. Whether to use $X0$ or $X1$ or $X2$ numbers of GEO modes may be dependent on a syntax element.
 - 1) For example, $X0$ may be smaller than $X1$, and/or $X1$ may be smaller than $X2$.

- 2) For example, $X_0 = 2$ or 4 or 5 or 8 or 12 or 14.
 - 3) For example, $X_1 = 8$ or 12 or 14 or 16.
 - 4) For example, $X_2 = 32$ or 64 or 82.
20. Which GEO modes/angles/displacements are allowed for a block may be dependent on syntax element (e.g., a syntax flag or a syntax parameter), and/or block dimensions.
- a) The syntax element may be signaled in VPS/DPS/SPS/PPS/sequence header/picture header/slice header/CTU/CU.
 - b) In one example, whether non-diagonal angles are allowed for a block may be dependent on a syntax element (e.g., a syntax flag), and/or block dimensions.
 - i. For example, whether a first set A of GEO modes or a second set B of GEO modes are allowed for a block may be dependent on the syntax element. E.g. Set A may have less elements than set B.
 - 1) In one example, A may indicate the GEO modes that projected to diagonal angles with displacement equal to 0.
 - 2) In one example, A may indicate the GEO modes that projected to diagonal angles with all displacements.
 - 3) In one example, B may indicate the GEO modes that projected to both diagonal angles and non-diagonal angles.
 - c) In one example, whether non-zero displacements are allowed for a block may be dependent on a syntax element (e.g., a syntax flag), and/or block dimensions.
 - i. For example, selected X GEO modes in the GEO mode mapping table (e.g., Table 36 in JVET-Q0160_CE4_1_CommonBaseWD_w_fixes) may be with zero displacement (such as distanceIdx = 0). For example, the selected X modes may be the first X modes in the GEO mode mapping table.
 - 1) In one example, $X=2$ or 4 or 6 or 8 or 12 or 14 or 16 or 20 or 24 or 32.
 - 2) For example, if the number of allowed GEO modes for a block derived by the syntax element is equal to X, the first X GEO modes in the GEO mapping table are used and their corresponding displacements may be equal to zero.
 - 3) In one example, the GEO modes which corresponding to zero displacement (such as distanceIdx = 0) may be placed at the first of the GEO mode mapping table.

- 4) In one example, the GEO modes which corresponding to non-zero displacements (such as distanceIdx = 1 or 2 or 3) may be placed after the GEO modes that corresponding to zero displacement (such as distanceIdx = 0).
- d) In one example, the allowed GEO modes/angles/displacements may be different for different block categories. Suppose a subset of GEO modes/angles/displacements S_i ($i = 0 \dots N-1$, where N the total number of block categories) are allowed for a block from block category C_i .
- In one example, whether subset S_i ($i = 0 \dots N-1$) or subset S_j ($j = 0 \dots N-1$) is used for a block may be dependent on the ratios of block width and height (such as width/height).
 - In one example, whether subset S_i ($i = 0 \dots N-1$) or subset S_j ($j = 0 \dots N-1$) is used for a block may be dependent on the block category, where the block category may be classified by the relationship between block width and height (such as width > height, and/or width < height, and/or width = height).
 - In one example, at least one element in S_i ($i = 0 \dots N-1$) may be not included in S_j ($j = 0 \dots N-1$).
- e) Exemplary spec change based on JVET-Q0160_CE4_1_CommonBaseWD_w_fixes is as below, newly added parts are highlighted with **underline bold**, and the deleted parts from VVC working draft are marked with double brackets (e.g., $[[a]]$ denotes the deletion of the character “a”).

7.3.2.3 Sequence parameter set RBSP syntax

<i>seq_parameter_set_rbsp()</i> {	Descriptor
<i>sps_decoding_parameter_set_id</i>	$u(4)$
<i>sps_video_parameter_set_id</i>	$u(4)$
..	
<i>sps_ciip_enabled_flag</i>	$u(1)$
<i>if(sps_mmvd_enabled_flag)</i>	
<i>sps_fpel_mmvd_enabled_flag</i>	$u(1)$
<i>sps_geo_enabled_flag</i>	$u(1)$
<u>if(sps_geo_enabled_flag)</u>	
<u>sps_X_modes_geo_enabled_flag</u>	<u>$u(1)$</u>
<i>sps_lmcs_enabled_flag</i>	$u(1)$

<i>sps_lfst_enabled_flag</i>	<i>u(1)</i>
...	
<i>rbsp_trailing_bits()</i>	
}	

sps_X_modes_geo_enabled_flag equal to 1 specifies that X modes geo is used for specified blocks. *sps_X_modes_geo_enabled_flag* equal to 0 specifies that Y modes geo is used for specified blocks.

[notes:above X and Y may be integer numbers, such as X=2 or 4 or 6 or 8 or 12 or 14 or 16; and Y = 32 or 64 or 82].

7.4.10.7 Merge data semantics

merge_geo_partition_idx[x0][y0] specifies the geometric splitting direction of the merge geometric mode. The array indices *x0*, *y0* specify the location (*x0*, *y0*) of the top-left luma sample of the considered coding block relative to the top-left luma sample of the picture.

When *merge_geo_partition_idx[x0][y0]* is not present, it is inferred to be equal to 0.

It is constraint that the maximum value of *merge_geo_partition_idx* should be less than (*sps_16_modes_geo_enabled_flag* && *cbWidth* >= *cbHeight*) ? X : Y.

f) Another exemplary spec change based on JVET-

Q0160_CE4_1_CommonBaseWD_w_fixes is as below:

8.5.7 Decoding process for geo inter blocks

8.5.7.1 General

This process is invoked when decoding a coding unit with *MergeGeoFlag[xCb][yCb]* equal to 1.

Inputs to this process are:

- a luma location (*xCb*, *yCb*) specifying the top-left sample of the current coding block relative to the top-left luma sample of the current picture,
- a variable *cbWidth* specifying the width of the current coding block in luma samples,
- a variable *cbHeight* specifying the height of the current coding block in luma samples,
- the luma motion vectors in 1/16 fractional-sample accuracy *mvA* and *mvB*,
- the chroma motion vectors *mvCA* and *mvCB*,
- the reference indices *refIdxA* and *refIdxB*,
- the prediction list flags *predListFlagA* and *predListFlagB*.

Outputs of this process are:

- an $(cbWidth) \times (cbHeight)$ array $predSamples_L$ of luma prediction samples,
- an $(cbWidth / SubWidthC) \times (cbHeight / SubHeightC)$ array $predSamples_{Cb}$ of chroma prediction samples for the component Cb,
- an $(cbWidth / SubWidthC) \times (cbHeight / SubHeightC)$ array $predSamples_{Cr}$ of chroma prediction samples for the component Cr.

Let $predSamplesLA_L$ and $predSamplesLB_L$ be $(cbWidth) \times (cbHeight)$ arrays of predicted luma sample values and, $predSamplesLA_{Cb}$, $predSamplesLB_{Cb}$, $predSamplesLA_{Cr}$ and $predSamplesLB_{Cr}$ be $(cbWidth / SubWidthC) \times (cbHeight / SubHeightC)$ arrays of predicted chroma sample values.

The $predSamples_L$, $predSamples_{Cb}$ and $predSamples_{Cr}$ are derived by the following ordered steps:

7. For N being each of A and B, the following applies:

- The reference picture consisting of an ordered two-dimensional array $refPicLN_L$ of luma samples and two ordered two-dimensional arrays $refPicLN_{Cb}$ and $refPicLN_{Cr}$ of chroma samples is derived by invoking the process specified in clause 8.5.6.2 with X set equal to $predListFlagN$ and $refIdxX$ set equal to $refIdxN$ as input.
- The array $predSamplesLN_L$ is derived by invoking the fractional sample interpolation process specified in clause 8.5.6.3 with the luma location (xCb, yCb) , the luma coding block width $sbWidth$ set equal to $cbWidth$, the luma coding block height $sbHeight$ set equal to $cbHeight$, the motion vector offset $mvOffset$ set equal to $(0, 0)$, the motion vector $mvLX$ set equal to mvN and the reference array $refPicLX_L$ set equal to $refPicLN_L$, the variable $bdofFlag$ set equal to FALSE, the variable $cIdx$ is set equal to 0, and $RefPicScale[predListFlagN][refIdxN]$ as inputs.
- The array $predSamplesLN_{Cb}$ is derived by invoking the fractional sample interpolation process specified in clause 8.5.6.3 with the luma location (xCb, yCb) , the coding block width $sbWidth$ set equal to $cbWidth / SubWidthC$, the coding block height $sbHeight$ set equal to $cbHeight / SubHeightC$, the motion vector offset $mvOffset$ set equal to $(0, 0)$, the motion vector $mvLX$ set equal to $mvCN$, and the reference array $refPicLX_{Cb}$ set equal to $refPicLN_{Cb}$, the variable $bdofFlag$ set equal to FALSE, the variable $cIdx$ is set equal to 1, and $RefPicScale[predListFlagN][refIdxN]$ as inputs.

- The array $\text{predSamplesLN}_{Cr}$ is derived by invoking the fractional sample interpolation process specified in clause 8.5.6.3 with the luma location (x_{Cb}, y_{Cb}) , the coding block width $sbWidth$ set equal to $cbWidth / \text{SubWidthC}$, the coding block height $sbHeight$ set equal to $cbHeight / \text{SubHeightC}$, the motion vector offset $mvOffset$ set equal to $(0, 0)$, the motion vector $mvLX$ set equal to $mvCN$, and the reference array refPicLX_{Cr} set equal to refPicLN_{Cr} , the variable $bdofFlag$ set equal to `FALSE`, the variable $cIdx$ is set equal to 2, and $\text{RefPicScale}[\text{predListFlagN}][\text{refIdxN}]$ as inputs.
- 8. The partition angle and distance of merge geo mode variable angleIdx and distanceIdx are set according to Table 36, with the value of $\text{merge_geo_partition_idx}[x_{Cb}][y_{Cb}]$, and the variable isNarrowBlk set equal to $cbHeight > cbWidth$ as inputs.
- 9. The partition angle and distance of merge geo mode variable angleIdx and distanceIdx are set according to [[the value of $\text{merge_geo_partition_idx}[x_{Cb}][y_{Cb}]$ as specified in]] Table 36, with the value of $\text{merge_geo_partition_idx}[x_{Cb}][y_{Cb}]$, and the variable isNarrowBlk set equal to $cbHeight > cbWidth$ as inputs.
- 10. The prediction samples inside the current luma coding block, $\text{predSamples}_L[x_L][y_L]$ with $x_L = 0..cbWidth - 1$ and $y_L = 0..cbHeight - 1$, are derived by invoking the weighted sample prediction process for geo merge mode specified in clause 8.5.7.2 with the coding block width $nCbW$ set equal to $cbWidth$, the coding block height $nCbH$ set equal to $cbHeight$, the sample arrays predSamplesLA_L and predSamplesLB_L , and the variables angleIdx and distanceIdx , and $cIdx$ equal to 0 as inputs.
- 11. The prediction samples inside the current chroma component Cb coding block, $\text{predSamples}_{Cb}[x_C][y_C]$ with $x_C = 0..cbWidth / \text{SubWidthC} - 1$ and $y_C = 0..cbHeight / \text{SubHeightC} - 1$, are derived by invoking the weighted sample prediction process for geo merge mode specified in clause 8.5.7.2 with the coding block width $nCbW$ set equal to $cbWidth / \text{SubWidthC}$, the coding block height $nCbH$ set equal to $cbHeight / \text{SubHeightC}$, the sample arrays $\text{predSamplesLA}_{Cb}$ and $\text{predSamplesLB}_{Cb}$, and the variables angleIdx and distanceIdx , and $cIdx$ equal to 1 as inputs.
- 12. The prediction samples inside the current chroma component Cr coding block, $\text{predSamples}_{Cr}[x_C][y_C]$ with $x_C = 0..cbWidth / \text{SubWidthC} - 1$ and $y_C = 0..cbHeight / \text{SubHeightC} - 1$, are derived by invoking the weighted sample prediction

process for geo merge mode specified in clause 8.5.7.2 with the coding block width $nCbW$ set equal to $cbWidth / SubWidthC$, the coding block height $nCbH$ set equal to $cbHeight / SubHeightC$, the sample arrays $predSamplesLA_{Cr}$ and $predSamplesLB_{Cr}$, and the variables $angleIdx$ and $distanceIdx$, and $cIdx$ equal to 2 as inputs.

13. The motion vector storing process for merge geo mode specified in clause 8.5.7.3 is invoked with the luma coding block location (xCb, yCb), the luma coding block width $cbWidth$, the luma coding block height $cbHeight$, the partition direction $angleIdx$ and $distanceIdx$, the luma motion vectors mvA and mvB , the reference indices $refIdxA$ and $refIdxB$, and the prediction list flags $predListFlagA$ and $predListFlagB$ as inputs.

FIG. 20A shows Table 36 which is now deleted from the relevant working draft.

FIG. 20B shows an example of Table 36 indicating specification of the $angleIdx$ and $distanceIdx$ values based on the $merge_geo_partition_idx$ value. In the example as shown in FIG. 20B, **whether the first N (such as $N = 16$) modes or full set of modes are used may be dependent on syntax elements.**

FIG. 20C shows another example of Table 36 indicating specification of the $angleIdx$ and $distanceIdx$ values based on the $merge_geo_partition_idx$ value. In the example as shown in FIG. 20C, **whether the first N (such as $N = 2$) modes or full set of modes are used may be dependent on syntax elements.**

FIG. 20D shows another example of Table 36 indicating specification of the $angleIdx$ and $distanceIdx$ values based on the $merge_geo_partition_idx$ value. In the example as shown in FIG. 20D, **whether the first N (such as $N = 2$) modes or full set of modes are used may be dependent on syntax elements.**

FIG. 20E shows another example of Table 36 indicating specification of the $angleIdx$ and $distanceIdx$ values based on the $merge_geo_partition_idx$ value. In the example as shown in FIG. 20E, **whether the first N (such as $N = 4$) modes or full set of modes are used may be dependent on syntax elements.**

FIG. 20F shows another example of Table 36 indicating specification of the $angleIdx$ and $distanceIdx$ values based on the $merge_geo_partition_idx$ value. In the example as shown in FIG. 20F, **whether the first N (such as $N = 4$) modes or full set of modes are used may be dependent on syntax elements.**

FIG. 20G shows another example of Table 36 indicating specification of the $angleIdx$ and $distanceIdx$ values based on the $merge_geo_partition_idx$ value. In the example as shown in FIG. 20G, **whether the first N (such as $N = 6$) modes or full set of modes are used may be dependent on syntax elements.**

21. In one example, X (such as $X < N$) elements may be included in both S_i ($i = 0 \dots N-1$) and S_j ($j = 0 \dots N-1$).
22. The number of angles for a GEO block may be less than $T1$ (such as $T1 = 24$). Suppose the number of angles used in the decoding process is denoted as NUM_ANGLE .
 - a) Alternatively, the number of GEO modes may be less than $T2$ (such as $T2 = 82$).
 - i. In one example, the mapping of $angleIdx$ and $distanceIdx$ from $wedge_partition_idx$ may be dependent on how many angles supported for GEO mode and/or how many distances supported for each angle.
 - b) Alternatively, the number of distances for a GEO block may be less than $T3$ (such as $T3 = 4$ or 3).
 - i. In one example, the number of distances for one or more angles may be less than $T3$.
 - 1) For example, the number of distances for vertical and horizontal angles may be equal to X (such as $X=2$)
 - c) In one example, the number of angles, NUM_ANGLE , used in the decoding process may be equal to the max $angleIdx$ plus 1.
 - i. For example, $NUM_ANGLE = 24$, such as the max $angleIdx$ defined in Table 8-10 in the working draft of JVET-P0884 is equal to 23.
 - ii. For another example, $NUM_ANGLE < T1$ (such as $T1 = 24$).
 - d) In one example, the calculation of $displacementY$, that is used in the processes of weighted sample prediction and/or motion vector storing for a GEO mode coded block, may be dependent on the total number of angles used in the decoding process.
 - i. In one example, $displacementY$ may be set to $(displacementX + (NUM_ANGLE >> 2)) \% NUM_ANGLE$.
 - e) In one example, the calculation of $shiftHor$, that is used in the processes of weighted sample prediction and/or motion vector storing for a GEO mode coded block, may be dependent on the total number of angles used in the decoding process.
 - i. In one example, $shiftHor$ may be set to 0 is one of the following conditions is true. Otherwise, $shiftHor$ is set equal to 1.
 - 1) $angleIdx \% (NUM_ANGLE/2)$ is equal to $(NUM_ANGLE >> 2)$

- 2) $\text{angleIdx} \% (\text{NUM_ANGLE}/2)$ is **not equal to 0** and $\text{hwRatio} \geq 1$, wherein hwRatio is set to H/W .
- f) In one example, the derivation of offsetX and/or offsetY for deriving the blending weight index of a GEO block may be dependent on the number of angles and/or the value of shiftHor .
- i. In one example, if shiftHor is equal to 0, offsetY for deriving the blending weight index of a GEO block may be derived as follows:
- 1)
$$\text{offsetY} = (256 - nH) \gg 1 + \text{angleIdx} < (\text{NUM_ANGLE}/2) ? (\text{distanceIdx} * nH) \gg 3 : -((\text{distanceIdx} * nH) \gg 3)$$
- ii. In one example, if shiftHor is equal to 1, offsetX for deriving the blending weight index of a GEO block may be derived as follows:
- 1)
$$\text{offsetX} = (256 - nW) \gg 1 + \text{angleIdx} < (\text{NUM_ANGLE}/2) ? (\text{distanceIdx} * nW) \gg 3 : -((\text{distanceIdx} * nW) \gg 3)$$
- g) In one example, the derivation of offsetX and/or offsetY for deriving the motion index of a GEO block may be dependent on the number of angles and/or the value of shiftHor .
- i. In one example, if shiftHor is equal to 0, offsetY for deriving the motion index of a GEO block may be derived as follows:
- 1)
$$\text{offsetY} = (64 - \text{numSbY}) \gg 1 + \text{angleIdx} < (\text{NUM_ANGLE}/2) ? (\text{distanceIdx} * nCbH) \gg 5 : -((\text{distanceIdx} * nCbH) \gg 5)$$
- ii. In one example, if shiftHor is equal to 1, offsetX for deriving the motion index of a GEO block may be derived as follows:
- 1)
$$\text{offsetX} = (64 - \text{numSbX}) \gg 1 + \text{angleIdx} < (\text{NUM_ANGLE}/2) ? (\text{distanceIdx} * nCbW) \gg 5 : -((\text{distanceIdx} * nCbW) \gg 5)$$
- h) In one example, the length of the look-up table for derivation of GEO partitioning distance may be dependent on the number of angles used in the GEO block decoding process.
- i. In one example, the length of the look-up table for derivation of GEO partitioning distance, as illustrated in Table 8-12 in the working draft of JVET-P0884, may be equal to NUM_ANGLE .
- 1) In one example, $\text{NUM_ANGLE} < 24$.

i) In one example, the values of the look-up table for derivation of GEO partitioning distance may be re-designed and with the length equal to NUM_ANGLE.

1) In one example, the re-designed look-up-table may be a subset of the Table 8-12 in the working draft of JVET-P0884.

2) In one example, the table may be designed as below in case of NUM_ANGLE = 20.

idx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Dis[idx]	8	8	8	4	2	-2	-4	-8	-8	-8	-8	-8	-8	-4	-2	2	4	8	8	8

3) In one example, the table may be designed as below in case of NUM_ANGLE = 20.

idx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Dis[idx]	8	8	8	4	2	0	-2	-4	-8	-8	-8	-8	-8	-4	-2	0	2	4	8	8

4) In one example, the table may be designed as below in case of NUM_ANGLE = 20.

idx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Dis[idx]	8	8	8	8	2	0	-2	-8	-8	-8	-8	-8	-8	-8	-2	0	2	8	8	8

5) In one example, the table may be designed as below in case of NUM_ANGLE = 20.

idx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Dis[idx]	8	8	8	8	4	0	-4	-8	-8	-8	-8	-8	-8	-8	-4	0	4	8	8	8

6) In one example, the table may be designed as below in case of NUM_ANGLE = 16.

idx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Dis[idx]	8	8	4	2	0	-2	-4	-8	-8	-8	-4	-2	0	2	4	8

j) In one example, whether PART1 and PART2 are equal to A or B, used for weighted sample prediction process for GEO mode, may be dependent on angle index T1 and angle index T2, wherein A and B indicate the two input arrays predSamplesLA and predSamplesLB for weighted sample prediction process for GEO mode, and PART1 and PART2 are the representations of A and B for the derivation of the output weighted prediction sample values of a GEO predicted block.

- i. In one example, if $\text{angleIdx} \geq T1$ & $\text{angleIdx} \leq T2$, PART1 and PART2 may be set equal to A and B respectively, otherwise PART1 and PART2 may be set equal to B and A respectively.
- ii. In one example, T1 and T2 may be constant values and $T1 < \text{NUM_ANGLE}$ and $T2 \leq \text{NUM_ANGLE}$.
 - 1) In one example, $T1 = 10$ and $T2 = 20$ in case of $\text{NUM_ANGLE} = 24$.
 - 2) In one example, $T1 = 8$ and $T2 = 16$ in case of $\text{NUM_ANGLE} = 20$.
 - 3) In one example, $T1 = 8$ and $T2 = 17$ in case of $\text{NUM_ANGLE} = 20$.
 - 4) In one example, $T1 = 9$ and $T2 = 16$ in case of $\text{NUM_ANGLE} = 20$.
 - 5) In one example, $T1 = 7$ and $T2 = 13$ in case of $\text{NUM_ANGLE} = 16$.
- iii. In one example T1 and T2 may be calculated based on the number of angles.
- k) In one example, whether the partIdx is set to 0 or 1, used in the motion vectore storing process for GEO mode, may be dependent on angle index T1 and angle index T2, where the partIdx is used to derive the variable sType for assigning the motion vectors for GEO motion storage.
 - i. In one example, if $\text{angleIdx} \geq T1$ & $\text{angleIdx} \leq T2$, partIdx may be set to 1, otherwise partIdx may be set to 0. $\text{sType} = \text{abs}(\text{motionIdx}) < 32 ? 2 : \text{motionIdx} < 0 ? \text{partIdx} : 1 - \text{partIdx}$, where the variable motionIdx is calculated using the look-up table for derivation of GEO partitioning distance (e.g., Table 8-12 in the working draft of JVET-P0884)
 - ii. In one example, T1 and T2 may be constant values and $T1 < \text{NUM_ANGLE}$ and $T2 \leq \text{NUM_ANGLE}$.
 - 1) In one example, $T1 = 10$ and $T2 = 20$ in case of $\text{NUM_ANGLE} = 24$.
 - 2) In one example, $T1 = 8$ and $T2 = 16$ in case of $\text{NUM_ANGLE} = 20$.
 - 3) In one example, $T1 = 8$ and $T2 = 17$ in case of $\text{NUM_ANGLE} = 20$.
 - 4) In one example, $T1 = 9$ and $T2 = 16$ in case of $\text{NUM_ANGLE} = 20$.
 - 5) In one example, $T1 = 7$ and $T2 = 13$ in case of $\text{NUM_ANGLE} = 16$.
 - iii. In one example T1 and T2 may be calculated based on the number of angles.
- l) In one example, the values of the look-up table for derivation of GEO/wedgemetric partitioning distance (such as $\text{Dis}[i]$, $i=0 \dots \text{NUM_ANGLE}-1$) may be set as below table.

