

MOTION PRECISION IN SUB-BLOCK BASED INTER PREDICTION

CROSS-REFERENCE TO RELATED APPLICATIONS

[001] 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/CN2019/100396, filed on August 13, 2019, and International Patent Application No. PCT/CN2019/107159, filed on September 22, 2019. 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

[002] This patent document relates to image and video coding and decoding.

BACKGROUND

[003] In spite of the advances in video compression, digital video still accounts for the largest bandwidth use on the internet and other digital communication networks. As the number of connected user devices capable of receiving and displaying video increases, it is expected that the bandwidth demand for digital video usage will continue to grow.

SUMMARY

[004] Devices, systems and methods related to digital video coding, which include sub-block based inter prediction methods, are described. The described methods may be applied to both the existing video coding standards (e.g., High Efficiency Video Coding (HEVC) and/or Versatile Video Coding (VVC)) and future video coding standards or video codecs.

[005] In one representative aspect, the disclosed technology may be used to provide a method for video processing. This method includes determining, for a conversion between a current block of video and a bitstream representation of the video, a maximum number of candidates (ML) in a sub-block based merge candidate list and/or whether to add sub-block based temporal motion vector prediction (SbTMVP) candidates to the sub-block based merge candidate list based on whether temporal motion vector prediction (TMVP) is enabled for use during the

conversion or whether a current picture referencing (CPR) coding mode is used for the conversion; and performing, based on the determining, the conversion.

[006] In another representative aspect, the disclosed technology may be used to provide a method for video processing. This method includes determining, for a conversion between a current block of video and a bitstream representation of the video, a maximum number of candidates (ML) in a sub-block based merge candidate list based on whether a temporal motion vector prediction (TMVP), a sub-block based temporal motion vector prediction (SbTMVP), and an affine coding mode are enabled for use during the conversion; and performing, based on the determining, the conversion.

[007] In yet another representative aspect, the disclosed technology may be used to provide a method for video processing. This method includes determining, for a conversion between a current block of a first video segment of a video and a bitstream representation of the video, that a sub-block based motion vector prediction (SbTMVP) mode is disabled for the conversion due to a temporal motion vector prediction (TMVP) mode being disabled at a first video segment level; and performing the conversion based on the determining, wherein the bitstream representation complies with a format that specifies whether an indication of the SbTMVP mode is included and/or a position of the indication of the SbTMVP mode in a merge candidate list, with respect to an indication of the TMVP mode.

[008] In yet another representative aspect, the disclosed technology may be used to provide a method for video processing. This method includes performing a conversion between a current block of a video that is coded using a sub-block based temporal motion vector prediction (SbTMVP) tool or a temporal motion vector prediction (TMVP) tool and a bitstream representation of the video, wherein a coordinate of a corresponding position of the current block or a sub-block of the current block is selectively masked using a mask based on a compression of motion vectors associated with the SbTMVP tool or the TMVP tool, and wherein an application of the mask comprises computing a bitwise AND operation between a value of the coordinate and a value of the mask.

[009] In yet another representative aspect, the disclosed technology may be used to provide a method for video processing. This method includes determining, based on one or more characteristics of a current block of a video segment of a video, a valid corresponding region of the current block for an application of a sub-block based motion vector prediction (SbTMVP)

tool on the current block; and performing, based on the determining, a conversion between the current block and a bitstream representation of the video.

[0010] In yet another representative aspect, the disclosed technology may be used to provide a method for video processing. This method includes determining, for a current block of a video that is coded using a sub-block based temporal motion vector prediction (SbTMVP) tool, a default motion vector; and performing, based on the determining, a conversion between the current block and a bitstream representation of the video, wherein the default motion vector is determined in case that a motion vector is not obtained from a block covering a corresponding position in the collocated picture that is associated with a center position of the current block.

[0011] In yet another representative aspect, the disclosed technology may be used to provide a method for video processing. This method includes inferring, for a current block of a video segment of a video, that a sub-block based temporal motion vector prediction (SbTMVP) tool or a temporal motion vector prediction (TMVP) tool is disabled for the video segment in case that a current picture of the current block is a reference picture with an index set to M in a reference picture list X, wherein M and X are integers, and wherein $X = 0$ or $X = 1$; and performing, based on the inferring, a conversion between the current block and a bitstream representation of the video.