- 1) Alternatively, the values of the look-up table for derivation of GEO/wedgemetric partitioning distance (such as Dis[i], $i=0 \dots \text{NUM_ANGLE}-1$) may be set as a subset of below table.
- 2) In one example, the GEO/wedgemetric partitioning distance for angle index equal to 3 and/or 21 may be equal to 4.
- 3) In one example, the GEO/wedgemetric partitioning distance for angle index equal to 9 and/or 15 may be equal to -4.

idx	0	1	2	3	4	5	6	7	8	9	10	11
Dis[idx]	8	8	8	4	4	2	0	-2	-4	-4	-8	-8
idx	12	13	14	15	16	17	18	19	20	21	22	23
Dis[idx]	-8	-8	-8	-4	-4	-2	0	2	4	4	8	8

GEO angle derivation

23. The derivation of GEO mode indices and/or GEO angle indices and/or GEO distance indices and/or GEO displacements and/or GEO blending weights / GEO motion weights may be dependent on the block dimensions (such as functions with block width and/or height).

- a) In one example, the derivation of GEO mode indices and/or angle indices and/or distance indices may be dependent on the value X.
 - i. In one example, X may depend on $\log_2(\text{height})$ and/or $\log_2(\text{width})$.
 - ii. In one example, X may be $\log_2(\text{height}) - \log_2(\text{width})$.
 - iii. In one example, X may be $\log_2(\text{height}) + \log_2(\text{width})$.
 - iv. In one example, X may be $\log_2(\text{width}) - \log_2(\text{height})$.
 - v. In one example, X may be $\text{abs}(\log_2(\text{width}) - \log_2(\text{height}))$.
 - vi. In one example, X may be $\text{Clip3}(M, N, \log_2(\text{height}) - \log_2(\text{width}))$, where M and N are constants such as $M=-2, N=2$.
 - vii. In one example, X may be $\text{Clip3}(M, N, \log_2(\text{width}) - \log_2(\text{height}))$, where M and N are constants such as $M=-2, N=2$.
 - viii. In one example, X may be $\text{Clip3}(M, N, \log_2(\text{height}) + \log_2(\text{width}))$, where M and N are constants such as $M=-2, N=2$.
 - ix. In one example, X may be $\text{Clip3}(M, N, \text{abs}(\log_2(\text{height}) - \log_2(\text{width})))$, where M and N are constants such as $M=-2, N=2$.

- b) An exemplary spec change based on JVET-Q0268 is as below, newly added parts are highlighted with **underline bold**, and the deleted parts from VVC working draft are marked with double brackets(e.g., [[a]] denotes the deletion of the character “a”).

8.5.7.2. Weighted sample prediction process for geo merge mode

Inputs to this process are:

- two variables $nCbW$ and $nCbH$ specifying the width and the height of the current coding block,
- two $(nCbW) \times (nCbH)$ arrays $predSamplesLA$ and $predSamplesLB$,
- a variable $angleSquareIdx$ specifying the angle index of the geo partition for square blocks,
- a variable $distanceIdx$ specifying the distance index of the geo partition,
- a variable $cIdx$ specifying colour component index.

Output of this process is the $(nCbW) \times (nCbH)$ array $pbSamples$ of prediction sample values.

The variable nW , nH and $nCbR$ is derived as follows:

$$nCbWY = (cIdx == 0) ? nCbW : nCbW * SubWidthC \quad (1027)$$

$$nCbHY = (cIdx == 0) ? nCbH : nCbH * SubHeightC \quad (1028)$$

Variables $shift1$ and $offset1$ are derived as follows:

- The variable $shift1$ is set equal to $\text{Max}(5, 17 - \text{BitDepth})$.
- The variable $offset1$ is set equal to $1 \ll (shift1 - 1)$.

The values of the following variables are set:

- $hwRatio$ is set to nH / nW
- $hwRatioLog2$ is set to $\text{Clip3}(-2, 2, \lfloor \log_2(hwRatio) \rfloor \lfloor \log_2(nH) - \log_2(nW) \rfloor)$
- $angleIdx$ is set to $angleSquareIdx$

If $angleSquareIdx \% 6$ is not equal to 0, then

$$angleIdx += (angleSquareIdx \% 12 < 6) ? -hwRatioLog2 : hwRatioLog2$$

$$angleIdx = (angleIdx \% 6 == 0) ? ((angleSquareIdx \% 12 < 6) ? angleSquareIdx + (hwRatioLog2 >> 1) : angleSquareIdx - (hwRatioLog2 >> 1)) : angleIdx$$

Otherwise (if $angleSquareIdx \% 6$ is equal to 0), and if $hwRatioLog2$ is less than 0, then

$$angleIdx += (angleSquareIdx \% 12 < 6) ? 6 : -6 \quad \text{displacementX is set to } angleIdx$$

- $displacementY$ is set to $(displacementX + 6) \% 24$
- If $angleIdx \geq 10$ & $angleIdx \leq 20$, $PART1$ and $PART2$ are set equal to A and B respectively, otherwise $PART1$ and $PART2$ are set equal to B and A respectively.

If one of the following conditons is true, variable *shiftHor* is set equal to 0:

angleIdx % 12 is equal to 6

angleIdx % 12 is not equal to 0 and *hwRatio* ≥ 1

Otherwise, *shiftHor* is set equal to 1.

If *shiftHor* is equal to 0, *offsetX* and *offsetY* are derived as follows:

$offsetX = (-nW) \gg 1$

$offsetY = ((-nH) \gg 1) + (angleIdx < 12 ? (distanceIdx * nH) \gg 3 : -((distanceIdx * nH) \gg 3))$

Otherwise, if *shiftHor* is equal to 1, *offsetX* and *offsetY* are derived as follows:

$offsetX = ((-nW) \gg 1) + (angleIdx < 12 ? (distanceIdx * nW) \gg 3 : -((distanceIdx * nW) \gg 3))$

$offsetY = (-nH) \gg 1$

– The variables *xL* and *yL* are derived as follows:

$xL = (cIdx == 0) ? x : x * SubWidthC$

$yL = (cIdx == 0) ? y : y * SubHeightC$

– The variable *weightIdx* and *weightIdxAbs* specifying the weight of the prediction sample is derived as follows:

– The variable *weightIdx* and *weightIdxAbs* are calculated using the look-up table Table 37 as follows:

$weightIdx = (((xL + offsetX) < 1) + 1) * Dis[displacementX] + (((yL + offsetY) < 1) + 1) * Dis[displacementY]$

$weightIdxAbs = Clip3(0, 26, abs(weightIdx))$.

– The value of *wValue* is derived according to according to Table 38 as follows:

$wValue = weightIdx \leq 0 ? GeoFilter[weightIdxAbs] : 8 - GeoFilter[weightIdxAbs]$

NOTE – The value of sample *wValue* with located at (*x*, *y*) can also be derived from *wValue* located at (*x-shiftX*, *y-shiftY*). If the *angleIdx* is larger than 4 and smaller than 12, or *angleIdx* is larger than 20 and smaller than 24, *shiftX* is the tangent of the split angle and *shiftY* is 1, otherwise *shiftX* is 1 of the split angle and *shiftY* is cotangent of the split angle. If tangent (resp. cotangent) value is infinity, *shiftX* is 1 (resp. 0) or *shiftY* is 0 (reps. 1).

– The prediction sample values are derived as follows:

$$\begin{aligned}
 pbSamples[x][y] = & Clip3(0, (1 \ll BitDepth) - 1, \\
 & (predSamplesLPART1[x][y] * wValue + \\
 & \quad predSamplesLPART2[x][y] * (8 - wValue) + offset1) \gg shift1 \\
 &)
 \end{aligned}
 \tag{1032}$$

Table 37 - Look-up table Dis for derivation of geoc partitioning distance.

idx	0	1	2	3	4	5	6	7	8	9	10	11
Dis[idx]	8	8	8	4	4	2	0	-2	-4	-4	-8	-8
idx	12	13	14	15	16	17	18	19	20	21	22	23
Dis[idx]	-8	-8	-8	-4	-4	-2	0	2	4	4	8	8

Table 38 - Filter weight look-up table GeoFilter for derivation of geo partitioning filter weights.

idx	0	1	2	3	4	5	6	7	8	9	10	11	12	13
GeoFilter[idx]	4	4	4	4	5	5	5	5	5	5	5	6	6	6
idx	14	15	16	17	18	19	20	21	22	23	24	25	26	
GeoFilter[idx]	6	6	6	6	7	7	7	7	7	7	7	7	8	

c) an exemplary spec change based on JVET-Q0268 is as below:

8.5.7.2. Weighted sample prediction process for geo merge mode

Inputs to this process are:

- two variables $nCbW$ and $nCbH$ specifying the width and the height of the current coding block,
- two $(nCbW) \times (nCbH)$ arrays $predSamplesLA$ and $predSamplesLB$,
- a variable $angleSquareIdx$ specifying the angle index of the geo partition for square blocks,
- a variable $distanceIdx$ specifying the distance index of the geo partition,
- a variable $cIdx$ specifying colour component index.

Output of this process is the $(nCbW) \times (nCbH)$ array $pbSamples$ of prediction sample values.

The variable nW , nH and $nCbR$ is derived as follows:

$$nCbWY = (cIdx == 0) ? nCbW : nCbW * SubWidthC \tag{1027}$$

$$nCbHY = (cIdx == 0) ? nCbH : nCbH * SubHeightC \tag{1028}$$

Variables $shift1$ and $offset1$ are derived as follows:

- The variable $shift1$ is set equal to $\text{Max}(5, 17 - \text{BitDepth})$.
- The variable $offset1$ is set equal to $1 \ll (shift1 - 1)$.

The values of the following variables are set:

- $hwRatio$ is set to nH / nW
- $hwRatioLog2$ is set to $[\log_2(hwRatio)] - \log_2(nH) - \log_2(nW)$
- $angleIdx$ is set to $angleSquareIdx$

If $angleSquareIdx \% 8$ is not equal to 0, then

$angleIdx += (angleSquareIdx \% 16 < 8) ? -hwRatioLog2 : hwRatioLog2$

$angleIdx = (angleIdx \% 8 == 0) ? angleSquareIdx : angleIdx$

Otherwise (if $angleSquareIdx \% 8$ is equal to 0), and if $hwRatioLog2$ is less than 0, then

$angleIdx += (angleSquareIdx \% 16 < 8) ? 8 : -8$ — $displacementX$ is set to $angleIdx$

- $displacementY$ is set to $(displacementX + 6) \% 24$
- If $angleIdx \geq 10$ & $angleIdx \leq 20$, $PART1$ and $PART2$ are set equal to A and B respectively, otherwise $PART1$ and $PART2$ are set equal to B and A respectively.

If one of the following conditions is true, variable $shiftHor$ is set equal to 0:

$angleIdx \% 12$ is equal to 6

$angleIdx \% 12$ is not equal to 0 and $hwRatio \geq 1$

Otherwise, $shiftHor$ is set equal to 1.

If $shiftHor$ is equal to 0, $offsetX$ and $offsetY$ are derived as follows:

$offsetX = (-nW) >> 1$

$offsetY = ((-nH) >> 1) + (angleIdx < 12 ? (distanceIdx * nH) >> 3 : -((distanceIdx * nH) >> 3))$

Otherwise, if $shiftHor$ is equal to 1, $offsetX$ and $offsetY$ are derived as follows:

$offsetX = ((-nW) >> 1) + (angleIdx < 12 ? (distanceIdx * nW) >> 3 : -((distanceIdx * nW) >> 3))$

$offsetY = (-nH) >> 1$

- The variables xL and yL are derived as follows:

$xL = (cIdx == 0) ? x : x * SubWidthC$

$yL = (cIdx == 0) ? y : y * SubHeightC$

- The variable $weightIdx$ and $weightIdxAbs$ specifying the weight of the prediction sample is derived as follows:

- The variable *weightIdx* and *weightIdxAbs* are calculated using the look-up table Table 37 as follows:

$$\text{weightIdx} = (((xL + \text{offsetX}) \ll 1) + 1) * \text{Dis}[\text{displacementX}] \\ + (((yL + \text{offsetY}) \ll 1) + 1) * \text{Dis}[\text{displacementY}].$$

$$\text{weightIdxAbs} = \text{Clip3}(0, 26, \text{abs}(\text{weightIdx})).$$

- The value of *wValue* is derived according to according to Table 38 as follows:

$$wValue = \text{weightIdx} \leq 0 ? \text{GeoFilter}[\text{weightIdxAbs}] : 8 - \text{GeoFilter}[\text{weightIdxAbs}]$$

NOTE – The value of sample *wValue* with located at (x, y) can also be derived from *wValue* located at (x-shiftX, y-shiftY). If the *angleIdx* is larger than 4 and smaller than 12, or *angleIdx* is larger than 20 and smaller than 24, *shiftX* is the tangent of the split angle and *shiftY* is 1, otherwise *shiftX* is 1 of the split angle and *shiftY* is cotangent of the split angle. If tangent (resp. cotangent) value is infinity, *shiftX* is 1 (resp. 0) or *shiftY* is 0 (reps. 1).

- The prediction sample values are derived as follows:

$$\text{pbSamples}[x][y] = \text{Clip3}(0, (1 \ll \text{BitDepth}) - 1, \\ (\text{predSamplesLPART1}[x][y] * wValue +$$

$$\text{predSamplesLPART2}[x][y] * (8 - wValue) + \text{offset1}) \gg \text{shift1}$$

)

Table 37 - Look-up table *Dis* for derivation of geoc partitioning distance.

<i>idx</i>	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
<i>Dis[idx]</i>	8	8	8	8	4	4	2	1	0	-1	-2	-4	-4	-8	-8	-8
<i>idx</i>	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
<i>Dis[idx]</i>	-8	-8	-8	-8	-4	-4	-2	-1	0	1	2	4	4	8	8	8

Table 38 - Filter weight look-up table *GeoFilter* for derivation of geo partitioning filter weights.

<i>idx</i>	0	1	2	3	4	5	6	7	8	9	10	11	12	13
<i>GeoFilter[idx]</i>	4	4	4	4	5	5	5	5	5	5	5	6	6	6
<i>idx</i>	14	15	16	17	18	19	20	21	22	23	24	25	26	
<i>GeoFilter[idx]</i>	6	6	6	6	7	7	7	7	7	7	7	7	8	

GEO combined with other coding tools

24. The coding tool X may be used to GEO-coded blocks. In this case, indications of usage/side information of X and GEO may be both signaled.

- a) In one example, X may be SBT.
 - b) In one example, X may be CIIP.
 - c) In one example, X may be MMVD.
 - d) The GEO process may be different when coding X is used or not. For example, GEO directions/distances then can be used when coding tool X is used may be a subset of GEO directions/distances then can be used when coding tool X is not used.
25. Whether to/how to apply the filtering process may depend on the usage of GEO.
- a) In one example, the value of boundary filtering strength (e.g., bS) during the deblocking process may be dependent on whether the block is coded with GEO.
 - b) In one example, if the block edge is a transform block edge and the sample p0 or q0 is in a coding block with MergeGeoFlag equal to 1, the value of bS may be set equal to T (such as T=2).
 - c) In one example, the value of deblocking edges (e.g., edgeFlags) within a GEO block may be never equal to 2.
 - i. In one example, the value of deblocking edges (e.g., edgeFlags) within a GEO block may be equal to 2.
 - ii. In one example, given a block edge, if the sample p0 or q0 is in a coding block with MergeGeoFlag equal to 1, the value of bS may be dependent on the motion vectors, and/or reference pictures.

[0107] Additional Embodiments

[0108] Below are example embodiments, which can be applied to VVC specification. The modifications are based on the CE anchor of GEO working draft (JVET-P0884_P0885_WD (on_top_of_JVET-O2001-vE)_r2). Newly added parts are highlighted in bolded underlined text, and the deleted parts from VVC working draft are marked with double brackets(e.g., *[[a]]* denotes the deletion of the character “a”).

[0109] 5.1. An example embodiment #1: GEO mode constraint 1

7.3.8.7 Merge data syntax

merge_data(x0, y0, cbWidth, cbHeight, chType) {	Descriptor
if (CuPredMode[chType][x0][y0] == MODE_IBC) {	–
if(MaxNumIbcMergeCand > 1)	–

merge_idx [x0][y0]	– ae(v)
} else {	–
if(MaxNumSubblockMergeCand > 0 && cbWidth >= 8 && cbHeight >= 8)	–
merge_subblock_flag [x0][y0]	– ae(v)
if(merge_subblock_flag[x0][y0] == 1) {	–
if(MaxNumSubblockMergeCand > 1)	–
merge_subblock_idx [x0][y0]	– ae(v)
} else {	–
if((cbWidth * cbHeight) >= 64 && (sps_ciip_enabled_flag && cu_skip_flag[x0][y0] == 0 && cbWidth < 128 && cbHeight < 128) (sps_wedge_enabled_flag && MaxNumWedgeMergeCand > 1 && cbWidth >= 8 && cbHeight >= 8 && slice_type == B && <u>cbWidth * cbHeight <= 2048 && cbHeight / cbWidth <= 4 && cbWidth / cbHeight <= 2</u>)))	–
regular_merge_flag [x0][y0]	– ae(v)
if (regular_merge_flag[x0][y0] == 1) {	–
if(sps_mmvd_enabled_flag)	–
mmvd_merge_flag [x0][y0]	– ae(v)
if(mmvd_merge_flag[x0][y0] == 1) {	–
if(MaxNumMergeCand > 1)	–
mmvd_cand_flag [x0][y0]	– ae(v)
mmvd_distance_idx [x0][y0]	– ae(v)
mmvd_direction_idx [x0][y0]	– ae(v)
} else {	–
if(MaxNumMergeCand > 1)	–
merge_idx [x0][y0]	– ae(v)
}	–
} else {	–
if(sps_ciip_enabled_flag && sps_wedge_enabled_flag && MaxNumWedgeMergeCand > 1 && slice_type == B && cu_skip_flag[x0][y0] == 0 && cbWidth >= 8 && cbHeight >= 8 && cbWidth < 128 && cbHeight < 128 && <u>cbWidth * cbHeight <= 2048 && cbHeight / cbWidth <= 4 && cbWidth / cbHeight <= 2</u>)	–
{	
ciip_flag [x0][y0]	– ae(v)
if(ciip_flag[x0][y0] && MaxNumMergeCand > 1)	–

merge_idx [x0][y0]	– ae(v)
if(!ciip_flag[x0][y0] && MaxNumWedgeMergeCand > 1) {	–
wedge_partition_idx [x0][y0]	– ae(v)
merge_wedge_idx0 [x0][y0]	– ae(v)
merge_wedge_idx1 [x0][y0]	– ae(v)
}	–
}	–
}	–
}	–
}	

5.2. An example embodiment #2: GEO mode constraint 2

7.3.8.7 Merge data syntax

merge_data(x0, y0, cbWidth, cbHeight, chType) {	Descriptor
if (CuPredMode[chType][x0][y0] == MODE_IBC) {	–
if(MaxNumIbcMergeCand > 1)	–
merge_idx [x0][y0]	– ae(v)
} else {	–
if(MaxNumSubblockMergeCand > 0 && cbWidth >= 8 && cbHeight >= 8)	–
merge_subblock_flag [x0][y0]	– ae(v)
if(merge_subblock_flag[x0][y0] == 1) {	–
if(MaxNumSubblockMergeCand > 1)	–
merge_subblock_idx [x0][y0]	– ae(v)
} else {	–
if((cbWidth * cbHeight) >= 64 && (sps_ciip_enabled_flag && cu_skip_flag[x0][y0] == 0 && cbWidth < 128 && cbHeight < 128) (sps_wedge_enabled_flag && MaxNumWedgeMergeCand > 1 && cbWidth >= 8 && cbHeight >= 8 && slice_type == B && (<u>cbWidth * cbHeight <= 512 cbWidth * cbHeight <= 2048 && Abs(Log2(cbWidth) – Log2(cbHeight)) <= 1</u>)))	–
regular_merge_flag [x0][y0]	– ae(v)
if (regular_merge_flag[x0][y0] == 1) {	–
if(sps_mmvd_enabled_flag)	–
mmvd_merge_flag [x0][y0]	– ae(v)
if(mmvd_merge_flag[x0][y0] == 1) {	–
if(MaxNumMergeCand > 1)	–

mmvd_cand_flag[x0][y0]	– ae(v)
mmvd_distance_idx[x0][y0]	– ae(v)
mmvd_direction_idx[x0][y0]	– ae(v)
} else {	–
if(MaxNumMergeCand > 1)	–
merge_idx[x0][y0]	– ae(v)
}	–
} else {	–
if(sps_ciip_enabled_flag && sps_wedge_enabled_flag && MaxNumWedgeMergeCand > 1 && slice_type == B && cu_skip_flag[x0][y0] == 0 && cbWidth >= 8 && cbHeight >= 8 && cbWidth < 128 && cbHeight < 128 <u>&&</u> <u>(cbWidth * cbHeight <= 512 cbWidth * cbHeight <= 2048 &&</u> <u>Abs(Log2(cbWidth) – Log2(cbHeight)) <= 1)</u>) {	–
ciip_flag[x0][y0]	– ae(v)
if(ciip_flag[x0][y0] && MaxNumMergeCand > 1)	–
merge_idx[x0][y0]	– ae(v)
if(!ciip_flag[x0][y0] && MaxNumWedgeMergeCand > 1) {	–
wedge_partition_idx[x0][y0]	– ae(v)
merge_wedge_idx0[x0][y0]	– ae(v)
merge_wedge_idx1[x0][y0]	– ae(v)
}	–
}	–
}	–
}	–
}	

5.3. An example embodiment #3: block size dependent GEO mode selection 1

8.5.7 Decoding process for wedge inter blocks

8.5.7.1 General

[0110] This process is invoked when decoding a coding unit with

wedge_merge_mode[xCb][yCb] equal to 1.