[0012] In yet another representative aspect, the disclosed technology may be used to provide a method for video processing. This method includes determining, for a current block of a video, that an application of an sub-block based temporal motion vector prediction (SbTMVP) tool is enabled in case that a current picture of the current block is a reference picture with an index set to M in a reference picture list X, wherein M and X are integers; and performing, based on the determining, a conversion between the current block and a bitstream representation of the video.

[0013] In yet another representative aspect, the disclosed technology may be used to provide a method for video processing. This method includes performing a conversion between a current block of a video and a bitstream representation of the video, wherein the current block is coded using a sub-block based coding tool, and wherein performing the conversion comprises using a plurality of bins (N) to code a sub-block merge index with a unified method in case that a sub-block based temporal motion vector prediction (SbTMVP) tool is enabled or disabled.

[0014] In yet another representative aspect, the disclosed technology may be used to provide a method for video processing. This method includes determining, for a current block of a video

coded using a subblock-based temporal motion vector prediction (SbTMVP) tool, a motion vector used by the SbTMVP tool to locate a corresponding block in a picture different from a current picture comprising the current block; and performing, based on the determining, a conversion between the current block and a bitstream representation of the video.

[0015] In yet another representative aspect, the disclosed technology may be used to provide a method for video processing. This method includes determining, for a conversion between a current block of a video and a bitstream representation of the video, whether a zero motion affine merge candidate is inserted into a sub-block merge candidate list based on whether affine prediction is enabled for the conversion of the current block; and performing, based on the determining, the conversion.

[0016] In yet another representative aspect, the disclosed technology may be used to provide a method for video processing. This method includes inserting, for a conversion between a current block of a video and a bitstream representation of the video that uses a sub-block merge candidate list, zero motion non-affine padding candidates into the sub-block merge candidate list in case that the sub-block merge candidate list is not fulfilled; and performing, subsequent to the inserting, the conversion.

[0017] In yet another representative aspect, the disclosed technology may be used to provide a method for video processing. This method includes determining, for a conversion between a current block of a video and a bitstream representation of the video, motion vectors using a rule that determines that the motion vectors are derived from one or more motion vectors of a block covering a corresponding position in a collocated picture; and performing, based on the motion vectors, the conversion.

[0018] In yet another representative aspect, the disclosed technology may be used to provide a method for video processing. This method includes determining, for a conversion between a current block of video and a bitstream representation of the video, to use an sub-block based temporal motion vector prediction (sbTMVP) tool with a default motion candidate for the conversion in case that a temporal block associated with the current block or a sub-block of the current block in a collocated picture is coded using a coding mode in which a video unit is reconstructed based on previously coded samples in a same picture; and performing, based on the default motion candidate, the conversion.

[0019] In yet another representative aspect, the disclosed technology may be used to provide a method for video processing. This method includes deriving, for an sub-block based temporal motion vector prediction (sbTMVP) process that is part of a conversion between a current block of video and a bitstream representation of the video, default motion information for the sbTMVP process based on a location of a position that is used in a sub-block motion information derivation process for a sub-block of the current block; and performing, based on the default motion information, the conversion.

[0020] In yet another representative aspect, the disclosed technology may be used to provide a method for video processing. This method includes determining, for a conversion between a current block of a video coded using an sub-block based temporal motion vector prediction (sbTMVP) tool and a bitstream representation of the video, a modified motion vector used to locate a corresponding block in a different picture from the current picture of the current block, wherein the modified motion vector is generated by right-shifting a motion vector, used for prediction in the sbTMVP tool, to an integer precision, and performing, based on the determining, the conversion.

[0021] In yet another example aspect, a video encoder apparatus is disclosed. The video encoder apparatus includes a processor that is configured to implement a method described herein.

[0022] In yet another example aspect, a video decoder apparatus is disclosed. The video decoder apparatus includes a processor that is configured to implement a method described herein.

[0023] In yet another aspect, a computer readable medium having code stored thereupon is disclosed. The code, when executed by a processor, causes the processor to implement a method described in the present document.

[0024] These, and other, aspects are described in the present document.

BRIEF DESCRIPTION OF THE DRAWINGS

[0025] FIG. 1 is an example of derivation process for merge candidates list construction.

[0026] FIG. 2 shows example positions of spatial merge candidates.

[0027] FIG. 3 shows an example of candidate pairs considered for redundancy check of spatial merge candidates.