[0111] Inputs to this process are:

[0112] a luma location (xCb, yCb) specifying the top-left sample of the current coding block relative to the top-left luma sample of the current picture,

- a variable `cbWidth` specifying the width of the current coding block in luma samples,
- a variable `cbHeight` specifying the height of the current coding block in luma samples,
- the luma motion vectors in 1/16 fractional-sample accuracy `mvA` and `mvB`,
- the chroma motion vectors `mvCA` and `mvCB`,
- the reference indices `refIdxA` and `refIdxB`,
- the prediction list flags `predListFlagA` and `predListFlagB`.

[0113] Outputs of this process are:

- an $(cbWidth) \times (cbHeight)$ array `predSamplesL` of luma prediction samples,
- an $(cbWidth / SubWidthC) \times (cbHeight / SubHeightC)$ array `predSamplesCb` of chroma prediction samples for the component `Cb`,
- an $(cbWidth / SubWidthC) \times (cbHeight / SubHeightC)$ array `predSamplesCr` of chroma prediction samples for the component `Cr`.

[0114] Let `predSamplesLAL` and `predSamplesLBL` be $(cbWidth) \times (cbHeight)$ arrays of predicted luma sample values and, `predSamplesLACb`, `predSamplesLBCb`, `predSamplesLACr` and `predSamplesLBCr` be $(cbWidth / SubWidthC) \times (cbHeight / SubHeightC)$ arrays of predicted chroma sample values.

The `predSamplesL`, `predSamplesCb` and `predSamplesCr` are derived by the following ordered steps:

1. For `N` being each of `A` and `B`, the following applies:
 - The reference picture including an ordered two-dimensional array `refPicLNL` of luma samples and two ordered two-dimensional arrays `refPicLNCb` and `refPicLNCr` of chroma samples is derived by invoking the process specified in clause 8.5.6.2 with `X` set equal to `predListFlagN` and `refIdxX` set equal to `refIdxN` as input.
 - The array `predSamplesLNL` is derived by invoking the fractional sample interpolation process specified in clause 8.5.6.3 with the luma location (xCb, yCb) , the luma coding block width `sbWidth` set equal to `cbWidth`, the luma coding block height `sbHeight` set equal to `cbHeight`, the motion vector offset `mvOffset` set equal to $(0, 0)$, the motion vector `mvLX` set equal to `mvN` and the reference array `refPicLXL` set equal to `refPicLNL`, the variable `bdofFlag` set equal to `FALSE`, and the variable `cIdx` is set equal to 0 as inputs.
 - The array `predSamplesLNCb` is derived by invoking the fractional sample interpolation process specified in clause 8.5.6.3 with the luma location (xCb, yCb) , the coding block

width $sbWidth$ set equal to $cbWidth / SubWidthC$, the coding block height $sbHeight$ set equal to $cbHeight / SubHeightC$, the motion vector offset $mvOffset$ set equal to $(0, 0)$, the motion vector $mvLX$ set equal to $mvCN$, and the reference array $refPicLXC_b$ set equal to $refPicLNC_b$, the variable $bdofFlag$ set equal to $FALSE$, and the variable $cIdx$ is set equal to 1 as inputs.

- The array $predSamplesLNC_r$ is derived by invoking the fractional sample interpolation process specified in clause 8.5.6.3 with the luma location (xCb, yCb) , the coding block width $sbWidth$ set equal to $cbWidth / SubWidthC$, the coding block height $sbHeight$ set equal to $cbHeight / SubHeightC$, the motion vector offset $mvOffset$ set equal to $(0, 0)$, the motion vector $mvLX$ set equal to $mvCN$, and the reference array $refPicLXC_r$ set equal to $refPicLNC_r$, the variable $bdofFlag$ set equal to $FALSE$, and the variable $cIdx$ is set equal to 2 as inputs.

2. **The value of wedge_partition_idx'[xCb][yCb] are set according to the value of wedge_partition_idx[xCb][yCb] and the coding block width cbWidth and the coding block height cbHeight, as specified in Table 8-xx and Table 8-xxx.**
3. The partition angle and distance of the wedge merge mode $angleIdx$ and $distanceIdx$ are set according to the value of $wedge_partition_idx'[xCb][yCb]$ as specified in Table 8-10
4. The prediction samples inside the current luma coding block, $predSamples_L[x_L][y_L]$ with $x_L = 0..cbWidth - 1$ and $y_L = 0..cbHeight - 1$, are derived by invoking the weighted sample prediction process for wedge merge mode specified in clause 8.5.7.2 with the coding block width $nCbW$ set equal to $cbWidth$, the coding block height $nCbH$ set equal to $cbHeight$, the sample arrays $predSamplesLA_L$ and $predSamplesLB_L$, and the variables $angleIdx$ and $distanceIdx$, and $cIdx$ equal to 0 as inputs.
5. The prediction samples inside the current chroma component Cb coding block, $predSamplesCb[x_C][y_C]$ with $x_C = 0..cbWidth / SubWidthC - 1$ and $y_C = 0..cbHeight / SubHeightC - 1$, are derived by invoking the weighted sample prediction process for wedge merge mode specified in clause 8.5.7.2 with the coding block width $nCbW$ set equal to $cbWidth / SubWidthC$, the coding block height $nCbH$ set equal to $cbHeight / SubHeightC$, the sample arrays $predSamplesLAC_b$ and $predSamplesLBC_b$, and the variables $angleIdx$ and $distanceIdx$, and $cIdx$ equal to 1 as inputs.

6. The prediction samples inside the current chroma component Cr coding block, $\text{predSamples}_{Cr}[x_C][y_C]$ with $x_C = 0..cbWidth / \text{SubWidthC} - 1$ and $y_C = 0..cbHeight / \text{SubHeightC} - 1$, are derived by invoking the weighted sample prediction process for wedge merge mode specified in clause 8.5.7.2 with the coding block width $nCbW$ set equal to $cbWidth / \text{SubWidthC}$, the coding block height $nCbH$ set equal to $cbHeight / \text{SubHeightC}$, the sample arrays $\text{predSamplesLA}_{Cr}$ and $\text{predSamplesLB}_{Cr}$, and the variables angleIdx and distanceIdx , and $cIdx$ equal to 2 as inputs.
7. The motion vector storing process for merge wedge mode specified in clause 8.5.7.3 is invoked with the luma coding block location (xCb, yCb) , the luma coding block width $cbWidth$, the luma coding block height $cbHeight$, the partition direction angleIdx and distanceIdx , the luma motion vectors mvA and mvB , the reference indices refIdxA and refIdxB , and the prediction list flags predListFlagA and predListFlagB as inputs.

Table 8-xx – Mapping table of the MaxGEOMode based on the block width and block height.
 (comment: below table is an example for the case that the maximum number of GEO modes allowed for a GEO block with block category (from 0...k) is equal to $\text{MaxGEOMode}[i]$ ($i=0...k$), and the block category may be decided by the block width and block height)

Block Category	0	1	2	...	k
MaxGEOMode	MaxGEOMode[0]	MaxGEOMode[1]	MaxGEOMode[2]	...	MaxGEOMode[k]
	j	j			

Table 8-xxx – Mapping table of the wedge partition_idx' values based on the wedge partition_idx value.

Block Category	wedge partition_idx	0	1	2	3	4	...	MaxGEOMode[0]-1
0	wedge partition_idx'	x	y	z	u	v	...	w
Block Category 1	wedge partition_idx	0	1	2	3	4	...	MaxGEOMode[1]-1
	wedge partition_idx'	xx	yy	zz	uu	vv	...	ww
...	...	...	...	...	...	...	...	...
	...	...	...	...	...	...	...	...
Block Category k	wedge partition_idx	0	1	2	3	4	...	MaxGEOMode[k]-1
	wedge partition_idx'	xxx	yyy	zzz	uuu	xxx	...	www

FIG. 21 shows a corresponding Table 8-10 indicating specification of the angleIdx and distanceIdx values based on the $\text{wedge_partition_idx'}$ value.

Table 9-77 – Syntax elements and associated binarizations

Syntax structure	Syntax element	Binarization	
		Process	Input parameters
merge_data()	regular_merge_flag[][]	FL	cMax = 1
	mmvd_merge_flag[][]	FL	cMax = 1
	mmvd_cand_flag[][]	FL	cMax = 1
	mmvd_distance_idx[][]	TR	cMax = 7, cRiceParam = 0
	mmvd_direction_idx[][]	FL	cMax = 3
	ciip_flag[][]	FL	cMax = 1
	merge_subblock_flag[][]	FL	cMax = 1
	merge_subblock_idx[][]	TR	cMax = MaxNumSubblockMergeCand - 1, cRiceParam = 0
	wedge_partition_idx[][]	TB	cMax = [[82]] <u>MaxGEOModel[i]</u> (comment: here <u>MaxGEOModel[i]</u> means the number of GEO modes allowed for a GEO block with block category i, wherein i may be decided by the block block width and block height)
	merge_wedge_idx0[][]	TR	cMax = MaxNumWedgeMergeCand - 1, cRiceParam = 0
	merge_wedge_idx1[][]	TR	cMax = MaxNumWedgeMergeCand - 2, cRiceParam = 0
	merge_idx[][]	TR	cMax = MaxNumMergeCand - 1, cRiceParam = 0

5.4. An example embodiment #4: block size dependent GEO mode selection 2

8.5.7 Decoding process for wedge inter blocks

8.5.7.1 General

This process is invoked when decoding a coding unit with wedge_merge_mode[xCb][yCb] equal to 1.

Inputs to this process are:

- a luma location (xCb, yCb) specifying the top-left sample of the current coding block relative to the top-left luma sample of the current picture,
- a variable cbWidth specifying the width of the current coding block in luma samples,
- a variable cbHeight specifying the height of the current coding block in luma samples,
- the luma motion vectors in 1/16 fractional-sample accuracy mvA and mvB,
- the chroma motion vectors mvCA and mvCB,
- the reference indices refIdxA and refIdxB,

- the prediction list flags `predListFlagA` and `predListFlagB`.

Outputs of this process are:

- an $(cbWidth) \times (cbHeight)$ array `predSamplesL` of luma prediction samples,
- an $(cbWidth / SubWidthC) \times (cbHeight / SubHeightC)$ array `predSamplesCb` of chroma prediction samples for the component `Cb`,
- an $(cbWidth / SubWidthC) \times (cbHeight / SubHeightC)$ array `predSamplesCr` of chroma prediction samples for the component `Cr`.

Let `predSamplesLAL` and `predSamplesLBL` be $(cbWidth) \times (cbHeight)$ arrays of predicted luma sample values and, `predSamplesLACb`, `predSamplesLBCb`, `predSamplesLACr` and `predSamplesLBCr` be $(cbWidth / SubWidthC) \times (cbHeight / SubHeightC)$ arrays of predicted chroma sample values.

The `predSamplesL`, `predSamplesCb` and `predSamplesCr` are derived by the following ordered steps:

8. For `N` being each of `A` and `B`, the following applies:
 - The reference picture including an ordered two-dimensional array `refPicLNL` of luma samples and two ordered two-dimensional arrays `refPicLNCb` and `refPicLNCr` of chroma samples is derived by invoking the process specified in clause 8.5.6.2 with `X` set equal to `predListFlagN` and `refIdxX` set equal to `refIdxN` as input.
 - The array `predSamplesLNL` is derived by invoking the fractional sample interpolation process specified in clause 8.5.6.3 with the luma location (xCb, yCb) , the luma coding block width `sbWidth` set equal to `cbWidth`, the luma coding block height `sbHeight` set equal to `cbHeight`, the motion vector offset `mvOffset` set equal to $(0, 0)$, the motion vector `mvLX` set equal to `mvN` and the reference array `refPicLXL` set equal to `refPicLNL`, the variable `bdofFlag` set equal to `FALSE`, and the variable `cIdx` is set equal to 0 as inputs.
 - The array `predSamplesLNCb` is derived by invoking the fractional sample interpolation process specified in clause 8.5.6.3 with the luma location (xCb, yCb) , the coding block width `sbWidth` set equal to $cbWidth / SubWidthC$, the coding block height `sbHeight` set equal to $cbHeight / SubHeightC$, the motion vector offset `mvOffset` set equal to $(0, 0)$, the motion vector `mvLX` set equal to `mvCN`, and the reference array `refPicLXCb` set equal to `refPicLNCb`, the variable `bdofFlag` set equal to `FALSE`, and the variable `cIdx` is set equal to 1 as inputs.

- The array $\text{predSamplesLN}_{Cr}$ is derived by invoking the fractional sample interpolation process specified in clause 8.5.6.3 with the luma location (x_{Cb}, y_{Cb}) , the coding block width $sbWidth$ set equal to $cbWidth / SubWidthC$, the coding block height $sbHeight$ set equal to $cbHeight / SubHeightC$, the motion vector offset $mvOffset$ set equal to $(0, 0)$, the motion vector $mvLX$ set equal to $mvCN$, and the reference array refPicLX_{Cr} set equal to refPicLN_{Cr} , the variable $bdofFlag$ set equal to `FALSE`, and the variable $cIdx$ is set equal to 2 as inputs.

9. The value of $\text{wedge_partition_idx}'[x_{Cb}][y_{Cb}]$ are set according to the value of $\text{wedge_partition_idx}[x_{Cb}][y_{Cb}]$ and the coding block width $cbWidth$ and the coding block height $cbHeight$, as specified in Table 8-xx.

10. The partition angle and distance of the wedge merge mode angleIdx and distanceIdx are set according to the value of $\text{wedge_partition_idx}'[x_{Cb}][y_{Cb}]$ as specified in Table 8-10
11. The prediction samples inside the current luma coding block, $\text{predSamples}_L[x_L][y_L]$ with $x_L = 0..cbWidth - 1$ and $y_L = 0..cbHeight - 1$, are derived by invoking the weighted sample prediction process for wedge merge mode specified in clause 8.5.7.2 with the coding block width $nCbW$ set equal to $cbWidth$, the coding block height $nCbH$ set equal to $cbHeight$, the sample arrays predSamplesLA_L and predSamplesLB_L , and the variables angleIdx and distanceIdx , and $cIdx$ equal to 0 as inputs.
12. The prediction samples inside the current chroma component Cb coding block, $\text{predSamples}_{Cb}[x_C][y_C]$ with $x_C = 0..cbWidth / SubWidthC - 1$ and $y_C = 0..cbHeight / SubHeightC - 1$, are derived by invoking the weighted sample prediction process for wedge merge mode specified in clause 8.5.7.2 with the coding block width $nCbW$ set equal to $cbWidth / SubWidthC$, the coding block height $nCbH$ set equal to $cbHeight / SubHeightC$, the sample arrays $\text{predSamplesLA}_{Cb}$ and $\text{predSamplesLB}_{Cb}$, and the variables angleIdx and distanceIdx , and $cIdx$ equal to 1 as inputs.
13. The prediction samples inside the current chroma component Cr coding block, $\text{predSamples}_{Cr}[x_C][y_C]$ with $x_C = 0..cbWidth / SubWidthC - 1$ and $y_C = 0..cbHeight / SubHeightC - 1$, are derived by invoking the weighted sample prediction process for wedge merge mode specified in clause 8.5.7.2 with the coding block width $nCbW$ set equal to $cbWidth / SubWidthC$, the coding block height $nCbH$ set equal to

cbHeight / SubHeightC, the sample arrays predSamplesLA_{Cr} and predSamplesLB_{Cr}, and the variables angleIdx and distanceIdx, and cIdx equal to 2 as inputs.

14. The motion vector storing process for merge wedge mode specified in clause 8.5.7.3 is invoked with the luma coding block location (xCb, yCb), the luma coding block width cbWidth, the luma coding block height cbHeight, the partition direction angleIdx and distanceIdx, the luma motion vectors mvA and mvB, the reference indices refIdxA and refIdxB, and the prediction list flags predListFlagA and predListFlagB as inputs.

Table 8-xx – Mapping table of the wedge_partition_idx' values based on the wedge_partition_idx value.

(comment: below table is an example for the case that the maximum number of GEO modes allowed for a GEO block with block category (from 0...k) is equal to a constant Max)

<u>Block Category</u>	<u>wedge_partition_idx</u>	<u>0</u>	<u>1</u>	<u>2</u>	<u>3</u>	<u>4</u>	<u>...</u>	<u>Max - 1</u>
<u>0</u>	<u>wedge_partition_idx'</u>	<u>x</u>	<u>y</u>	<u>z</u>	<u>u</u>	<u>v</u>	<u>...</u>	<u>w</u>
<u>Block Category 1</u>	<u>wedge_partition_idx</u>	<u>0</u>	<u>1</u>	<u>2</u>	<u>3</u>	<u>4</u>	<u>...</u>	<u>Max - 1</u>
	<u>wedge_partition_idx'</u>	<u>xx</u>	<u>yy</u>	<u>zz</u>	<u>uu</u>	<u>vv</u>	<u>...</u>	<u>ww</u>
<u>...</u>	<u>...</u>	<u>...</u>	<u>...</u>	<u>...</u>	<u>...</u>	<u>...</u>	<u>...</u>	<u>...</u>
<u>...</u>	<u>...</u>	<u>...</u>	<u>...</u>	<u>...</u>	<u>...</u>	<u>...</u>	<u>...</u>	<u>...</u>
<u>Block Category</u>	<u>wedge_partition_idx</u>	<u>0</u>	<u>1</u>	<u>2</u>	<u>3</u>	<u>4</u>	<u>...</u>	<u>Max - 1</u>
<u>k</u>	<u>wedge_partition_idx'</u>	<u>xxx</u>	<u>yyy</u>	<u>zzz</u>	<u>uuu</u>	<u>xxx</u>	<u>...</u>	<u>www</u>

FIG. 22 shows a corresponding Table 8-10 indicating specification of the angleIdx and distanceIdx values based on the wedge_partition_idx' value.

Table 9-77 – Syntax elements and associated binarizations

Syntax structure	Syntax element	Binarization	
		Process	Input parameters
merge_data()	regular_merge_flag[][]	FL	cMax = 1
	mmvd_merge_flag[][]	FL	cMax = 1
	mmvd_cand_flag[][]	FL	cMax = 1
	mmvd_distance_idx[][]	TR	cMax = 7, cRiceParam = 0
	mmvd_direction_idx[][]	FL	cMax = 3
	ciip_flag[][]	FL	cMax = 1
	merge_subblock_flag[][]	FL	cMax = 1
	merge_subblock_idx[][]	TR	cMax = MaxNumSubblockMergeCand - 1, cRiceParam = 0
	wedge_partition_idx[][]	TB	cMax = [[82]] <u>Max (comment: here Max means the number of GEO modes allowed for a GEO block, which is a constant)</u>
	merge_wedge_idx0[][]	TR	cMax = MaxNumWedgeMergeCand - 1, cRiceParam = 0
	merge_wedge_idx1[][]	TR	cMax = MaxNumWedgeMergeCand - 2, cRiceParam = 0
	merge_idx[][]	TR	cMax = MaxNumMergeCand - 1, cRiceParam = 0

5.5. An example embodiment #5: 64 GEO modes with 20 angles supported

In the below example, NUM_ANGLE, T1 and T2 may be constant values, and Table 8-10 and Table 8-12 are changed accordingly.

FIG. 23A shows an old version of Table 8-10 indicating specification of the angleIdx and distanceIdx values based on the wedge_partition_idx value, which is now deleted from the relevant working draft. FIG. 23B shows an example of Table 8-10 indicating specification of the angleIdx and distanceIdx values based on the wedge_partition_idx value. Table 8-10 as shown in FIG. 23B is an example of 64 modes that in case of NUM_ANGLE = 20, and only 2 distances allowed for angleIdx = 0/5/10/15).

In the descriptions below, newly added parts are highlighted in bolded underlined text, and the deleted parts from the relevant working draft are marked with double brackets (e.g., ~~[[a]]~~ denotes the deletion of the character “a”).

8.5.7.2 Weighted sample prediction process for wedge merge mode

Inputs to this process are:

- two variables nCbW and nCbH specifying the width and the height of the current coding block,
- two (nCbW)x(nCbH) arrays predSamplesLA and predSamplesLB,
- a variable angleIdx specifying the angle index of the wedge partition ,
- a variable distanceIdx specifying the distance idx of the wedge partition,
- a variable cIdx specifying colour component index.

Output of this process is the (nCbW)x(nCbH) array pbSamples of prediction sample values.

The variable bitDepth is derived as follows:

- If cIdx is equal to 0, bitDepth is set equal to BitDepth_Y.
- If cIdx is equal to 0, nW and nH are set equal to nCbW and nCbH respectively, otherwise (cIdx is not equal to 0) nW and nH are set equal to nCbW x SubWidthC and nCbH x SubHeightC respectively.
- If cIdx is equal to 0, subW and subH are both set 1, otherwise (cIdx is not equal to 0) subW and subH are set equal to SubWidthC and SubHeightC respectively.
- Otherwise, bitDepth is set equal to BitDepth_C.

Variables shift1 and offset1 are derived as follows:

- The variable shift1 is set equal to Max(5, 17 – bitDepth).
- The variable offset1 is set equal to 1 << (shift1 – 1).

The values of the following variables are set:

- hwRatio is set to nH / nW
- displacementX is set to angleIdx
- displacementY is set to (displacementX + ~~[[6]](NUM_ANGLE>2))%[[24]]NUM_ANGLE~~
- If angleIdx >[[=10]]**T1** && angleIdx <[[=20]]**T2**, PART1 and PART2 are set equal to A and B respectively, otherwise PART1 and PART2 are set equal to B and A respectively.
- rho is set to the following value using the look-up tables denoted as Dis, specified in Table 8-12:

$$\rho = (Dis[displacementX] < 8) + (Dis[displacementY] < 8)$$

If one of the following conditions is true, variable shiftHor is set equal to 0:

$$angleIdx \% [[12]](\underline{NUM_ANGLE/2}) \text{ is equal to } [[6]](\underline{NUM_ANGLE > 2})$$

$$angleIdx \% [[12]](\underline{NUM_ANGLE/2}) \text{ is not equal to 0 and } hwRatio \geq 1$$

Otherwise, shiftHor is set equal to 1.

If shiftHor is equal to 0, offsetX and offsetY are derived as follows:

$$offsetX = (256 - nW) >> 1$$

$$offsetY = (256 - nH) >> 1 + angleIdx < [[12]](\underline{NUM_ANGLE/2}) ? (distanceIdx * nH) >> 3 : -((distanceIdx * nH) >> 3)$$

Otherwise, if shiftHor is equal to 1, offsetX and offsetY are derived as follows:

$$offsetX = (256 - nW) >> 1 + angleIdx < [[12]](\underline{NUM_ANGLE/2}) ? (distanceIdx * nW) >> 3 : -((distanceIdx * nW) >> 3)$$

$$offsetY = (256 - nH) >> 1$$

The prediction sample values pbSamples[x][y] with $x = 0..nCbw - 1$ and $y = 0..nCbh - 1$ is set according the following ordered steps:

- The variable weightIdx and weightIdxAbs are calculated using the look-up table Table 8-12 as follows:

$$\begin{aligned} weightIdx &= (((x * subW + offsetX) < 1) + 1) * Dis[displacementX] \\ &\quad + (((y * subH + offsetY) < 1) + 1) * Dis[displacementY] - \rho. \\ weightIdxAbs &= Clip3(0, 26, abs(weightIdx)). \end{aligned}$$

- The value of sampleWeight is derived according to according to Table 8-13 as follows:

$$sampleWeight = weightIdx \leq 0 ? WedgeFilter[weightIdxAbs] : 8 - WedgeFilter[weightIdxAbs]$$

NOTE – The value of sample sampleWeight_L[x][y] can also be derived from sampleWeight_L[x -shiftX][y -shiftY]. If the angleIdx is larger than 4 and smaller than 12, or angleIdx is larger than 20 and smaller than 24, shiftX is the tangent of the split angle and shiftY is 1, otherwise shiftX is 1 of the split angle and shiftY is cotangent of the split angle. If tangent (resp. cotangent) value is infinity, shiftX is 1 (resp. 0) or shift Y is 0 (reps. 1).

- The prediction sample value $pbSamples[x][y]$ is derived as follows :

$$pbSamples[x][y] = \text{Clip3}(0, (1 \ll \text{bitDepth}) - 1, (\text{predSamplesLPART1}[x][y] * (8 - \text{sampleWeight}) + \text{predSamplesLPART2}[x][y] * \text{sampleWeight} + \text{offset1}) \gg \text{shift1})$$

Table 8-12 - Look-up table *Dis* for derivation of wedgemetric partitioning distance.

The table below is now deleted from the relevant working draft.

idx	0	1	2	3	4	5	6	7	8	9	10	11
Dis[idx]	8	8	8	8	4	2	0	-2	-4	-8	-8	-8
idx	12	13	14	15	16	17	18	19	20	21	22	23
Dis[idx]	-8	-8	-8	-8	-4	-2	0	2	4	8	8	8

The examples of the table changed in the relevant working draft are shown below.

- In one example, $T1 = 8$ and $T2 = 16$ in case of $\text{NUM_ANGLE} = 20$, and Table 8-12 is set below.

idx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Dis[idx]	8	8	8	4	2	-2	-4	-8	-8	-8	-8	-8	-8	-4	-2	2	4	8	8	8

- In one example, $T1 = 8$ and $T2 = 17$ in case of $\text{NUM_ANGLE} = 20$, and Table 8-12 is set below.

idx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Dis[idx]	8	8	8	4	2	0	-2	-4	-8	-8	-8	-8	-8	-4	-2	0	2	4	8	8

- In one example, $T1 = 9$ and $T2 = 16$ in case of $\text{NUM_ANGLE} = 20$, and Table 8-12 is set below.

idx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Dis[idx]	8	8	8	8	2	0	-2	-8	-8	-8	-8	-8	-8	-8	-2	0	2	8	8	8

- In one example, $T1 = 9$ and $T2 = 16$ in case of $\text{NUM_ANGLE} = 20$, and Table 8-12 is set below.

idx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Dis[idx]	8	8	8	8	4	0	-4	-8	-8	-8	-8	-8	-8	-8	-4	0	4	8	8	8

- In one example, $T1 = 7$ and $T2 = 13$ in case of $\text{NUM_ANGLE} = 16$, and Table 8-12 is set below.

idx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Dis[idx]	8	8	4	2	0	-2	-4	-8	-8	-8	-4	-2	0	2	4	8

Table 8-13 - Filter weight look-up table *WedgeFilter* for derivation of wedge partitioning filter weights.

idx	0	1	2	3	4	5	6	7	8	9	10	11	12	13
WedgeFilter [idx]	4	4	4	4	5	5	5	5	5	5	5	6	6	6
idx	14	15	16	17	18	19	20	21	22	23	24	25	26	

WedgeFilter [idx]	6	6	6	6	7	7	7	7	7	7	7	7	8	
------------------------------------	---	---	---	---	---	---	---	---	---	---	---	---	---	--

8.5.7.3 Motion vector storing process for wedge merge mode

This process is invoked when decoding a coding unit with MergeWedgeFlag[xCb][yCb] equal to 1.

Inputs to this process are:

- a luma location (xCb, yCb) specifying the top-left sample of the current coding block relative to the top-left luma sample of the current picture,
- a variable cbWidth specifying the width of the current coding block in luma samples,
- a variable cbHeight specifying the height of the current coding block in luma samples,
- the luma motion vectors in 1/16 fractional-sample accuracy mvA and mvB,
- the reference indices refIdxA and refIdxB,
- the prediction list flags predListFlagA and predListFlagB.

The variables numSbX and numSbY specifying the number of 4x4 blocks in the current coding block in horizontal and vertical direction are set equal to numSbX = cbWidth >> 2 and numSbY = cbHeight >> 2.

The value of the following variables are set:

- displacementX is set to angleIdx, displacementY is set to (displacementX + $[[6]](\text{NUM_ANGLE} > 2) \% [[24]] \text{NUM_ANGLE}$)
- hwRatio is set equal to nCbH / nCbW

If one of the following conditions is true, variable shiftHor is set equal to 0:

angleIdx % $[[12]](\text{NUM_ANGLE}/2)$ is equal to $[[8]](\text{NUM_ANGLE} > 2)$
angleIdx % $[[12]](\text{NUM_ANGLE}/2)$ is not equal to 0 and hwRatio ≥ 1

Otherwise, shiftHor is set equal to 1.

partIdx is set to angleIdx > $[[=10]]T1$ && angleIdx < $[[=20]]T2$? 1 : 0.

If shiftHor is equal to 0, offsetX and offsetY are derived as follows:

- offsetX = (64 – numSbX) >> 1
- offsetY = (64 – numSbY) >> 1 + angleIdx < $[[12]](\text{NUM_ANGLE}/2)$? (distanceIdx * nCbH) >> 5 : -((distanceIdx * nCbH) >> 5)

Otherwise, if shiftHor is equal to 1, offsetX and offsetY are derived as follows:

- $\text{offsetX} = (64 - \text{numSbX}) \gg 1 + \text{angleIdx} < [[12]](\underline{\text{NUM_ANGLE/2}})? (\text{distanceIdx} * \text{nCbW}) \gg 5 : -((\text{distanceIdx} * \text{nCbW}) \gg 5)$
- $\text{offsetY} = (64 - \text{numSbY}) \gg 1$

The value of the variable rho is derived according to the equation below and according to the Dis lookup table specified in Table 8-12:

- $\text{rho} = (\text{Dis}[\text{displacementX}] < 8) + (\text{Dis}[\text{displacementY}] < 8) .$

motionOffset is set equal to the following value using the look-up tables denoted as Dis, specified in Table 8-11 and Table 8-12:

- $\text{motionOffset} = 3 * \text{Dis}[\text{displacementX}] + 3 * \text{Dis}[\text{displacementY}] .$

For each 4x4 subblock at subblock index (xSbIdx, ySbIdx) with xSbIdx = 0..numSbX – 1, and ySbIdx = 0..numSbY – 1, the following applies:

The variable motionIdx is calculated using the look-up table Table 8-12 as following:

- $\text{motionIdx} = (((\text{xSbIdx} + \text{offsetX}) < 3) + 1) * \text{Dis}[\text{displacementX}] + (((\text{xSbIdx} + \text{offsetY}) < 3) + 1) * \text{Dis}[\text{displacementY}] - \text{rho} + \text{motionOffset}$

The variable sType is derived as follows:

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

Below are example embodiments, which can be applied to VVC specification. The modifications are based on the CE anchor of GEO working draft (JVET-P0884_P0885_WD (on_top_of_JVET-O2001-vE)_r2). Newly added parts are highlighted in bolded underlined text, and the deleted parts

from VVC working draft are marked with double brackets (e.g., $[[a]]$ denotes the deletion of the character “a”).

An example embodiment: GEO mode constraint

7.3.9.7 Merge data syntax

merge_data(x0, y0, cbWidth, cbHeight, chType) {	Descriptor
...	
if((cbWidth * cbHeight) >= 64 && cbWidth <= 64 && cbHeight <= 64 && ((sps_ciip_enabled_flag && cu_skip_flag[x0][y0] == 0 [&& cbWidth < 128 && cbHeight < 128]) (sps_geo_enabled_flag && MaxNumGeoMergeCand > 1 && cbWidth >= 8 && cbHeight >= 8 && slice_type == B)))	
regular_merge_flag[x0][y0]	ae(v)
if (regular_merge_flag[x0][y0] == 1){	
...	
} else {	
if(sps_ciip_enabled_flag && sps_geo_enabled_flag && MaxNumGeoMergeCand > 1 && slice_type == B && cu_skip_flag[x0][y0] == 0 && cbWidth >= 8 && cbHeight >= 8 && cbWidth < 128 && cbHeight < 128)	
ciip_flag[x0][y0]	ae(v)
...	

An example embodiment: GEO mode constraint

7.3.9.7 Merge data syntax

merge_data(x0, y0, cbWidth, cbHeight, chType) {	Descriptor
...	
if((cbWidth * cbHeight) >= 64 && ((sps_ciip_enabled_flag && cu_skip_flag[x0][y0] == 0 && cbWidth < 128 && cbHeight < 128) (sps_geo_enabled_flag && MaxNumGeoMergeCand > 1 && cbWidth >= 8 && cbHeight >= 8 && slice_type == B && cbWidth <= 32 && cbHeight <= 32))) regular_merge_flag[x0][y0]	ae(v)
if (regular_merge_flag[x0][y0] == 1){	
...	
} else {	
if(sps_ciip_enabled_flag && sps_geo_enabled_flag && MaxNumGeoMergeCand > 1 && slice_type == B && cu_skip_flag[x0][y0] == 0 && cbWidth >= 8 && cbHeight >= 8 && cbWidth <= 128 32 && cbHeight <= 128 32) ciip_flag[x0][y0]	ae(v)
...	

An example embodiment: GEO mode constraint

7.3.9.7 Merge data syntax

merge_data(x0, y0, cbWidth, cbHeight, chType) {	Descriptor
...	
if((cbWidth * cbHeight) >= 64 && cbWidth <= 64 && cbHeight <= 64 && ((sps_ciip_enabled_flag && cu_skip_flag[x0][y0] == 0 [&& cbWidth < 128 && cbHeight < 128]) (sps_geo_enabled_flag && MaxNumGeoMergeCand > 1 && cbWidth >= 8 && cbHeight >= 8 && slice_type == B && cbWidth/cbHeight <= 2 && cbHeight/cbWidth <= 4))) regular_merge_flag[x0][y0]	ae(v)
if (regular_merge_flag[x0][y0] == 1){	
...	
} else {	
if(sps_ciip_enabled_flag && sps_geo_enabled_flag && MaxNumGeoMergeCand > 1 && slice_type == B && cu_skip_flag[x0][y0] == 0 && cbWidth >= 8 && cbHeight >= 8 && cbWidth < 128 && cbHeight < 128 && cbWidth/cbHeight <= 2 && cbHeight/cbWidth <= 4) ciip_flag[x0][y0]	ae(v)

...	
-----	--

An example embodiment: GEO mode constraint

7.3.9.7 Merge data syntax

merge_data(x0, y0, cbWidth, cbHeight, chType) {	Descriptor
...	
if((cbWidth * cbHeight) >= 64 && cbWidth <= 64 && cbHeight <= 64 && ((sps_ciip_enabled_flag && cu_skip_flag[x0][y0] == 0 [&& cbWidth < 128 && cbHeight < 128]) (sps_geo_enabled_flag && MaxNumGeoMergeCand > 1 && cbWidth >= 8 && cbHeight >= 8 && slice_type == B && cbWidth/cbHeight <= 4 && cbHeight/cbWidth <= 4)))	
regular_merge_flag[x0][y0]	ae(v)
if (regular_merge_flag[x0][y0] == 1) {	
...	
} else {	
if(sps_ciip_enabled_flag && sps_geo_enabled_flag && MaxNumGeoMergeCand > 1 && slice_type == B && cu_skip_flag[x0][y0] == 0 && cbWidth >= 8 && cbHeight >= 8 && cbWidth < 128 && cbHeight < 128 && cbWidth/cbHeight <= 4 && cbHeight/cbWidth <= 4)	
ciip_flag[x0][y0]	ae(v)
...	

An example embodiment: block size dependent GEO mode selection

8.5.7 Decoding process for geo inter blocks

8.5.7.1 General

This process is invoked when decoding a coding unit with MergeGeoFlag[xCb][yCb] equal to 1.

Inputs to this process are:

- a luma location (xCb, yCb) specifying the top-left sample of the current coding block relative to the top-left luma sample of the current picture,
- a variable cbWidth specifying the width of the current coding block in luma samples,
- a variable cbHeight specifying the height of the current coding block in luma samples,
- the luma motion vectors in 1/16 fractional-sample accuracy mvA and mvB,
- the chroma motion vectors mvCA and mvCB,
- the reference indices refIdxA and refIdxB,
- the prediction list flags predListFlagA and predListFlagB.

Outputs of this process are:

- an $(cbWidth) \times (cbHeight)$ array $predSamples_L$ of luma prediction samples,
- an $(cbWidth / SubWidthC) \times (cbHeight / SubHeightC)$ array $predSamples_{Cb}$ of chroma prediction samples for the component Cb,
- an $(cbWidth / SubWidthC) \times (cbHeight / SubHeightC)$ array $predSamples_{Cr}$ of chroma prediction samples for the component Cr.

Let $predSamples_{LA_L}$ and $predSamples_{LB_L}$ be $(cbWidth) \times (cbHeight)$ arrays of predicted luma sample values and, $predSamples_{LA_{Cb}}$, $predSamples_{LB_{Cb}}$, $predSamples_{LA_{Cr}}$ and $predSamples_{LB_{Cr}}$ be $(cbWidth / SubWidthC) \times (cbHeight / SubHeightC)$ arrays of predicted chroma sample values.

The $predSamples_L$, $predSamples_{Cb}$ and $predSamples_{Cr}$ are derived by the following ordered steps:

1. For N being each of A and B, the following applies:
 - The reference picture consisting of an ordered two-dimensional array $refPicLN_L$ of luma samples and two ordered two-dimensional arrays $refPicLN_{Cb}$ and $refPicLN_{Cr}$ of chroma samples is derived by invoking the process specified in clause 8.5.6.2 with X set equal to $predListFlagN$ and $refIdxX$ set equal to $refIdxN$ as input.
 - The array $predSamplesLN_L$ is derived by invoking the fractional sample interpolation process specified in clause 8.5.6.3 with the luma location (x_{Cb}, y_{Cb}) , the luma coding block width $sbWidth$ set equal to $cbWidth$, the luma coding block height $sbHeight$ set equal to $cbHeight$, the motion vector offset $mvOffset$ set equal to $(0, 0)$, the motion vector $mvLX$ set equal to mvN and the reference array $refPicLX_L$ set equal to $refPicLN_L$, the variable $bdofFlag$ set equal to FALSE, the variable $cIdx$ is set equal to 0, and $RefPicScale[predListFlagN][refIdxN]$ as inputs.
 - The array $predSamplesLN_{Cb}$ is derived by invoking the fractional sample interpolation process specified in clause 8.5.6.3 with the luma location (x_{Cb}, y_{Cb}) , the coding block width $sbWidth$ set equal to $cbWidth / SubWidthC$, the coding block height $sbHeight$ set equal to $cbHeight / SubHeightC$, the motion vector offset $mvOffset$ set equal to $(0, 0)$, the motion vector $mvLX$ set equal to $mvCN$, and the reference array $refPicLX_{Cb}$ set equal to $refPicLN_{Cb}$, the variable $bdofFlag$ set equal to FALSE, the variable $cIdx$ is set equal to 1, and $RefPicScale[predListFlagN][refIdxN]$ as inputs.
 - The array $predSamplesLN_{Cr}$ is derived by invoking the fractional sample interpolation process specified in clause 8.5.6.3 with the luma location (x_{Cb}, y_{Cb}) , the coding block width $sbWidth$ set equal to $cbWidth / SubWidthC$, the coding block height $sbHeight$ set equal to $cbHeight / SubHeightC$, the motion vector offset $mvOffset$ set equal to $(0, 0)$, the motion vector $mvLX$ set equal to $mvCN$, and the reference array $refPicLX_{Cr}$ set equal to $refPicLN_{Cr}$, the variable

bdofFlag set equal to FALSE, the variable cIdx is set equal to 2, and

RefPicScale[predListFlagN][refIdxN] as inputs.

2. **The value of merge_geo_partition_idx'[xCb][yCb] are set according to the value of merge_geo_partition_idx[xCb][yCb] and the coding block width cbWidth and the coding block height cbHeight, as specified in Table xx.**
3. The partition angle and distance of merge geo mode variable angleIdx and distanceIdx are set according to the value of merge_geo_partition_idx'[xCb][yCb] as specified in Table 36.
4. The prediction samples inside the current luma coding block, predSamplesL[xL][yL] with $x_L = 0..cbWidth - 1$ and $y_L = 0..cbHeight - 1$, are derived by invoking the weighted sample prediction process for geo merge mode specified in clause 8.5.7.2 with the coding block width nCbW set equal to cbWidth, the coding block height nCbH set equal to cbHeight, the sample arrays predSamplesLA_L and predSamplesLB_L, and the variables angleIdx and distanceIdx, and cIdx equal to 0 as inputs.
5. The prediction samples inside the current chroma component Cb coding block, predSamplesCb[xC][yC] with $x_C = 0..cbWidth / SubWidthC - 1$ and $y_C = 0..cbHeight / SubHeightC - 1$, are derived by invoking the weighted sample prediction process for geo merge mode specified in clause 8.5.7.2 with the coding block width nCbW set equal to cbWidth / SubWidthC, the coding block height nCbH set equal to cbHeight / SubHeightC, the sample arrays predSamplesLA_{Cb} and predSamplesLB_{Cb}, and the variables angleIdx and distanceIdx, and cIdx equal to 1 as inputs.
6. The prediction samples inside the current chroma component Cr coding block, predSamplesCr[xC][yC] with $x_C = 0..cbWidth / SubWidthC - 1$ and $y_C = 0..cbHeight / SubHeightC - 1$, are derived by invoking the weighted sample prediction process for geo merge mode specified in clause 8.5.7.2 with the coding block width nCbW set equal to cbWidth / SubWidthC, the coding block height nCbH set equal to cbHeight / SubHeightC, the sample arrays predSamplesLA_{Cr} and predSamplesLB_{Cr}, and the variables angleIdx and distanceIdx, and cIdx equal to 2 as inputs.
7. The motion vector storing process for merge geo mode specified in clause 8.5.7.3 is invoked with the luma coding block location (xCb, yCb), the luma coding block width cbWidth, the luma coding block height cbHeight, the partition direction angleIdx and distanceIdx, the luma motion vectors mvA and mvB, the reference indices refIdxA and refIdxB, and the prediction list flags predListFlagA and predListFlagB as inputs.

FIG. 24A shows a table indicating mapping table of the geo_partition_idx' values based on the geo_partition_idx value. FIG. 24B shows an old version of Table 36 which is now deleted from the

relevant working draft. FIG. 24C shows Table 36 indicating specification of the angleIdx and distanceIdx values based on the geo_partition_idx value.

Table 123 – Syntax elements and associated binarizations

merge_data()	regular_merge_flag[][]	FL	cMax = 1
	mmvd_merge_flag[][]	FL	cMax = 1
	mmvd_cand_flag[][]	FL	cMax = 1
	mmvd_distance_idx[][]	TR	cMax = 7, cRiceParam = 0
	mmvd_direction_idx[][]	FL	cMax = 3
	ciip_flag[][]	FL	cMax = 1
	merge_subblock_flag[][]	FL	cMax = 1
	merge_subblock_idx[][]	TR	cMax = MaxNumSubblockMergeCand - 1, cRiceParam = 0
	merge_geo_partition_idx[][]	TB	cMax = [[82]] 32
	merge_geo_idx0[][]	TR	cMax = MaxNumGeoMergeCand - 1, cRiceParam = 0
	merge_geo_idx1[][]	TR	cMax = MaxNumGeoMergeCand - 2, cRiceParam = 0
	merge_idx[][]	TR	cMax = (CuPredMode[0][x0][y0] != MODE_IBC ? MaxNumMergeCand : MaxNumIbcMergeCand) - 1, cRiceParam = 0

An example embodiment: block size dependent GEO mode selection

8.5.7 Decoding process for geo inter blocks

8.5.7.1 General

This process is invoked when decoding a coding unit with MergeGeoFlag[xCb][yCb] equal to 1.

Inputs to this process are:

- a luma location (xCb, yCb) specifying the top-left sample of the current coding block relative to the top-left luma sample of the current picture,
- a variable cbWidth specifying the width of the current coding block in luma samples,
- a variable cbHeight specifying the height of the current coding block in luma samples,
- the luma motion vectors in 1/16 fractional-sample accuracy mvA and mvB,
- the chroma motion vectors mvCA and mvCB,
- the reference indices refIdxA and refIdxB,
- the prediction list flags predListFlagA and predListFlagB.

Outputs of this process are:

- an (cbWidth)x(cbHeight) array predSamples_L of luma prediction samples,

- an $(cbWidth / SubWidthC) \times (cbHeight / SubHeightC)$ array $predSamples_{Cb}$ of chroma prediction samples for the component Cb,
- an $(cbWidth / SubWidthC) \times (cbHeight / SubHeightC)$ array $predSamples_{Cr}$ of chroma prediction samples for the component Cr.

Let $predSamples_{LA_L}$ and $predSamples_{LB_L}$ be $(cbWidth) \times (cbHeight)$ arrays of predicted luma sample values and, $predSamples_{LA_{Cb}}$, $predSamples_{LB_{Cb}}$, $predSamples_{LA_{Cr}}$ and $predSamples_{LB_{Cr}}$ be $(cbWidth / SubWidthC) \times (cbHeight / SubHeightC)$ arrays of predicted chroma sample values.

The $predSamples_L$, $predSamples_{Cb}$ and $predSamples_{Cr}$ are derived by the following ordered steps:

1. For N being each of A and B, the following applies:
 - The reference picture consisting of an ordered two-dimensional array $refPicLN_L$ of luma samples and two ordered two-dimensional arrays $refPicLN_{Cb}$ and $refPicLN_{Cr}$ of chroma samples is derived by invoking the process specified in clause 8.5.6.2 with X set equal to $predListFlagN$ and $refIdxX$ set equal to $refIdxN$ as input.
 - The array $predSamplesLN_L$ is derived by invoking the fractional sample interpolation process specified in clause 8.5.6.3 with the luma location (xCb, yCb) , the luma coding block width $sbWidth$ set equal to $cbWidth$, the luma coding block height $sbHeight$ set equal to $cbHeight$, the motion vector offset $mvOffset$ set equal to $(0, 0)$, the motion vector $mvLX$ set equal to mvN and the reference array $refPicLX_L$ set equal to $refPicLN_L$, the variable $bdofFlag$ set equal to FALSE, the variable $cIdx$ is set equal to 0, and $RefPicScale[predListFlagN][refIdxN]$ as inputs.
 - The array $predSamplesLN_{Cb}$ is derived by invoking the fractional sample interpolation process specified in clause 8.5.6.3 with the luma location (xCb, yCb) , the coding block width $sbWidth$ set equal to $cbWidth / SubWidthC$, the coding block height $sbHeight$ set equal to $cbHeight / SubHeightC$, the motion vector offset $mvOffset$ set equal to $(0, 0)$, the motion vector $mvLX$ set equal to $mvCN$, and the reference array $refPicLX_{Cb}$ set equal to $refPicLN_{Cb}$, the variable $bdofFlag$ set equal to FALSE, the variable $cIdx$ is set equal to 1, and $RefPicScale[predListFlagN][refIdxN]$ as inputs.
 - The array $predSamplesLN_{Cr}$ is derived by invoking the fractional sample interpolation process specified in clause 8.5.6.3 with the luma location (xCb, yCb) , the coding block width $sbWidth$ set equal to $cbWidth / SubWidthC$, the coding block height $sbHeight$ set equal to $cbHeight / SubHeightC$, the motion vector offset $mvOffset$ set equal to $(0, 0)$, the motion vector $mvLX$ set equal to $mvCN$, and the reference array $refPicLX_{Cr}$ set equal to $refPicLN_{Cr}$, the variable $bdofFlag$ set equal to FALSE, the variable $cIdx$ is set equal to 2, and $RefPicScale[predListFlagN][refIdxN]$ as inputs.

2. The partition angle and distance of merge geo mode variable angleIdx and distanceIdx are set according to the value of merge_geo_partition_idx[xCb][yCb] as specified in Table 36.
3. The prediction samples inside the current luma coding block, predSamples_L[x_L][y_L] with x_L = 0..cbWidth - 1 and y_L = 0..cbHeight - 1, are derived by invoking the weighted sample prediction process for geo merge mode specified in clause 8.5.7.2 with the coding block width nCbW set equal to cbWidth, the coding block height nCbH set equal to cbHeight, the sample arrays predSamplesLA_L and predSamplesLB_L, and the variables angleIdx and distanceIdx, and cIdx equal to 0 as inputs.
4. The prediction samples inside the current chroma component Cb coding block, predSamples_{Cb}[x_C][y_C] with x_C = 0..cbWidth / SubWidthC - 1 and y_C = 0..cbHeight / SubHeightC - 1, are derived by invoking the weighted sample prediction process for geo merge mode specified in clause 8.5.7.2 with the coding block width nCbW set equal to cbWidth / SubWidthC, the coding block height nCbH set equal to cbHeight / SubHeightC, the sample arrays predSamplesLA_{Cb} and predSamplesLB_{Cb}, and the variables angleIdx and distanceIdx, and cIdx equal to 1 as inputs.
5. The prediction samples inside the current chroma component Cr coding block, predSamples_{Cr}[x_C][y_C] with x_C = 0..cbWidth / SubWidthC - 1 and y_C = 0..cbHeight / SubHeightC - 1, are derived by invoking the weighted sample prediction process for geo merge mode specified in clause 8.5.7.2 with the coding block width nCbW set equal to cbWidth / SubWidthC, the coding block height nCbH set equal to cbHeight / SubHeightC, the sample arrays predSamplesLA_{Cr} and predSamplesLB_{Cr}, and the variables angleIdx and distanceIdx, and cIdx equal to 2 as inputs.
6. The motion vector storing process for merge geo mode specified in clause 8.5.7.3 is invoked with the luma coding block location (xCb, yCb), the luma coding block width cbWidth, the luma coding block height cbHeight, the partition direction angleIdx and distanceIdx, the luma motion vectors mvA and mvB, the reference indices refIdxA and refIdxB, and the prediction list flags predListFlagA and predListFlagB as inputs.

The old version of Table 36 indicating specification of the angleIdx and distanceIdx values based on the geo_partition_idx value, which is shown in FIG. 24B, is now deleted from the relevant working draft. FIG. 24D shows Table 36 indicating specification of the angleIdx and distanceIdx values based on the geo_partition_idx value. **hwRatio = cbHeight > cbWidth ? 1: 0**

Table 123 – Syntax elements and associated binarizations

merge_data()	regular_merge_flag[][]	FL	cMax = 1
	mmvd_merge_flag[][]	FL	cMax = 1
	mmvd_cand_flag[][]	FL	cMax = 1
	mmvd_distance_idx[][]	TR	cMax = 7, cRiceParam = 0
	mmvd_direction_idx[][]	FL	cMax = 3
	ciip_flag[][]	FL	cMax = 1
	merge_subblock_flag[][]	FL	cMax = 1
	merge_subblock_idx[][]	TR	cMax = MaxNumSubblockMergeCand - 1, cRiceParam = 0
	merge_geo_partition_idx[][]	TB	cMax = [[82]] 32
	merge_geo_idx0[][]	TR	cMax = MaxNumGeoMergeCand - 1, cRiceParam = 0
	merge_geo_idx1[][]	TR	cMax = MaxNumGeoMergeCand - 2, cRiceParam = 0
	merge_idx[][]	TR	cMax = (CuPredMode[0][x0][y0] != MODE_JBC ? MaxNumMergeCand : MaxNumIbcMergeCand) - 1, cRiceParam = 0

An example embodiment: block size dependent GEO mode selection

7.3.2.3 Sequence parameter set RBSP syntax

seq_parameter_set_rbsp() {	Descriptor
sps_decoding_parameter_set_id	u(4)
sps_video_parameter_set_id	u(4)
..	
sps_ciip_enabled_flag	u(1)
if(sps_mmvd_enabled_flag)	
sps_fpel_mmvd_enabled_flag	u(1)
sps_geo_enabled_flag	u(1)
<u>if(sps_geo_enabled_flag)</u>	
<u>sps_16_modes_geo_enabled_flag</u>	<u>u(1)</u>
sps_lmcs_enabled_flag	u(1)

sps_lfst_enabled_flag	u(1)
...	
rbsp_trailing_bits()	
}	

sps_16_modes_geo_enabled_flag equal to 1 specifies that 16 modes geo is used.

sps_16_modes_geo_enabled_flag equal to 0 specifies that 32 modes geo is used.

7.4.10.7 Merge data semantics

merge_geo_partition_idx[x0][y0] specifies the geometric splitting direction of the merge geometric mode. The array indices x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered coding block relative to the top-left luma sample of the picture.

When **merge_geo_partition_idx[x0][y0]** is not present, it is inferred to be equal to 0.

It is constraint that the maximum value of merge_geo_partition_idx should be less than

sps_16_modes_geo_enabled_flag ? 16 : 32.

8.5.7 Decoding process for geo inter blocks

8.5.7.1 General

This process is invoked when decoding a coding unit with **MergeGeoFlag[xCb][yCb]** equal to 1.

Inputs to this process are:

- a luma location (xCb, yCb) specifying the top-left sample of the current coding block relative to the top-left luma sample of the current picture,
- a variable cbWidth specifying the width of the current coding block in luma samples,
- a variable cbHeight specifying the height of the current coding block in luma samples,
- the luma motion vectors in 1/16 fractional-sample accuracy mvA and mvB,
- the chroma motion vectors mvCA and mvCB,
- the reference indices refIdxA and refIdxB,
- the prediction list flags predListFlagA and predListFlagB.

Outputs of this process are:

- an (cbWidth)x(cbHeight) array **predSamples_L** of luma prediction samples,
- an (cbWidth / SubWidthC)x(cbHeight / SubHeightC) array **predSamples_{Cb}** of chroma prediction samples for the component Cb,
- an (cbWidth / SubWidthC)x(cbHeight / SubHeightC) array **predSamples_{Cr}** of chroma prediction samples for the component Cr.

Let predSamplesLA_L and predSamplesLB_L be $(\text{cbWidth}) \times (\text{cbHeight})$ arrays of predicted luma sample values and, predSamplesLAC_b , predSamplesLBC_b , predSamplesLAC_r and predSamplesLBC_r be $(\text{cbWidth} / \text{SubWidthC}) \times (\text{cbHeight} / \text{SubHeightC})$ arrays of predicted chroma sample values.

The predSamples_L , predSamples_{Cb} and predSamples_{Cr} are derived by the following ordered steps:

1. For N being each of A and B, the following applies:
 - The reference picture consisting of an ordered two-dimensional array refPicLN_L of luma samples and two ordered two-dimensional arrays refPicLNC_b and refPicLNC_r of chroma samples is derived by invoking the process specified in clause 8.5.6.2 with X set equal to predListFlagN and refIdxX set equal to refIdxN as input.
 - The array predSamplesLN_L is derived by invoking the fractional sample interpolation process specified in clause 8.5.6.3 with the luma location (x_{Cb}, y_{Cb}) , the luma coding block width sbWidth set equal to cbWidth , the luma coding block height sbHeight set equal to cbHeight , the motion vector offset mvOffset set equal to $(0, 0)$, the motion vector mvLX set equal to mvN and the reference array refPicLXL set equal to refPicLN_L , the variable bdofFlag set equal to FALSE, the variable cIdx is set equal to 0, and $\text{RefPicScale}[\text{predListFlagN}][\text{refIdxN}]$ as inputs.
 - The array predSamplesLNC_b is derived by invoking the fractional sample interpolation process specified in clause 8.5.6.3 with the luma location (x_{Cb}, y_{Cb}) , the coding block width sbWidth set equal to $\text{cbWidth} / \text{SubWidthC}$, the coding block height sbHeight set equal to $\text{cbHeight} / \text{SubHeightC}$, the motion vector offset mvOffset set equal to $(0, 0)$, the motion vector mvLX set equal to mvCN , and the reference array refPicLXC_b set equal to refPicLNC_b , the variable bdofFlag set equal to FALSE, the variable cIdx is set equal to 1, and $\text{RefPicScale}[\text{predListFlagN}][\text{refIdxN}]$ as inputs.
 - The array predSamplesLNC_r is derived by invoking the fractional sample interpolation process specified in clause 8.5.6.3 with the luma location (x_{Cb}, y_{Cb}) , the coding block width sbWidth set equal to $\text{cbWidth} / \text{SubWidthC}$, the coding block height sbHeight set equal to $\text{cbHeight} / \text{SubHeightC}$, the motion vector offset mvOffset set equal to $(0, 0)$, the motion vector mvLX set equal to mvCN , and the reference array refPicLXC_r set equal to refPicLNC_r , the variable bdofFlag set equal to FALSE, the variable cIdx is set equal to 2, and $\text{RefPicScale}[\text{predListFlagN}][\text{refIdxN}]$ as inputs.
2. **The value of merge_geo_partition_idx'[xCb][yCb] are set according to the value of merge_geo_partition_idx[xCb][yCb] and the coding block width cbWidth and the coding block height cbHeight, as specified in Table xx.**

3. The partition angle and distance of merge geo mode variable angleIdx and distanceIdx are set according to the value of merge_geo_partition_idx'[xCb][yCb] as specified in Table 36.
4. The prediction samples inside the current luma coding block, predSamples_L[x_L][y_L] with x_L = 0..cbWidth - 1 and y_L = 0..cbHeight - 1, are derived by invoking the weighted sample prediction process for geo merge mode specified in clause 8.5.7.2 with the coding block width nCbW set equal to cbWidth, the coding block height nCbH set equal to cbHeight, the sample arrays predSamplesLA_L and predSamplesLB_L, and the variables angleIdx and distanceIdx, and cIdx equal to 0 as inputs.
5. The prediction samples inside the current chroma component Cb coding block, predSamples_{Cb}[x_C][y_C] with x_C = 0..cbWidth / SubWidthC - 1 and y_C = 0..cbHeight / SubHeightC - 1, are derived by invoking the weighted sample prediction process for geo merge mode specified in clause 8.5.7.2 with the coding block width nCbW set equal to cbWidth / SubWidthC, the coding block height nCbH set equal to cbHeight / SubHeightC, the sample arrays predSamplesLA_{Cb} and predSamplesLB_{Cb}, and the variables angleIdx and distanceIdx, and cIdx equal to 1 as inputs.
6. The prediction samples inside the current chroma component Cr coding block, predSamples_{Cr}[x_C][y_C] with x_C = 0..cbWidth / SubWidthC - 1 and y_C = 0..cbHeight / SubHeightC - 1, are derived by invoking the weighted sample prediction process for geo merge mode specified in clause 8.5.7.2 with the coding block width nCbW set equal to cbWidth / SubWidthC, the coding block height nCbH set equal to cbHeight / SubHeightC, the sample arrays predSamplesLA_{Cr} and predSamplesLB_{Cr}, and the variables angleIdx and distanceIdx, and cIdx equal to 2 as inputs.
7. The motion vector storing process for merge geo mode specified in clause 8.5.7.3 is invoked with the luma coding block location (xCb, yCb), the luma coding block width cbWidth, the luma coding block height cbHeight, the partition direction angleIdx and distanceIdx, the luma motion vectors mvA and mvB, the reference indices refIdxA and refIdxB, and the prediction list flags predListFlagA and predListFlagB as inputs.

In this example, a mapping table of the merge_geo_partition_idx' values based on the geo_partition_idx value as shown in FIG. 24A is used. The old version of Table 36 indicating Specification of the angleIdx and distanceIdx values based on the geo_partition_idx value, which is deleted from the working draft is shown in FIG. 24B. FIG. 24E shows Table 36 indicating specification of the angleIdx and distanceIdx values based on the geo_partition_idx value.

Table 123 – Syntax elements and associated binarizations

merge_data()	regular_merge_flag[][]	FL	cMax = 1
	mmvd_merge_flag[][]	FL	cMax = 1
	mmvd_cand_flag[][]	FL	cMax = 1
	mmvd_distance_idx[][]	TR	cMax = 7, cRiceParam = 0
	mmvd_direction_idx[][]	FL	cMax = 3
	ciip_flag[][]	FL	cMax = 1
	merge_subblock_flag[][]	FL	cMax = 1
	merge_subblock_idx[][]	TR	cMax = MaxNumSubblockMergeCand – 1, cRiceParam = 0
	merge_geo_partition_idx[][]	TB	cMax = [[82]](sps_16_modes_geo_enabled_flag ? 16 : 32)
	merge_geo_idx0[][]	TR	cMax = MaxNumGeoMergeCand – 1, cRiceParam = 0
	merge_geo_idx1[][]	TR	cMax = MaxNumGeoMergeCand – 2, cRiceParam = 0
	merge_idx[][]	TR	cMax = (CuPredMode[0][x0][y0] != MODE_IBC ? MaxNumMergeCand : MaxNumIbcMergeCand) – 1, cRiceParam = 0

An example embodiment: block size dependent GEO mode selection

7.3.2.3 Sequence parameter set RBSP syntax

seq_parameter_set_rbsp() {	Descriptor
sps_decoding_parameter_set_id	u(4)
sps_video_parameter_set_id	u(4)

..	
sps_ciip_enabled_flag	u(1)
if(sps_mmvd_enabled_flag)	
sps_fpel_mmvd_enabled_flag	u(1)
sps_geo_enabled_flag	u(1)
<u>if(sps_geo_enabled_flag)</u>	
<u>sps_16_modes_geo_enabled_flag</u>	<u>u(1)</u>
sps_lmcs_enabled_flag	u(1)
sps_lfnst_enabled_flag	u(1)
...	
rbsp_trailing_bits()	
}	

sps_16_modes_geo_enabled_flag equal to 1 specifies that 16 modes geo is used for specified blocks.

sps_16_modes_geo_enabled_flag equal to 0 specifies that 32 modes geo is used for specified blocks.

7.4.10.7 Merge data semantics

merge_geo_partition_idx[x0][y0] specifies the geometric splitting direction of the merge geometric mode. The array indices x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered coding block relative to the top-left luma sample of the picture.

When **merge_geo_partition_idx[x0][y0]** is not present, it is inferred to be equal to 0.

It is constraint that the maximum value of merge_geo_partition_idx should be less than

(sps_16_modes_geo_enabled_flag && cbWidth >= cbHeight) ? 16 : 32.

8.5.7 Decoding process for geo inter blocks

8.5.7.1 General

This process is invoked when decoding a coding unit with **MergeGeoFlag[xCb][yCb]** equal to 1.

Inputs to this process are:

- a luma location (xCb, yCb) specifying the top-left sample of the current coding block relative to the top-left luma sample of the current picture,
- a variable cbWidth specifying the width of the current coding block in luma samples,

- a variable $cbHeight$ specifying the height of the current coding block in luma samples,
- the luma motion vectors in 1/16 fractional-sample accuracy mvA and mvB ,
- the chroma motion vectors $mvCA$ and $mvCB$,
- the reference indices $refIdxA$ and $refIdxB$,
- the prediction list flags $predListFlagA$ and $predListFlagB$.

Outputs of this process are:

- an $(cbWidth) \times (cbHeight)$ array $predSamples_L$ of luma prediction samples,
- an $(cbWidth / SubWidthC) \times (cbHeight / SubHeightC)$ array $predSamples_{Cb}$ of chroma prediction samples for the component Cb ,
- an $(cbWidth / SubWidthC) \times (cbHeight / SubHeightC)$ array $predSamples_{Cr}$ of chroma prediction samples for the component Cr .

Let $predSamples_{LA_L}$ and $predSamples_{LB_L}$ be $(cbWidth) \times (cbHeight)$ arrays of predicted luma sample values and, $predSamples_{LA_{Cb}}$, $predSamples_{LB_{Cb}}$, $predSamples_{LA_{Cr}}$ and $predSamples_{LB_{Cr}}$ be $(cbWidth / SubWidthC) \times (cbHeight / SubHeightC)$ arrays of predicted chroma sample values.

The $predSamples_L$, $predSamples_{Cb}$ and $predSamples_{Cr}$ are derived by the following ordered steps:

1. For N being each of A and B , the following applies:
 - The reference picture consisting of an ordered two-dimensional array $refPicLN_L$ of luma samples and two ordered two-dimensional arrays $refPicLN_{Cb}$ and $refPicLN_{Cr}$ of chroma samples is derived by invoking the process specified in clause 8.5.6.2 with X set equal to $predListFlagN$ and $refIdxX$ set equal to $refIdxN$ as input.
 - The array $predSamplesLN_L$ is derived by invoking the fractional sample interpolation process specified in clause 8.5.6.3 with the luma location (x_{Cb}, y_{Cb}) , the luma coding block width $sbWidth$ set equal to $cbWidth$, the luma coding block height $sbHeight$ set equal to $cbHeight$, the motion vector offset $mvOffset$ set equal to $(0, 0)$, the motion vector $mvLX$ set equal to mvN and the reference array $refPicLX_L$ set equal to $refPicLN_L$, the variable $bdofFlag$ set equal to $FALSE$, the variable $cIdx$ is set equal to 0 , and $RefPicScale[predListFlagN][refIdxN]$ as inputs.
 - The array $predSamplesLN_{Cb}$ is derived by invoking the fractional sample interpolation process specified in clause 8.5.6.3 with the luma location (x_{Cb}, y_{Cb}) , the coding block width $sbWidth$ set equal to $cbWidth / SubWidthC$, the coding block height $sbHeight$ set equal to $cbHeight / SubHeightC$, the motion vector offset $mvOffset$ set equal to $(0, 0)$, the motion vector $mvLX$ set equal to $mvCN$, and the reference array $refPicLX_{Cb}$ set equal to $refPicLN_{Cb}$, the variable $bdofFlag$ set equal to $FALSE$, the variable $cIdx$ is set equal to 1 , and $RefPicScale[predListFlagN][refIdxN]$ as inputs.

- The array $\text{predSamplesLN}_{Cr}$ is derived by invoking the fractional sample interpolation process specified in clause 8.5.6.3 with the luma location (x_{Cb} , y_{Cb}), the coding block width $sbWidth$ set equal to $cbWidth / SubWidthC$, the coding block height $sbHeight$ set equal to $cbHeight / SubHeightC$, the motion vector offset $mvOffset$ set equal to (0, 0), the motion vector $mvLX$ set equal to $mvCN$, and the reference array refPicLX_{Cr} set equal to refPicLN_{Cr} , the variable $bdofFlag$ set equal to FALSE, the variable $cIdx$ is set equal to 2, and $\text{RefPicScale}[\text{predListFlagN}][\text{refIdxN}]$ as inputs.
2. **The value of merge_geo_partition_idx'[x_{Cb}][y_{Cb}] are set according to the value of merge_geo_partition_idx[x_{Cb}][y_{Cb}] and the coding block width $cbWidth$ and the coding block height $cbHeight$, as specified in Table xx.**
 3. The partition angle and distance of merge geo mode variable $angleIdx$ and $distanceIdx$ are set according to the value of $\text{merge_geo_partition_idx}'[x_{Cb}][y_{Cb}]$ as specified in Table 36.
 4. The prediction samples inside the current luma coding block, $\text{predSamples}_L[x_L][y_L]$ with $x_L = 0..cbWidth - 1$ and $y_L = 0..cbHeight - 1$, are derived by invoking the weighted sample prediction process for geo merge mode specified in clause 8.5.7.2 with the coding block width $nCbW$ set equal to $cbWidth$, the coding block height $nCbH$ set equal to $cbHeight$, the sample arrays predSamplesLA_L and predSamplesLB_L , and the variables $angleIdx$ and $distanceIdx$, and $cIdx$ equal to 0 as inputs.
 5. The prediction samples inside the current chroma component Cb coding block, $\text{predSamples}_{Cb}[x_C][y_C]$ with $x_C = 0..cbWidth / SubWidthC - 1$ and $y_C = 0..cbHeight / SubHeightC - 1$, are derived by invoking the weighted sample prediction process for geo merge mode specified in clause 8.5.7.2 with the coding block width $nCbW$ set equal to $cbWidth / SubWidthC$, the coding block height $nCbH$ set equal to $cbHeight / SubHeightC$, the sample arrays $\text{predSamplesLA}_{Cb}$ and $\text{predSamplesLB}_{Cb}$, and the variables $angleIdx$ and $distanceIdx$, and $cIdx$ equal to 1 as inputs.
 6. The prediction samples inside the current chroma component Cr coding block, $\text{predSamples}_{Cr}[x_C][y_C]$ with $x_C = 0..cbWidth / SubWidthC - 1$ and $y_C = 0..cbHeight / SubHeightC - 1$, are derived by invoking the weighted sample prediction process for geo merge mode specified in clause 8.5.7.2 with the coding block width $nCbW$ set equal to $cbWidth / SubWidthC$, the coding block height $nCbH$ set equal to $cbHeight / SubHeightC$, the sample arrays $\text{predSamplesLA}_{Cr}$ and $\text{predSamplesLB}_{Cr}$, and the variables $angleIdx$ and $distanceIdx$, and $cIdx$ equal to 2 as inputs.
 7. The motion vector storing process for merge geo mode specified in clause 8.5.7.3 is invoked with the luma coding block location (x_{Cb} , y_{Cb}), the luma coding block width $cbWidth$, the luma coding block height $cbHeight$, the partition direction $angleIdx$ and $distanceIdx$, the luma motion vectors mvA and

mvB, the reference indices refIdxA and refIdxB, and the prediction list flags predListFlagA and predListFlagB as inputs.

In this example, a mapping table of the geo_partition_idx' values based on the geo_partition_idx value as shown in FIG. 24A is used. The old version of Table 36 indicating Specification of the angleIdx and distanceIdx values based on the geo_partition_idx value, which is delted from the working draft is shown in FIG. 24B. Table 36 indicating specification of the angleIdx and distanceIdx values based on the geo_partition_idx value is shown in FIG. 24E.

Table 123 – Syntax elements and associated binarizations

merge_data()	regular_merge_flag[][]	FL	cMax = 1
	mmvd_merge_flag[][]	FL	cMax = 1
	mmvd_cand_flag[][]	FL	cMax = 1
	mmvd_distance_idx[][]	TR	cMax = 7, cRiceParam = 0
	mmvd_direction_idx[][]	FL	cMax = 3
	ciip_flag[][]	FL	cMax = 1
	merge_subblock_flag[][]	FL	cMax = 1
	merge_subblock_idx[][]	TR	cMax = MaxNumSubblockMergeCand - 1, cRiceParam = 0
	merge_geo_partition_idx[][]	TB	cMax = [[82]] (sps_16_modes_geo_enabled_flag && cbWidth >= cbHeight ? 16 : 32)
	merge_geo_idx0[][]	TR	cMax = MaxNumGeoMergeCand - 1, cRiceParam = 0
	merge_geo_idx1[][]	TR	cMax = MaxNumGeoMergeCand - 2, cRiceParam = 0
	merge_idx[][]	TR	cMax = (CuPredMode[0][x0][y0] != MODE_IBC ? MaxNumMergeCand : MaxNumIbcMergeCand) - 1, cRiceParam = 0

An example embodiment: block size dependent GEO mode selection

7.3.2.3 Sequence parameter set RBSP syntax

seq_parameter_set_rbsp() {	Descriptor
sps_decoding_parameter_set_id	u(4)
sps_video_parameter_set_id	u(4)

..	
sps_ciip_enabled_flag	u(1)
if(sps_mmvd_enabled_flag)	
sps_fpel_mmvd_enabled_flag	u(1)
sps_geo_enabled_flag	u(1)
<u>if(sps_geo_enabled_flag)</u>	
<u>sps_16_modes_geo_enabled_flag</u>	<u>u(1)</u>
sps_lmcs_enabled_flag	u(1)
sps_lfnst_enabled_flag	u(1)
...	
rbsp_trailing_bits()	
}	

sps_16_modes_geo_enabled_flag equal to 1 specifies that 16 modes geo is used for specified blocks.

sps_16_modes_geo_enabled_flag equal to 0 specifies that 32 modes geo is used for specified blocks.

7.4.10.7 Merge data semantics

`merge_geo_partition_idx[ x0 ][ y0 ]` specifies the geometric splitting direction of the merge geometric mode. The array indices `x0`, `y0` specify the location (`x0`, `y0`) of the top-left luma sample of the considered coding block relative to the top-left luma sample of the picture.

When `merge_geo_partition_idx[ x0 ][ y0 ]` is not present, it is inferred to be equal to 0.

It is constraint that the maximum value of `merge_geo_partition_idx` should be less than

(`sps_16_modes_geo_enabled_flag` & `cbWidth` >= `cbHeight`) ? 16 : 32.

8.5.7 Decoding process for geo inter blocks

8.5.7.1 General

This process is invoked when decoding a coding unit with `MergeGeoFlag[ xCb ][ yCb ]` equal to 1.

Inputs to this process are:

- a luma location (`xCb`, `yCb`) specifying the top-left sample of the current coding block relative to the top-left luma sample of the current picture,
- a variable `cbWidth` specifying the width of the current coding block in luma samples,

- a variable $cbHeight$ specifying the height of the current coding block in luma samples,
- the luma motion vectors in 1/16 fractional-sample accuracy mvA and mvB ,
- the chroma motion vectors $mvCA$ and $mvCB$,
- the reference indices $refIdxA$ and $refIdxB$,
- the prediction list flags $predListFlagA$ and $predListFlagB$.

Outputs of this process are:

- an $(cbWidth) \times (cbHeight)$ array $predSamples_L$ of luma prediction samples,
- an $(cbWidth / SubWidthC) \times (cbHeight / SubHeightC)$ array $predSamples_{Cb}$ of chroma prediction samples for the component Cb ,
- an $(cbWidth / SubWidthC) \times (cbHeight / SubHeightC)$ array $predSamples_{Cr}$ of chroma prediction samples for the component Cr .

Let $predSamples_{LA_L}$ and $predSamples_{LB_L}$ be $(cbWidth) \times (cbHeight)$ arrays of predicted luma sample values and, $predSamples_{LA_{Cb}}$, $predSamples_{LB_{Cb}}$, $predSamples_{LA_{Cr}}$ and $predSamples_{LB_{Cr}}$ be $(cbWidth / SubWidthC) \times (cbHeight / SubHeightC)$ arrays of predicted chroma sample values.

The $predSamples_L$, $predSamples_{Cb}$ and $predSamples_{Cr}$ are derived by the following ordered steps:

8. For N being each of A and B , the following applies:
 - The reference picture consisting of an ordered two-dimensional array $refPicLN_L$ of luma samples and two ordered two-dimensional arrays $refPicLN_{Cb}$ and $refPicLN_{Cr}$ of chroma samples is derived by invoking the process specified in clause 8.5.6.2 with X set equal to $predListFlagN$ and $refIdxX$ set equal to $refIdxN$ as input.
 - The array $predSamplesLN_L$ is derived by invoking the fractional sample interpolation process specified in clause 8.5.6.3 with the luma location (x_{Cb}, y_{Cb}) , the luma coding block width $sbWidth$ set equal to $cbWidth$, the luma coding block height $sbHeight$ set equal to $cbHeight$, the motion vector offset $mvOffset$ set equal to $(0, 0)$, the motion vector $mvLX$ set equal to mvN and the reference array $refPicLX_L$ set equal to $refPicLN_L$, the variable $bdofFlag$ set equal to $FALSE$, the variable $cIdx$ is set equal to 0 , and $RefPicScale[predListFlagN][refIdxN]$ as inputs.
 - The array $predSamplesLN_{Cb}$ is derived by invoking the fractional sample interpolation process specified in clause 8.5.6.3 with the luma location (x_{Cb}, y_{Cb}) , the coding block width $sbWidth$ set equal to $cbWidth / SubWidthC$, the coding block height $sbHeight$ set equal to $cbHeight / SubHeightC$, the motion vector offset $mvOffset$ set equal to $(0, 0)$, the motion vector $mvLX$ set equal to $mvCN$, and the reference array $refPicLX_{Cb}$ set equal to $refPicLN_{Cb}$, the variable $bdofFlag$ set equal to $FALSE$, the variable $cIdx$ is set equal to 1 , and $RefPicScale[predListFlagN][refIdxN]$ as inputs.

- The array $\text{predSamplesLN}_{Cr}$ is derived by invoking the fractional sample interpolation process specified in clause 8.5.6.3 with the luma location (x_{Cb} , y_{Cb}), the coding block width $sbWidth$ set equal to $cbWidth / SubWidthC$, the coding block height $sbHeight$ set equal to $cbHeight / SubHeightC$, the motion vector offset $mvOffset$ set equal to (0, 0), the motion vector $mvLX$ set equal to $mvCN$, and the reference array refPicLX_{Cr} set equal to refPicLN_{Cr} , the variable $bdofFlag$ set equal to FALSE, the variable $cIdx$ is set equal to 2, and $\text{RefPicScale}[\text{predListFlagN}][\text{refIdxN}]$ as inputs.
9. **If $\text{sps_16_modes_geo_enabled_flag}$ is equal to 1 and $cbWidth$ is greater than or equal to $cbHeight$ and $\text{merge_geo_partition_idx}[\text{xCb}][\text{yCb}]$ is larger than X (for example, $X=13$), the value of $\text{merge_geo_partition_idx}'[\text{xCb}][\text{yCb}]$ are set according to follows,**
- **If $\text{merge_geo_partition_idx}[\text{xCb}][\text{yCb}]$ is equal to XX (for example, $XX=14$), $\text{merge_geo_partition_idx}[\text{xCb}][\text{yCb}]$ is set to YY (for example, $YY=10$).**
 - **If $\text{merge_geo_partition_idx}[\text{xCb}][\text{yCb}]$ is equal to XXX (for example, $XX=15$), $\text{merge_geo_partition_idx}[\text{xCb}][\text{yCb}]$ is set to YYY (for example, $YYY=24$).**
- Otherwise, The value of $\text{merge_geo_partition_idx}'[\text{xCb}][\text{yCb}]$ are set according to the value of $\text{merge_geo_partition_idx}[\text{xCb}][\text{yCb}]$ and the coding block width $cbWidth$ and the coding block height $cbHeight$, as specified in Table xx.**
10. The partition angle and distance of merge geo mode variable angleIdx and distanceIdx are set according to the value of $\text{merge_geo_partition_idx}'[\text{xCb}][\text{yCb}]$ as specified in Table 36.
11. The prediction samples inside the current luma coding block, $\text{predSamples}_L[\text{x}_L][\text{y}_L]$ with $\text{x}_L = 0..cbWidth - 1$ and $\text{y}_L = 0..cbHeight - 1$, are derived by invoking the weighted sample prediction process for geo merge mode specified in clause 8.5.7.2 with the coding block width $nCbW$ set equal to $cbWidth$, the coding block height $nCbH$ set equal to $cbHeight$, the sample arrays predSamplesLA_L and predSamplesLB_L , and the variables angleIdx and distanceIdx , and $cIdx$ equal to 0 as inputs.
12. The prediction samples inside the current chroma component Cb coding block, $\text{predSamples}_{Cb}[\text{x}_C][\text{y}_C]$ with $\text{x}_C = 0..cbWidth / SubWidthC - 1$ and $\text{y}_C = 0..cbHeight / SubHeightC - 1$, are derived by invoking the weighted sample prediction process for geo merge mode specified in clause 8.5.7.2 with the coding block width $nCbW$ set equal to $cbWidth / SubWidthC$, the coding block height $nCbH$ set equal to $cbHeight / SubHeightC$, the sample arrays $\text{predSamplesLA}_{Cb}$ and $\text{predSamplesLB}_{Cb}$, and the variables angleIdx and distanceIdx , and $cIdx$ equal to 1 as inputs.
13. The prediction samples inside the current chroma component Cr coding block, $\text{predSamples}_{Cr}[\text{x}_C][\text{y}_C]$ with $\text{x}_C = 0..cbWidth / SubWidthC - 1$ and $\text{y}_C = 0..cbHeight / SubHeightC - 1$, are derived by invoking

the weighted sample prediction process for geo merge mode specified in clause 8.5.7.2 with the coding block width $nCbW$ set equal to $cbWidth / SubWidthC$, the coding block height $nCbH$ set equal to $cbHeight / SubHeightC$, the sample arrays $predSamplesLA_{Cr}$ and $predSamplesLB_{Cr}$, and the variables $angleIdx$ and $distanceIdx$, and $cIdx$ equal to 2 as inputs.

14. The motion vector storing process for merge geo mode specified in clause 8.5.7.3 is invoked with the luma coding block location (xCb , yCb), the luma coding block width $cbWidth$, the luma coding block height $cbHeight$, the partition direction $angleIdx$ and $distanceIdx$, the luma motion vectors mvA and mvB , the reference indices $refIdxA$ and $refIdxB$, and the prediction list flags $predListFlagA$ and $predListFlagB$ as inputs.

In this example, a mapping table of the $geo_partition_idx$ ' values based on the $geo_partition_idx$ value as shown in FIG. 24A is used. The old version of Table 36 indicating Specification of the $angleIdx$ and $distanceIdx$ values based on the $geo_partition_idx$ value, which is delted from the working draft is shown in FIG. 24B. FIG. 24E shows Table 36 indicating specification of the $angleIdx$ and $distanceIdx$ values based on the $geo_partition_idx$ value.

Table 123 – Syntax elements and associated binarizations

merge_data()	regular_merge_flag[][]	FL	cMax = 1
	mmvd_merge_flag[][]	FL	cMax = 1
	mmvd_cand_flag[][]	FL	cMax = 1
	mmvd_distance_idx[][]	TR	cMax = 7, cRiceParam = 0
	mmvd_direction_idx[][]	FL	cMax = 3
	ciip_flag[][]	FL	cMax = 1
	merge_subblock_flag[][]	FL	cMax = 1
	merge_subblock_idx[][]	TR	cMax = MaxNumSubblockMergeCand - 1, cRiceParam = 0
	merge_geo_partition_idx[][]	TB	cMax = [[82]] (sps_16_modes_geo_enabled_flag && cbWidth >= cbHeight ? 16 : 32)
	merge_geo_idx0[][]	TR	cMax = MaxNumGeoMergeCand - 1, cRiceParam = 0
	merge_geo_idx1[][]	TR	cMax = MaxNumGeoMergeCand - 2, cRiceParam = 0
	merge_idx[][]	TR	cMax = (CuPredMode[0][x0][y0] != MODE_IBC ? MaxNumMergeCand : MaxNumIbcMergeCand) - 1, cRiceParam = 0

[0115] Example implementations of the disclosed technology

[0116] FIG. 13A is a block diagram of a video processing apparatus 1300. The apparatus 1300 may be used to implement one or more of the methods described herein. The apparatus 1300 may be embodied in a smartphone, tablet, computer, Internet of Things (IoT) receiver, and so on. The apparatus 1300 may include one or more processors 1302, one or more memories 1304 and video processing hardware 1306. The processor(s) 1302 may be configured to implement one or more methods described in the present document. The memory (memories) 1304 may be used for storing data and code used for implementing the methods and techniques described herein. The video processing hardware 1306 may be used to implement, in hardware circuitry, some techniques described in the present document, and may be partly or completely be a part of the processors 1302 (e.g., graphics processor core GPU or other signal processing circuitry).

[0117] In the present document, the term “video processing” may refer to video encoding, video decoding, video compression or video decompression. For example, video compression algorithms

may be applied during conversion from pixel representation of a video to a corresponding bitstream representation or vice versa. The bitstream representation of a current video block may, for example, correspond to bits that are either co-located or spread in different places within the bitstream, as is defined by the syntax. For example, a macroblock may be encoded in terms of transformed and coded error residual values and also using bits in headers and other fields in the bitstream.

[0118] It will be appreciated that the disclosed methods and techniques will benefit video encoder and/or decoder embodiments incorporated within video processing devices such as smartphones, laptops, desktops, and similar devices by allowing the use of the techniques disclosed in the present document.

[0119] FIG. 13B is a block diagram showing an example video processing system 1310 in which various techniques disclosed herein may be implemented. Various implementations may include some or all of the components of the system 1310. The system 1310 may include input 1312 for receiving video content. The video content may be received in a raw or uncompressed format, e.g., 8 or 10 bit multi-component pixel values, or may be in a compressed or encoded format. The input 1312 may represent a network interface, a peripheral bus interface, or a storage interface. Examples of network interface include wired interfaces such as Ethernet, passive optical network (PON), etc. and wireless interfaces such as Wi-Fi or cellular interfaces.

[0120] The system 1310 may include a coding component 1314 that may implement the various coding or encoding methods described in the present document. The coding component 1314 may reduce the average bitrate of video from the input 1312 to the output of the coding component 1314 to produce a coded representation of the video. The coding techniques are therefore sometimes called video compression or video transcoding techniques. The output of the coding component 1314 may be either stored, or transmitted via a communication connected, as represented by the component 1316. The stored or communicated bitstream (or coded) representation of the video received at the input 1312 may be used by the component 1318 for generating pixel values or displayable video that is sent to a display interface 1320. The process of generating user-viewable video from the bitstream representation is sometimes called video decompression. Furthermore, while certain video processing operations are referred to as “coding” operations or tools, it will be appreciated that the coding tools or operations are used at an encoder and corresponding decoding tools or operations that reverse the results of the coding will be performed by a decoder.

[0121] Examples of a peripheral bus interface or a display interface may include universal serial bus (USB) or high definition multimedia interface (HDMI) or Displayport, and so on. Examples of storage interfaces include SATA (serial advanced technology attachment), PCI, IDE interface, and the like. The techniques described in the present document may be embodied in various electronic devices such as mobile phones, laptops, smartphones or other devices that are capable of performing digital data processing and/or video display.

[0122] Some embodiments of the disclosed technology include making a decision or determination to enable a video processing tool or mode. In an example, when the video processing tool or mode is enabled, the encoder will use or implement the tool or mode in the processing of a block of video, but may not necessarily modify the resulting bitstream based on the usage of the tool or mode. That is, a conversion from the block of video to the bitstream representation of the video will use the video processing tool or mode when it is enabled based on the decision or determination. In another example, when the video processing tool or mode is enabled, the decoder will process the bitstream with the knowledge that the bitstream has been modified based on the video processing tool or mode. That is, a conversion from the bitstream representation of the video to the block of video will be performed using the video processing tool or mode that was enabled based on the decision or determination.

[0123] Some embodiments of the disclosed technology include making a decision or determination to disable a video processing tool or mode. In an example, when the video processing tool or mode is disabled, the encoder will not use the tool or mode in the conversion of the block of video to the bitstream representation of the video. In another example, when the video processing tool or mode is disabled, the decoder will process the bitstream with the knowledge that the bitstream has not been modified using the video processing tool or mode that was enabled based on the decision or determination.

[0124] FIG. 14 is a flowchart for an example method 1400 of video processing. The method 1400 includes, at 1402, performing a conversion between a current video block of visual media data and a bitstream representation of the visual media data, wherein, during the conversion, a use of a geometry partition mode is selectively enabled or disabled based at least in part on a determination that that one or more dimensions of the current video block and/or a mathematical function of the one or more dimensions of the current video block achieves at least one threshold condition.

[0125] FIG.15 is a block diagram that illustrates an example video coding system 100 that may utilize the techniques of this disclosure.

[0126] As shown in FIG. 15, video coding system 100 may include a source device 110 and a destination device 120. Source device 110 generates encoded video data which may be referred to as a video encoding device. Destination device 120 may decode the encoded video data generated by source device 110 which may be referred to as a video decoding device.

[0127] Source device 110 may include a video source 112, a video encoder 114, and an input/output (I/O) interface 116.

[0128] Video source 112 may include a source such as a video capture device, an interface to receive video data from a video content provider, and/or a computer graphics system for generating video data, or a combination of such sources. The video data may comprise one or more pictures. Video encoder 114 encodes the video data from video source 112 to generate a bitstream. The bitstream may include a sequence of bits that form a coded representation of the video data. The bitstream may include coded pictures and associated data. The coded picture is a coded representation of a picture. The associated data may include sequence parameter sets, picture parameter sets, and other syntax structures. I/O interface 116 may include a modulator/demodulator (modem) and/or a transmitter. The encoded video data may be transmitted directly to destination device 120 via I/O interface 116 through network 130a. The encoded video data may also be stored onto a storage medium/server 130b for access by destination device 120.

[0129] Destination device 120 may include an I/O interface 126, a video decoder 124, and a display device 122.

[0130] I/O interface 126 may include a receiver and/or a modem. I/O interface 126 may acquire encoded video data from the source device 110 or the storage medium/ server 130b. Video decoder 124 may decode the encoded video data. Display device 122 may display the decoded video data to a user. Display device 122 may be integrated with the destination device 120, or may be external to destination device 120 which be configured to interface with an external display device.

[0131] Video encoder 114 and video decoder 124 may operate according to a video compression standard, such as the High Efficiency Video Coding (HEVC) standard, Versatile Video Coding(VVM) standard and other current and/or further standards.

[0132] FIG. 16 is a block diagram illustrating an example of video encoder 200, which may be video encoder 114 in the system 100 illustrated in FIG.15.

[0133] Video encoder 200 may be configured to perform any or all of the techniques of this disclosure. In the example of FIG. 16, video encoder 200 includes a plurality of functional components. The techniques described in this disclosure may be shared among the various components of video encoder 200. In some examples, a processor may be configured to perform any or all of the techniques described in this disclosure.

[0134] The functional components of video encoder 200 may include a partition unit 201, a predication unit 202 which may include a mode select unit 203, a motion estimation unit 204, a motion compensation unit 205 and an intra prediction unit 206, a residual generation unit 207, a transform unit 208, a quantization unit 209, an inverse quantization unit 210, an inverse transform unit 211, a reconstruction unit 212, a buffer 213, and an entropy encoding unit 214.

[0135] In other examples, video encoder 200 may include more, fewer, or different functional components. In an example, predication unit 202 may include an intra block copy(IBC) unit. The IBC unit may perform predication in an IBC mode in which at least one reference picture is a picture where the current video block is located.

[0136] Furthermore, some components, such as motion estimation unit 204 and motion compensation unit 205 may be highly integrated, but are represented in the example of FIG. 16 separately for purposes of explanation.

[0137] Partition unit 201 may partition a picture into one or more video blocks. Video encoder 200 and video decoder 300 may support various video block sizes.

[0138] Mode select unit 203 may select one of the coding modes, intra or inter, e.g., based on error results, and provide the resulting intra- or inter-coded block to a residual generation unit 207 to generate residual block data and to a reconstruction unit 212 to reconstruct the encoded block for use as a reference picture. In some example, Mode select unit 203 may select a combination of intra and inter predication (CIIP) mode in which the predication is based on an inter predication signal and an intra predication signal. Mode select unit 203 may also select a resolution for a motion vector (e.g., a sub-pixel or integer pixel precision) for the block in the case of inter-predication.

[0139] To perform inter prediction on a current video block, motion estimation unit 204 may generate motion information for the current video block by comparing one or more reference frames from

buffer 213 to the current video block. Motion compensation unit 205 may determine a predicted video block for the current video block based on the motion information and decoded samples of pictures from buffer 213 other than the picture associated with the current video block.

[0140] Motion estimation unit 204 and motion compensation unit 205 may perform different operations for a current video block, for example, depending on whether the current video block is in an I slice, a P slice, or a B slice.

[0141] In some examples, motion estimation unit 204 may perform uni-directional prediction for the current video block, and motion estimation unit 204 may search reference pictures of list 0 or list 1 for a reference video block for the current video block. Motion estimation unit 204 may then generate a reference index that indicates the reference picture in list 0 or list 1 that contains the reference video block and a motion vector that indicates a spatial displacement between the current video block and the reference video block. Motion estimation unit 204 may output the reference index, a prediction direction indicator, and the motion vector as the motion information of the current video block. Motion compensation unit 205 may generate the predicted video block of the current block based on the reference video block indicated by the motion information of the current video block.

[0142] In other examples, motion estimation unit 204 may perform bi-directional prediction for the current video block, motion estimation unit 204 may search the reference pictures in list 0 for a reference video block for the current video block and may also search the reference pictures in list 1 for another reference video block for the current video block. Motion estimation unit 204 may then generate reference indexes that indicate the reference pictures in list 0 and list 1 containing the reference video blocks and motion vectors that indicate spatial displacements between the reference video blocks and the current video block. Motion estimation unit 204 may output the reference indexes and the motion vectors of the current video block as the motion information of the current video block. Motion compensation unit 205 may generate the predicted video block of the current video block based on the reference video blocks indicated by the motion information of the current video block.

[0143] In some examples, motion estimation unit 204 may output a full set of motion information for decoding processing of a decoder.

[0144] In some examples, motion estimation unit 204 may do not output a full set of motion information for the current video. Rather, motion estimation unit 204 may signal the motion

information of the current video block with reference to the motion information of another video block. For example, motion estimation unit 204 may determine that the motion information of the current video block is sufficiently similar to the motion information of a neighboring video block.

[0145] In one example, motion estimation unit 204 may indicate, in a syntax structure associated with the current video block, a value that indicates to the video decoder 300 that the current video block has the same motion information as the another video block.

[0146] In another example, motion estimation unit 204 may identify, in a syntax structure associated with the current video block, another video block and a motion vector difference (MVD). The motion vector difference indicates a difference between the motion vector of the current video block and the motion vector of the indicated video block. The video decoder 300 may use the motion vector of the indicated video block and the motion vector difference to determine the motion vector of the current video block.

[0147] As discussed above, video encoder 200 may predictively signal the motion vector. Two examples of predictive signaling techniques that may be implemented by video encoder 200 include advanced motion vector predication (AMVP) and merge mode signaling.

[0148] Intra prediction unit 206 may perform intra prediction on the current video block. When intra prediction unit 206 performs intra prediction on the current video block, intra prediction unit 206 may generate prediction data for the current video block based on decoded samples of other video blocks in the same picture. The prediction data for the current video block may include a predicted video block and various syntax elements.

[0149] Residual generation unit 207 may generate residual data for the current video block by subtracting (e.g., indicated by the minus sign) the predicted video block(s) of the current video block from the current video block. The residual data of the current video block may include residual video blocks that correspond to different sample components of the samples in the current video block.

[0150] In other examples, there may be no residual data for the current video block for the current video block, for example in a skip mode, and residual generation unit 207 may not perform the subtracting operation.

[0151] Transform processing unit 208 may generate one or more transform coefficient video blocks for the current video block by applying one or more transforms to a residual video block associated with the current video block.

[0152] After transform processing unit 208 generates a transform coefficient video block associated with the current video block, quantization unit 209 may quantize the transform coefficient video block associated with the current video block based on one or more quantization parameter (QP) values associated with the current video block.

[0153] Inverse quantization unit 210 and inverse transform unit 211 may apply inverse quantization and inverse transforms to the transform coefficient video block, respectively, to reconstruct a residual video block from the transform coefficient video block. Reconstruction unit 212 may add the reconstructed residual video block to corresponding samples from one or more predicted video blocks generated by the predication unit 202 to produce a reconstructed video block associated with the current block for storage in the buffer 213.

[0154] After reconstruction unit 212 reconstructs the video block, loop filtering operation may be performed reduce video blocking artifacts in the video block.

[0155] Entropy encoding unit 214 may receive data from other functional components of the video encoder 200. When entropy encoding unit 214 receives the data, entropy encoding unit 214 may perform one or more entropy encoding operations to generate entropy encoded data and output a bitstream that includes the entropy encoded data.

[0156] FIG. 17 is a block diagram illustrating an example of video decoder 300 which may be video decoder 114 in the system 100 illustrated in FIG.15.

[0157] The video decoder 300 may be configured to perform any or all of the techniques of this disclosure. In the example of FIG. 17, the video decoder 300 includes a plurality of functional components. The techniques described in this disclosure may be shared among the various components of the video decoder 300. In some examples, a processor may be configured to perform any or all of the techniques described in this disclosure.

[0158] In the example of FIG. 17, video decoder 300 includes an entropy decoding unit 301, a motion compensation unit 302, an intra prediction unit 303, an inverse quantization unit 304, an inverse transformation unit 305, and a reconstruction unit 306 and a buffer 307. Video decoder 300 may, in some examples, perform a decoding pass generally reciprocal to the encoding pass described with respect to video encoder 200 (FIG. 16).

[0159] Entropy decoding unit 301 may retrieve an encoded bitstream. The encoded bitstream may include entropy coded video data (e.g., encoded blocks of video data). Entropy decoding unit 301

may decode the entropy coded video data, and from the entropy decoded video data, motion compensation unit 302 may determine motion information including motion vectors, motion vector precision, reference picture list indexes, and other motion information. Motion compensation unit 302 may, for example, determine such information by performing the AMVP and merge mode.

[0160] Motion compensation unit 302 may produce motion compensated blocks, possibly performing interpolation based on interpolation filters. Identifiers for interpolation filters to be used with sub-pixel precision may be included in the syntax elements.

[0161] Motion compensation unit 302 may use interpolation filters as used by video encoder 20 during encoding of the video block to calculate interpolated values for sub-integer pixels of a reference block. Motion compensation unit 302 may determine the interpolation filters used by video encoder 200 according to received syntax information and use the interpolation filters to produce predictive blocks.

[0162] Motion compensation unit 302 may use some of the syntax information to determine sizes of blocks used to encode frame(s) and/or slice(s) of the encoded video sequence, partition information that describes how each macroblock of a picture of the encoded video sequence is partitioned, modes indicating how each partition is encoded, one or more reference frames (and reference frame lists) for each inter-encoded block, and other information to decode the encoded video sequence.

[0163] Intra prediction unit 303 may use intra prediction modes for example received in the bitstream to form a prediction block from spatially adjacent blocks. Inverse quantization unit 303 inverse quantizes, i.e., de-quantizes, the quantized video block coefficients provided in the bitstream and decoded by entropy decoding unit 301. Inverse transform unit 303 applies an inverse transform.

[0164] Reconstruction unit 306 may sum the residual blocks with the corresponding prediction blocks generated by motion compensation unit 202 or intra-prediction unit 303 to form decoded blocks. If desired, a deblocking filter may also be applied to filter the decoded blocks in order to remove blockiness artifacts. The decoded video blocks are then stored in buffer 307, which provides reference blocks for subsequent motion compensation/intra prediction and also produces decoded video for presentation on a display device.

[0165] A listing of clauses preferred by some embodiments is provided next.

[0166] The first set of clauses show example embodiments of techniques discussed in the previous sections.

[0167] 1. A method of video processing, comprising: performing a conversion between a current video block of visual media data and a bitstream representation of the visual media data, wherein, during the conversion, a use of a geometry partition mode is selectively enabled or disabled based at least in part on a determination that that one or more dimensions of the current video block and/or a mathematical function of the one or more dimensions of the current video block achieves at least one threshold condition.

[0168] 2. The method of clause 1, wherein the geometry partition mode includes at least one of: a triangle prediction mode (TPM), a geometric merge mode (GEO), and/or a wedge prediction mode.

[0169] 3. The method of any one or more of clauses 1-2, wherein the geometry partition mode includes splitting a video block into two or more sub-regions, wherein at least one sub-region does not include a QT, BT, and/or a partition.

[0170] 4. The method of any one or more of clauses 1-3, wherein the one or more dimensions of the current video block includes a block width, a block height, and/or an aspect ratio of the current video block.

[0171] 5. The method of any one or more of clauses 1-4, wherein achieving the at least one threshold condition includes the one or more dimensions of the current video block or mathematical functions thereof being greater and/or lesser than corresponding threshold values.

[0172] 6. The method of clause 5, wherein the block width is denoted as W , the block height is denoted as H , wherein the threshold values are denoted as $T1$, $T2$, $T3$, $T4$, and wherein the geometry partition mode is enabled if $W \geq T1$ and/or $H \geq T2$ and/or $W * H < T3$ and/or $W * H > T4$.

[0173] 7. The method of clause 5, wherein the block width is denoted as W , the block height is denoted as H , wherein the threshold values are denoted as $T1$, $T2$, $T3$, $T4$, and wherein the geometry partition mode is enabled if $W \geq T1$ and/or $H \geq T2$ and/or $W * H \leq T3$ and/or $W * H \geq T4$.

[0174] 8. The method of clause 5, wherein the block width is denoted as W , the block height is denoted as H , wherein the threshold values are denoted as $T1$, $T2$, $T3$, $T4$, and wherein the geometry partition mode is enabled if $W * H < T1 \parallel (W * H \leq T2 \ \&\& \ W / H \leq T3 \ \&\& \ H / W \leq T4)$.

[0175] 9. The method of clause 5, wherein the block width is denoted as W , the block height is denoted as H , wherein the threshold values are denoted as $T1$, $T2$, $T3$, $T4$, and wherein the geometry partition mode is enabled if $W * H < T1 \parallel (W * H \leq T2 \ \&\& \ \text{abs}(\log W - \log H) \leq T3)$.

[0176] 10. The method of clause 5, wherein the block width is denoted as W , the block height is denoted as H , wherein the threshold values are denoted as $T1$, $T2$, $T3$, $T4$, and wherein the geometry partition mode is enabled if $W * H \leq T1 \ \&\& \ W/H \leq T2 \ \&\& \ H/W \leq T3$.

[0177] 11. The method of clause 5, wherein the block width is denoted as W , the block height is denoted as H , wherein the threshold values are denoted as T_x , T_y , and wherein the geometry partition mode is enabled if $W \geq T_x$ and $H \geq T_y$.

[0178] 12. The method of clause 5, wherein the block width is denoted as W , the block height is denoted as H , wherein the threshold values are denoted as N , M , and wherein the geometry partition mode is disabled if $W > N$ and/or $H > M$.

[0179] 13. The method of clause 5, wherein the block width is denoted as W , the block height is denoted as H , wherein the threshold values are denoted as T_i ($i=1 \dots 17$), and wherein the geometry partition mode is disabled if one or more below-specified threshold conditions are achieved:

$W < T1$ and/or $W > T2$ and/or $W = T3$

$H < T4$ and/or $H > T5$ and/or $H = T6$

$W * H < T7$ and/or $W * H > T8$ and/or $W * H = T8$

$W/H > T9$ and/or $W/H > T10$ and/or $W/H = T11$

$H/W > T12$ and/or $H/W > T13$ and/or $H/W = T14$

$Abs(\log W - \log H) > T15$ and/or $Abs(\log W - \log H) < T16$ and/or $Abs(\log W - \log H) = T17$.

[0180] 14. The method of clause 5, wherein the block width is denoted as W , the block height is denoted as H , wherein the threshold values are denoted as T_i ($i=1 \dots 17$), and wherein the geometry partition mode is enabled if one or more below-specified threshold conditions are achieved:

$W < T1$ and/or $W > T2$ and/or $W = T3$

$H < T4$ and/or $H > T5$ and/or $H = T6$

$W * H < T7$ and/or $W * H > T8$ and/or $W * H = T8$

$W/H > T9$ and/or $W/H > T10$ and/or $W/H = T11$

$H/W > T12$ and/or $H/W > T13$ and/or $H/W = T14$

$Abs(\log W - \log H) > T15$ and/or $Abs(\log W - \log H) < T16$ and/or $Abs(\log W - \log H) = T17$.

[0181] 15. The method of any one or more of clauses 5-14, wherein the current video block is a luma block.

[0182] 16. The method of any one or more of clauses 5-14, wherein the current video block is a chroma block.

[0183] 17. The method of any one or more of clauses 5-14, wherein the current video block includes a luma component and a chroma component, and wherein, upon determining from the at least one threshold condition that the geometry partition mode is disabled for the luma component, the geometry partition mode is also disabled for the chroma component.

[0184] 18. The method of any one or more of clauses 5-14, wherein the current video block includes a luma component and a chroma component, and wherein, upon determining from the at least one threshold condition that the geometry partition mode is enabled for the luma component, the geometry partition mode is also enabled for the chroma component.

[0185] 19. The method of any one or more of clauses 5-14, wherein the current video block includes a luma component and a chroma component, and wherein the at least one threshold condition is achieved for the luma component and not achieved for the chroma component.

[0186] 20. A method of video processing, comprising: performing a conversion between a current video block of visual media data and a bitstream representation of the visual media data, wherein, during the conversion, a use of multiple sets of geometry partition modes are allowed for the current video block, wherein the multiple sets of geometry partition modes are selected based at least in part on a size of the current video block.

[0187] 21. The method of clause 20, wherein an indication that the multiple sets of geometry partition modes are allowed is included in the bitstream representation.

[0188] 22. The method of clause 20, wherein at least two of the multiple sets of geometry partition modes include a different number of geometry partition modes.

[0189] 23. The method of clause 20, wherein at least two of the multiple sets of geometry partition modes include a same number of geometry partition modes, wherein at least one geometry partition mode included in one set is excluded in another set.

[0190] 24. The method of clause 20, wherein an indication of a total count of the multiple sets of geometry partition modes selected is included in the bitstream representation.

[0191] 25. The method of clause 25, wherein the total count of the multiple sets of geometry partition modes selected is less than a threshold value.

[0192] 26. The method of any one or more of clauses 20-26, wherein geometry partition modes associated with the multiple sets of geometry partition modes that are allowed are identified by a geometry partition mode index, and wherein the geometry partition mode index includes a corresponding partition angle index and/or a corresponding partition distance index of a wedge in connection with the current video block.

[0193] 27. The method of clause 26, wherein a mapping of the geometry partition mode index to a first geometry partition mode is based on determining which of the multiple sets of geometry partition modes is associated with the first geometry partition mode.

[0194] 28. A method of video processing, comprising: performing a conversion between video blocks of visual media data and a bitstream representation of the visual media data, wherein, during the conversion, a first count of geometry partition modes are used for computing a partition angle index and/or a partition distance index of a first video block, a second count of geometry partition modes are used in the bitstream representation of a second video block, and a third count of geometry partition modes are signaled in the bitstream representation of a third video block, wherein the first count and/or the second count and/or the third count are based at least on corresponding dimensions of the first, second, and third video blocks.

[0195] 29. The method of clause 28, wherein the second count and/or the third count are different from the first count.

[0196] 30. The method of clause 28, wherein the second count equals the third count.

[0197] 31. The method of clause 28, wherein the second count and/or the third count are smaller than the first count.

[0198] 32. The method of clause 28, wherein the first second, and third video blocks are associated with first, second, and third categories of video blocks.

[0199] 33. The method of clause 32, wherein the first, second, and third categories of video blocks are different, and wherein the first, second, and third categories of video blocks are associated with different dimensions.

[0200] 34. The method of clause 28, wherein the second count and/or the third count are smaller than the first count when the dimensions of the first block meet one or more threshold conditions.

[0201] 35. A method of video processing, comprising: performing a conversion between a current video block of visual media data and a bitstream representation of the visual media data, wherein,

during the conversion, a first geometry partition mode index value is signaled in the bitstream representation of the current video block and a second geometry partition mode index value is used for computing a partition angle index and/or a partition distance index of the current video block, and wherein the first geometry partition mode index value is different from the second geometry partition mode index value.

[0202] 36. The method of clause 35, wherein at least one mapping table defines the relationship between the first geometry partition mode index value and the second geometry partition mode index value.

[0203] 37. The method of clause 36, wherein the at least one mapping table includes a first mapping table and a second mapping table, and wherein the first mapping table is associated with a video block of a first type and a second mapping table is associated with a video block of a second type.

[0204] 38. A method of video processing, comprising: performing a conversion between a current video block of visual media data and a bitstream representation of the visual media data, wherein, during the conversion, a use of a geometry partition mode is allowed for the current video block, and wherein parameters of the geometry partition mode are computed using a reduced set of angles and/or a reduced set of distances.

[0205] 39. The method of clause 38, wherein a count of the reduced set of angles are less than a threshold value, and wherein the threshold value is 24.

[0206] 40. The method of clause 38, wherein a count of the reduced set of distances are less than a threshold value, and wherein the threshold value is 82.

[0207] 41. The method of clause 38, wherein a lookup table is used in computing the reduced set of distances, and wherein a size of the lookup table is based at least in part on the reduced set of angles.

[0208] 42. A video decoding apparatus comprising a processor configured to implement a method recited in one or more of clauses 1 to 41.

[0209] 43. A video encoding apparatus comprising a processor configured to implement a method recited in one or more of clauses 1 to 41.

[0210] 44. A computer program product having computer code stored thereon, the code, when executed by a processor, causes the processor to implement a method recited in any of clauses 1 to 44.

[0211] 45. A method, apparatus or system described in the present document.

[0212] The second set of clauses describe certain features and aspects of the disclosed techniques in the previous section.

[0213] 1. A method of video processing (e.g. method 1810 shown in FIG. 18A), comprising: performing 1812 a conversion between a current video block of a video and a bitstream representation of the video, wherein, during the conversion, a use of a geometric partitioning mode is allowed for the current video block, and wherein parameters of the geometric partitioning mode are computed using a set of angles including a first number of angles that is less than a first threshold value and/or a set of distances including a second number of distances that is less than a second threshold value.

[0214] 2. The method of clause 1, wherein the first threshold value is 24.

[0215] 3. The method of clause 1, wherein the geometric partitioning mode is selected from a set of geometric partitioning modes, and wherein the set of geometric partitioning modes includes a number of modes that is less than a third threshold value.

[0216] 4. The method of clause 3, wherein the third threshold value is 82.

[0217] 5. The method of clause 1, wherein the second threshold is 3 or 4.

[0218] 6. The method of clause 1, wherein a number of distances for one or more angles is less than the second threshold value.

[0219] 7. The method of clause 1, wherein a number of distances for vertical and horizontal angles is equal to X, whereby X is a positive integer.

[0220] 8. The method of clause 1, wherein X is 2.

[0221] 9. The method of clause 1, wherein a number of angles used in a decoding process for the current video block is equal to a sum of 1 and a maximum value of a variable of angleIdx, whereby the angleIdx specifying an angle index of the geometric partitioning mode.

[0222] 10. The method of clause 1, wherein a calculation of a variable of displacementY used in a processes of weighted sample prediction and/or motion vector storing for the current video block depends on a total number of angles used in a decoding process, whereby the displacement Y is set to $(\text{displacementX} + (\text{NUM_ANGLE} > 2)) \% \text{NUM_ANGLE}$, the displacementX is set to angleIdx

specifying an angle index of the geometric partitioning mode, and the NUM_ANGLE specifies a number of angles used in a decoding process for the current video block.

[0223] 11. The method of clause 1, wherein a calculation of a variable of shiftHor used in a processes of weighted sample prediction and/or motion vector storing for the current video block depends on a total number of angles used in a decoding process, whereby the shiftHor is set to 0 or 1.

[0224] 12. The method of clause 11, wherein the shiftHor is set to 0 in a case that at least one of following conditions is met:

- 1) $\text{angleIdx} \% (\text{NUM_ANGLE}/2)$ is equal to $(\text{NUM_ANGLE} >> 2)$
- 2) $\text{angleIdx} \% (\text{NUM_ANGLE}/2)$ is not equal to 0 and $\text{hwRatio} \geq 1$, wherein hwRatio is set to H/W ,

whereby angleIdx specifies an angle index of the geometric partitioning mode, and the NUM_ANGLE specifies a number of angles used in a decoding process for the current video block, and H and W are height and width of the current video block, respectively.

[0225] 13. The method of clause 11, wherein a derivation process of offset values for deriving a blending weight index of the current video block is dependent on a number of angles and/or a value of the shiftHor.

[0226] 14. The method of clause 11, wherein a derivation process of offset values for deriving a motion index of the current video block is dependent on a number of angles and/or a value of the shiftHor.

[0227] 15. The method of clause 11, wherein a lookup table is used in computing the set of distances, and wherein a size of the lookup table is based at least in part on the set of angles.

[0228] 16. The method of clause 15, wherein values of the lookup table are determined based on the size of the lookup table.

[0229] 17. The method of clause 1, wherein in a case that A and B indicate two input arrays, predSamplesLA and predSamplesLB, used for a weighted sample prediction process for the geometric partitioning mode and that PART1 and PART2 are representations of A and B for a derivation of output weighted prediction sample values of a predicted block of the current video block, whether PART1 and PART2 are equal to A or B depends on angel indices, T1 and T2.

[0230] 18. The method of clause 1, wherein a variable of a partIdx used in a motion vector storing process for the geometric partitioning mode is set to 0 or 1 based on angle indices, T1 and T2, where

the partIdx is used to derive another variable of sType for assigning motion vectors for a motion storage.

[0231] 19. A method of video processing (e.g., method 1820 shown in FIG. 18B), comprising: determining 1822, for a conversion between a current video block of a video and a bitstream representation of the video, an applicability of a geometric partitioning mode based on a rule that depends on one or more dimensions of the current video block and/or a mathematical function of the one or more dimensions of the current video; and performing 1824 the conversion based on the determining.

[0232] 20. The method of clause 19, wherein the geometric partitioning mode includes at least one of: a triangle prediction mode (TPM), a geometric merge mode (GEO), and/or a wedge prediction mode.

[0233] 21. The method of clause 19 or 20, wherein the geometric partitioning mode includes splitting the current video block into two or more sub-regions, wherein at least one sub-region does not include a QT, BT, and/or a partition.

[0234] 22. The method of any one or more of clauses 19 to 21, wherein the rule depends on a block width, a block height, and/or an aspect ratio of the current video block.

[0235] 23. The method of any one or more of clauses 19 to 22, wherein the rule specifies that the geometric partitioning mode is applied in a case that $W \geq T1$ and/or $H \geq T2$ and/or $W * H < T3$ and/or $W * H > T4$, whereby a width and a height of the current video block are denoted as W and H, respectively, and T1, T2, T3 and T4 are constant values.

[0236] 24. The method of any one or more of clauses 19 to 22, wherein the rule specifies that the geometric partitioning mode is applied in a case that $W \geq T1$ and/or $H \geq T2$ and/or $W * H \leq T3$ and/or $W * H \geq T4$, whereby a width and a height of the current video block are denoted as W and H, respectively, and T1, T2, T3 and T4 are constant values.

[0237] 25. The method of any one or more of clauses 19 to 22, wherein the rule specifies that the geometric partitioning mode is applied in a case that $W * H < T1 \parallel (W * H \leq T2 \ \&\& \ W/H < T3 \ \&\& \ HW \leq T4)$, whereby a width and a height of the current video block are denoted as W and H, respectively, and T1, T2, T3 and T4 are constant values.

[0238] 26. The method of any one or more of clauses 19 to 22, wherein the rule specifies that the geometric partitioning mode is applied in a case that $W * H < T1 \parallel (W * H \leq T2 \ \&\& \ \text{abs}(\log W - \log H))$

$\leq T3$), whereby a width and a height of the current video block are denoted as W and H , respectively, and $T1$, $T2$ and $T3$ are constant values.

[0239] 27. The method of any one or more of clauses 19 to 22, wherein the rule specifies that the geometric partitioning mode is applied in a case that $W \cdot H \leq T1$ && $W/H \leq T2$ && $H/W \leq T3$, whereby a width and a height of the current video block are denoted as W and H , respectively, and $T1$, $T2$ and $T3$ are constant values.

[0240] 28. The method of any one or more of clauses 19 to 22, wherein the rule specifies that the geometric partitioning mode is applied in a case that $W \geq Tx$ and $H \geq Ty$, whereby a width and a height of the current video block are denoted as W and H , respectively, and Tx , Ty are constant values.

[0241] 29. The method of any one or more of clauses 19 to 22, wherein the rule specifies that the geometric partitioning mode is not applied in a case that $W > N$ and/or $H > M$, whereby a width and a height of the current video block are denoted as W and H , respectively, and N , M are constant values.

[0242] 30. The method of any one or more of clauses 19 to 22, wherein the rule specifies that the geometric partitioning mode is not applied in a case that $W = N$ and/or $H = M$, whereby a width and a height of the current video block are denoted as W and H , respectively, and N , M are constant values.

[0243] 31. The method of any one or more of clauses 19 to 22, wherein the rule specifies that the geometric partitioning mode is not applied in a case that one or more following conditions are satisfied, whereby a width and a height of the block are denoted as W and H , respectively, and Ti ($i=1 \dots 17$) are constant values:

- a) $W < T1$ and/or $W > T2$ and/or $W = T3$
- b) $H < T4$ and/or $H > T5$ and/or $H = T6$
- c) $W * H < T7$ and/or $W * H > T8$ and/or $W * H = T8$
- d) $W/H > T9$ and/or $W/H > T10$ and/or $W/H = T11$
- e) $H/W > T12$ and/or $H/W > T13$ and/or $H/W = T14$
- f) $Abs(\log W - \log H) > T15$ and/or $Abs(\log W - \log H) < T16$ and/or $Abs(\log W - \log H) = T17$.

[0244] 32. The method of any one or more of clauses 19 to 22, wherein the rule specifies that the geometric partitioning mode is applied in a case that one or more following conditions are satisfied, whereby a width and a height of the block are denoted as W and H , respectively, and Ti ($i=1 \dots 17$) are constant values:

- a) $W < T1$ and/or $W > T2$ and/or $W = T3$
- b) $H < T4$ and/or $H > T5$ and/or $H = T6$
- c) $W * H < T7$ and/or $W * H > T8$ and/or $W * H = T8$
- d) $W/H > T9$ and/or $W/H > T10$ and/or $W/H = T11$
- e) $H/W > T12$ and/or $H/W > T13$ and/or $H/W = T14$
- f) $Abs(\log W - \log H) > T15$ and/or $Abs(\log W - \log H) < T16$ and/or $Abs(\log W - \log H) = T17$.

[0245] 33. The method of any one or more of clauses 19 to 22, wherein the rule depends on the mathematical function that depends on ratios of a width (W) and/or a height (H) of the current video block.

[0246] 34. The method of clause 33, wherein 33, wherein the mathematical function is $\max(H, W)/\min(H, W)$.

[0247] 35. The method of any one or more of clauses 19 to 22, wherein the mathematical function relates to differences and/or ratios between a width (W) and a height (H) of the current video block.

[0248] 36. The method of any one or more of clauses 19 to 22, wherein the rule specifies that the geometric partitioning mode is not applied for the current video block having a width (W) and a height (H) in a case that a width to height ratio (W/H) or a height to width ratio (H/W) is greater than or no less than X, whereby X is a constant value.

[0249] 37. The method of any one or more of clauses 19 to 22, wherein the current video block is a luma block or a chroma block.

[0250] 38. The method of any one or more of clauses 19 to 22, wherein the current video block includes a luma component and a chroma component and wherein the rule specifies that the geometric partitioning mode is applied for the luma component but not applied for the chroma block.

[0251] 39. The method of any one or more of clauses 19 to 22, wherein the current video block corresponds to a luma block in a video region and an applicability of the geometric partitioning mode to a chroma block in the video region depends on the one more dimensions of the current video block.

[0252] 40. A method of video processing (e.g., method 1830 as shown in FIG. 18C), comprising: determining 1832, for a conversion between a current video block of a video and a bitstream representation of the video, an applicability of a geometric partitioning mode according to a rule that

depends on coding characteristics of the current video block; and performing 1834 the conversion based on the determining.

[0253] 41. The method of clause 40, wherein the coding characteristics include at least one of a maximum transform size, a maximum allowed coding unit (CU) size, or a chroma format.

[0254] 42. The method of clause 41, wherein the rule specifies that the geometric partitioning mode is not applied for the current video block that has a width and/or a height greater than the maximum transform size.

[0255] 43. The method of clause 41, wherein the rule specifies that the geometric partitioning mode is not applied for the current video block that has a width and/or a height equal to the maximum allowed CU size.

[0256] 44. The method of clause 41, wherein the rule specifies that the geometric partitioning mode is not applied for the current video block having a 4:0:0 chroma format, 4:4:4 chroma format or 4:2:2 chroma format.

[0257] 45. The method of clause 41, wherein the rule specifies that the geometric partitioning mode is not applied for the current video block having a certain color component with a certain chroma format.

[0258] 46. A method of video processing (e.g., method 1840 as shown in FIG. 18D), comprising: determining that a rule of exclusion is applicable to a conversion between a current video block of a video and a bitstream representation of the video, wherein the rule of exclusion specifies that the conversion disallows using a geometric partitioning mode and a coding tool together for the current video block; and performing the conversion based on the determining.

[0259] 47. The method of clause 46, wherein the bitstream representation omits an indication of usage of the coding tool and/or information of the coding tool in a case that the geometric partitioning mode is applied to the current video block.

[0260] 48. The method of clause 46, wherein the bitstream representation omits an indication of usage of the geometric partitioning mode and/or information of the geometric partitioning mode in a case that the coding tool is applied to the current video block.

[0261] 49. The method of any of clauses 46 to 48, wherein the coding tool includes an adaptive color transform, a dual tree coding mode, a transform skip mode, a block differential pulse-code modulation (BDPCM) coding mode, or a sub-block transform (SBT).

[0262] 50. A method of video processing (e.g., method 1850 as shown in FIG. 18E), comprising: determining 1852, for a conversion between a video and a bitstream representation of the video, different geometric partitioning mode indices for different color components of the video according to a rule; and performing 1854 the conversion based on the determining.

[0263] 51. The method of clause 50, wherein the rule specifies that a chroma component of the video has a different geometric partitioning mode index from that of a luma component of the video.

[0264] 52. The method of clause 50, wherein the geometric partitioning mode is not applied to chroma components of the video.

[0265] 53. The method of clause 50, wherein the bitstream representation includes the different geometric partitioning mode indices.

[0266] 54. The method of clause 53, wherein the different geometric partitioning mode indices include a mode index for a luma component of the video and a mode index for chroma components of the video.

[0267] 55. The method of clause 50, wherein the bitstream representation includes a geometric partitioning mode index of a second color component and wherein a geometric partitioning mode index of a first color component is predicted from the mode index of the second color component.

[0268] 56. A method of video processing (e.g. method 1860 as shown in FIG. 18F), comprising: determining 1862, for a conversion between a current video block of a current picture of a video and a bitstream representation of the video, an applicability of a geometric partitioning mode according to a rule; and performing 1864 the conversion based on the determining, wherein the geometric partitioning mode includes splitting the current video block into two or more prediction sub-regions, and wherein the rule depends on a resolution of one or more reference pictures associated with the two or more prediction sub-regions.

[0269] 57. The method of clause 56, wherein the rule specifies that the geometric partitioning mode is not applied in a case that the resolutions of reference pictures associated with the two or more prediction sub-regions are different from each other.

[0270] 58. The method of clause 56, wherein the rule specifies that the geometric partitioning mode is not applied in a case that a resolution of a reference picture associated with a prediction sub-region is different from a resolution of the current picture.

[0271] 59. The method of clause 56, wherein the rule specifies that the geometric partitioning mode is applied in a case that a resolution of a reference picture associated with a prediction sub-region is different from a resolution of the current picture.

[0272] 60. The method of any of clauses 56 to 59, wherein a resolution of the current picture refers to a width or a height of the current picture or refers to a window in the picture.

[0273] 61. A method of video processing, comprising: performing a conversion between a current video block of a video and a bitstream representation of the video, wherein the bitstream representation conforms to a format rule, wherein the format rule specifies that the bitstream representation omits a syntax element related to a geometric partitioning mode in a case that the geometric partitioning mode is disabled for the current video block.

[0274] 62. The method of clause 61, wherein the syntax element includes at least one of `wedge_partition_idx`, `merge_wedge_idx0`, or `merge_wedge_idx1`.

[0275] 63. The method of clause 61, wherein the syntax element that is not included in the bitstream representation is inferred to be a default value.

[0276] 64. The method of clause 61, wherein semantic variables related to the geometric partitioning mode is inferred to be a default value in a case that the geometric partitioning mode is disabled or not allowed.

[0277] 65. A method of video processing (e.g., method 1870 as shown in FIG. 18G), comprising: determining 1872, for a conversion between a video unit comprising one or more video blocks of a video and a bitstream representation of the video, that multiple sets of geometric partitioning modes are allowed for coding the one or more video blocks of the video unit according to a rule; and performing 1874 the conversion based on the determining.

[0278] 66. The method of clause 65, wherein the rule specifies that a selection of a set from the multiple sets is dependent on decoded information.

[0279] 67. The method of clause 66, wherein the decoded information includes a dimension and/or a shape of a video block.

[0280] 68. The method of clause 65, wherein the rule specifies that at least two of the multiple sets have different numbers of allowed geometric partitioning modes.

[0281] 69. The method of clause 65, wherein the rule specifies that T sets among the multiple sets are with a same number of allowed geometric partitioning modes but at least one geometric partitioning mode included in one of the T sets is excluded in another one of the T sets.

[0282] 70. The method of clause 65, wherein the rule specifies that T sets among the multiple sets are with a same geometric partitioning modes but at least one geometric partitioning mode is arranged in a different position for any two of the T sets.

[0283] 71. The method of clause 65, wherein how to signal a geometric partitioning mode index depends on a corresponding set of allowed geometric partitioning modes.

[0284] 72. The method of clause 65, wherein a decoded geometric partitioning mode index depends on a corresponding geometric partitioning mode.

[0285] 73. The method of clause 65, wherein a total count of the geometric partitioning modes that are to be used for a video block in the bitstream representation is defined as a number that is less than a constant value.

[0286] 74. The method of clause 65, wherein a total count of the geometric partitioning modes that are to be signaled for a video block in the bitstream representation is defined as a number that is less than a constant value.

[0287] 75. The method of clause 73 or 74, wherein the number is signaled.

[0288] 76. A method of video processing, comprising: performing a conversion between a video including video blocks and a bitstream representation of the video, wherein, during the conversion, a first count of geometric partitioning modes are available for computing a partition angle index and/or a partition distance index, a second count of geometric partitioning modes are available for coding in the bitstream representation, and a third count of geometric partitioning modes are available for signaling in the bitstream representation of, wherein the first count and/or the second count and/or the third count are based at least on corresponding dimensions of the video blocks.

[0289] 77. The method of clause 76, wherein the second count and/or and the third count are different from the first count.

[0290] 78. The method of clause 76, wherein the second count equals the third count.

[0291] 79. The method of clause 76, wherein the second count and/or and the third count are smaller than the first count.

- [0292] 80. The method of clause 76, wherein the second count and/or the third count is defined differently for different block categories of the video blocks.
- [0293] 81. The method of clause 80, wherein a category of a video block is based on ratios of a width and a height of the video block, a mathematical function of the width and the height of the video block, or a dimension of the video block.
- [0294] 82. The method of clause 81, wherein the second count and/or the third count is smaller than the first count when the dimension of the first block meet one or more threshold conditions.
- [0295] 83. The method of clause 80, wherein a set of fixed numbers is defined for the second count and/or the third count for each category of the video block.
- [0296] 84. The method of clause 80, wherein the second count and/or the third count for each category of the video block is included in the bitstream representation.
- [0297] 85. The method of clause 80, wherein the second count and/or the third count for each category of the video block is predefined.
- [0298] 86. The method of clause 80, wherein a width and/or a height of a luma block is used for deriving the second count and/or the third count.
- [0299] 87. A method of video processing, comprising: performing a conversion between a current video block of a video and a bitstream representation of the video according to a rule, wherein the rule specifies that a first geometric partitioning mode index value is signaled in the bitstream representation of the current video block and a second geometric partitioning mode index value is used for computing a partition angle index and/or a partition distance index of the current video block, and wherein the first geometric partitioning mode index value is different from the second geometric partitioning mode index value.
- [0300] 88. The method of clause 87, wherein the current video block is associated with a certain category based on a width and/or a height of the current video block and wherein a subset of a full set of geometric partitioning modes, partition angles, and/or partition distances is used for the certain category.
- [0301] 89. The method of clause 87, wherein at least one mapping table defines a relationship between the first geometric partitioning mode index value and the second geometric partitioning mode index value.

- [0302] 90. The method of clause 89, wherein the at least one mapping table is included in the bitstream representation.
- [0303] 91. The method of clause 89, wherein the at least one mapping table is predefined.
- [0304] 92. The method of clause 87, wherein N mapping tables are used for the conversion and wherein N is a constant value depending on a number of block categories of the video blocks.
- [0305] 93. The method of clause 92, wherein lengths of the N mapping tables depend on the block categories of the video blocks.
- [0306] 94. A method of video processing, comprising: performing a conversion between a current video block of a video and a bitstream representation of the video, wherein a geometric partitioning mode index of the current video block is coded in the bitstream such that a binarization of the geometric partitioning mode index is performed according to a rule.
- [0307] 95. The method of clause 94, wherein the rule specifies that the binarization dependent on decoded information of the current video block.
- [0308] 96. The method of clause 95, wherein the decoded information includes a dimension of the current video block and/or a category of the current video block.
- [0309] 97. The method of clause 94, wherein the rule specifies that a truncated binary code is used for the binarization.
- [0310] 98. The method of clause 97, wherein the geometric partitioning mode index included in the bitstream representation is different from a derived geometric partitioning mode index that is used for deriving a partition angle index and/or a partition distance index.
- [0311] 99. The method of clause 94, wherein the rule specifies that a Kth order Exp-Golomb (EG) coding is used for the binarization, whereby K is an integer.
- [0312] 100. The method of clause 94, wherein the rule specifies that a context coding is used to code the geometric partitioning mode index.
- [0313] 101. The method of clause 94, wherein the rule specifies that first X bins of the geometric partitioning mode index are coded by a context coding and a remaining bin is coded by a by-pass coding without a context modeling.
- [0314] 102. A method of video processing (e.g. method 1880 as shown in FIG. 18H), comprising: determining 1882, for a conversion between a chroma block of a video unit of a video and a bitstream representation of the video, blending weights and/or motion storage weights for the chroma block

according to a rule; and performing 1884 the conversion based on the determining, wherein a geometric partitioning mode is applied to the chroma block, and wherein the rule depends on a chroma sample location type indicating a relative position of a certain chroma sample in the chroma block with respect to a corresponding luma sample.

[0315] 103. The method of clause 102, wherein the video unit includes a sequence parameter set (SPS), a video parameter set (VPS), a picture parameter set (PPS), a picture header, a subpicture, a slice, a slice header, a tile, a brick, a coding tree unit, or a virtual pipeline data unit (VPDU).

[0316] 104. The method of clause 102, wherein a type of a downsampling filter used for a blending weights derivation for chroma samples is signaled at a video unit level.

[0317] 105. The method of clause 104, wherein the bitstream representation includes a flag to switch between different chroma location types.

[0318] 106. The method of clause 105, wherein the flag specifies whether a top-left downsampled luma sample in a geometric partitioning mode is collocated with a top-left luma sample or the flag specifies whether the top-left downsampled luma sample is horizontally co-sited with the top-left luma sample but vertically shifted relatively to the top-left luma sample.

[0319] 107. The method of clause 104, wherein the type of the downsampling filter is signaled for the chroma block having 4:2:0 chroma format or 4:2:2 chroma format.

[0320] 108. The method of clause 104, wherein the bitstream representation includes a flag for specifying a type of a chroma downsampling filter used for a geometric partitioning prediction.

[0321] 109. The method of clause 102, wherein a type of downsampling filter used for a blending weights derivation for chroma samples is signaled at a video unit level.

[0322] 110. The method of clause 109, wherein a look up table is defined to specify a corresponding relationship between a chroma subsampling filter type and a chroma format type.

[0323] 111. The method of clause 102, wherein a downsampling filter used for a geometric partitioning prediction of the chroma block depends on the chroma sample location type.

[0324] 112. The method of clause 111, wherein chroma weights for the chroma block are subsampled from collocated top-left luma weights in a case that the chroma block has a certain chroma sample location type.

[0325] 113. The method of clause 111, wherein a specified X-tap filter is used for chroma weights subsampling in a case that the chroma block has a certain chroma sample location type.

[0326] 114. The method of any of clauses 1 to 113, wherein the geometric partitioning mode is selected from a set of geometric partitioning modes, and wherein the set of geometric partitioning modes includes one or more geometric partitioning modes divide a block into two partitions, at least one of which is non-square and non-rectangular.

[0327] 115. The method of any of clauses 1 to 114, wherein the conversion includes encoding the video into the bitstream representation.

[0328] 116. The method of any of clauses 1 to 114, wherein the conversion includes decoding the video from the bitstream representation.

[0329] 117. A video processing apparatus comprising a processor configured to implement a method recited in any one or more of clauses 1 to 116.

[0330] 118. A computer readable medium storing program code that, when executed, causes a processor to implement a method recited in any one or more of clauses 1 to 116.

[0331] 119. A computer readable medium that stores a coded representation or a bitstream representation generated according to any of the above described methods.

[0332] The disclosed and other solutions, examples, embodiments, modules and the functional operations described in this document can be implemented in digital electronic circuitry, or in computer software, firmware, or hardware, including the structures disclosed in this document and their structural equivalents, or in combinations of one or more of them. The disclosed and other embodiments can be implemented as one or more computer program products, i.e., one or more modules of computer program instructions encoded on a computer readable medium for execution by, or to control the operation of, data processing apparatus. The computer readable medium can be a machine-readable storage device, a machine-readable storage substrate, a memory device, a composition of matter effecting a machine-readable propagated signal, or a combination of one or more them. The term "data processing apparatus" encompasses all apparatus, devices, and machines for processing data, including by way of example a programmable processor, a computer, or multiple processors or computers. The apparatus can include, in addition to hardware, code that creates an execution environment for the computer program in question, e.g., code that constitutes processor firmware, a protocol stack, a database management system, an operating system, or a combination of one or more of them. A propagated signal is an artificially generated signal, e.g., a machine-generated

electrical, optical, or electromagnetic signal, that is generated to encode information for transmission to suitable receiver apparatus.

[0333] A computer program (also known as a program, software, software application, script, or code) can be written in any form of programming language, including compiled or interpreted languages, and it can be deployed in any form, including as a stand-alone program or as a module, component, subroutine, or other unit suitable for use in a computing environment. A computer program does not necessarily correspond to a file in a file system. A program can be stored in a portion of a file that holds other programs or data (e.g., one or more scripts stored in a markup language document), in a single file dedicated to the program in question, or in multiple coordinated files (e.g., files that store one or more modules, sub programs, or portions of code). A computer program can be deployed to be executed on one computer or on multiple computers that are located at one site or distributed across multiple sites and interconnected by a communication network.

[0334] The processes and logic flows described in this document can be performed by one or more programmable processors executing one or more computer programs to perform functions by operating on input data and generating output. The processes and logic flows can also be performed by, and apparatus can also be implemented as, special purpose logic circuitry, e.g., an FPGA (field programmable gate array) or an ASIC (application specific integrated circuit).

[0335] Processors suitable for the execution of a computer program include, by way of example, both general and special purpose microprocessors, and any one or more processors of any kind of digital computer. Generally, a processor will receive instructions and data from a read only memory or a random-access memory or both. The essential elements of a computer are a processor for performing instructions and one or more memory devices for storing instructions and data. Generally, a computer will also include, or be operatively coupled to receive data from or transfer data to, or both, one or more mass storage devices for storing data, e.g., magnetic, magneto optical disks, or optical disks. However, a computer need not have such devices. Computer readable media suitable for storing computer program instructions and data include all forms of non-volatile memory, media and memory devices, including by way of example semiconductor memory devices, e.g., EPROM, EEPROM, and flash memory devices; magnetic disks, e.g., internal hard disks or removable disks; magneto optical disks; and CD ROM and DVD-ROM disks. The processor and the memory can be supplemented by, or incorporated in, special purpose logic circuitry.

[0336] While this patent document contains many specifics, these should not be construed as limitations on the scope of any subject matter or of what may be claimed, but rather as descriptions of features that may be specific to particular embodiments of particular techniques. Certain features that are described in this patent document in the context of separate embodiments can also be implemented in combination in a single embodiment. Conversely, various features that are described in the context of a single embodiment can also be implemented in multiple embodiments separately or in any suitable subcombination. Moreover, although features may be described above as acting in certain combinations and even initially claimed as such, one or more features from a claimed combination can in some cases be excised from the combination, and the claimed combination may be directed to a subcombination or variation of a subcombination.

[0337] Similarly, while operations are depicted in the drawings in a particular order, this should not be understood as requiring that such operations be performed in the particular order shown or in sequential order, or that all illustrated operations be performed, to achieve desirable results. Moreover, the separation of various system components in the embodiments described in this patent document should not be understood as requiring such separation in all embodiments.

[0338] Only a few implementations and examples are described and other implementations, enhancements and variations can be made based on what is described and illustrated in this patent document.