[0028] FIGS. 4A and 4B show example positions for a second prediction unit (PU) of $N \times 2N$ and $2N \times N$ partitions.

[0029] FIG. 5 is an example illustration of motion vector scaling for temporal merge candidate.

[0030] FIG. 6 shows example candidate positions for temporal merge candidate, C0 and C1.

[0031] FIG. 7 shows an example of combined bi-predictive merge candidate.

[0032] FIG. 8 shows an example derivation process for motion vector prediction candidates.

[0033] FIG. 9 is an example illustration of motion vector scaling for spatial motion vector candidate.

[0034] FIG. 10 shows an example of alternative temporal motion vector prediction (ATMVP) motion prediction for a CU.

[0035] FIG. 11 shows an example of one CU with four sub-blocks (A-D) and its neighbouring blocks (a-d).

[0036] FIG. 12 is a flowchart of an example of encoding with different MV precision

[0037] FIGS. 13A and 13B show 135 degree partition type (splitting from top-left corner to bottom-right corner) and 45 degree splitting patterns, respectively.

[0038] FIG. 14 shows an example of position of the neighboring blocks.

[0039] FIG. 15 shows examples of above and left blocks.

[0040] FIGS. 16A and 16B show examples of 2 control point motion vectors (CPMVs) and 3 CPMVs, respectively.

[0041] FIG. 17 shows an example of affine motion vector field (MVF) per sub-block.

[0042] FIGS. 18A and 18B show examples of 4 and 6 parameter affine models, respectively.

[0043] FIG. 19 shows an example of MVP for AF_INTER for inherited affine candidates.

[0044] FIG. 20 shows an example of constructing affine motion predictors in AF_INTER.

[0045] FIGS. 21A and 21B show examples of control point motion vectors in affine coding in AF_MERGE.

[0046] FIG. 22 shows examples of candidate positions for affine merge mode.

[0047] FIG. 23 shows an example of intra-picture block copy operation.

[0048] FIG. 24 shows an example of a valid corresponding region in the collocated picture.

[0049] FIG. 25 shows an example flowchart for history based motion vector prediction.

- [0050] FIG. 26 shows modified merge list construction process.
- [0051] FIG. 27 shows an example embodiment of the proposed valid region when the current block is inside a basic region.
- [0052] FIG. 28 shows an example embodiment of a valid region when the current block is not inside a basic region.
- [0053] FIGS. 29A and 29B show existing and proposed examples of locations for identification of default motion information, respectively.
- [0054] FIGS. 30–45 are flowcharts for examples of method of video processing.
- [0055] FIG. 46 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.
- [0056] FIG. 47 is a block diagram of an example video processing system in which disclosed techniques may be implemented.

DETAILED DESCRIPTION

[0057] The present document provides various techniques that can be used by a decoder of video bitstreams to improve the quality of decompressed or decoded digital video or images. Furthermore, a video encoder may also implement these techniques during the process of encoding in order to reconstruct decoded frames used for further encoding.

[0058] Section headings are used in the present document for ease of understanding and do not limit the embodiments and techniques to the corresponding sections. As such, embodiments from one section can be combined with embodiments from other sections.

1. Summary

[0059] This patent document is related to video coding technologies. Specifically, it is related to motion vector coding in video coding. It may be applied to the existing video coding standard like HEVC, or the standard (Versatile Video Coding) to be finalized. It may be also applicable to future video coding standards or video codec.

2. Introductory remarks

[0060] Video coding standards have evolved primarily through the development of the well-known ITU-T and ISO/IEC standards. The ITU-T produced H.261 and H.263, ISO/IEC

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

[0061] The latest version of VVC draft, i.e., Versatile Video Coding (Draft 3) is found at:

[0062] http://phenix.it-sudparis.eu/jvet/doc_end_user/documents/12_Macao/wg11/JVET-L1001-v2.zip The latest reference software of VVC, named VTM, could be found at:

[0063] https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM/tags/VTM-3.0rc1

2.1 Inter prediction in HEVC/H.265

[0064] Each inter-predicted PU has motion parameters for one or two reference picture lists. Motion parameters include a motion vector and a reference picture index. Usage of one of the two reference picture lists may also be signalled using *inter_pred_idc*. Motion vectors may be explicitly coded as deltas relative to predictors.

[0065] When a CU is coded with skip mode, one PU is associated with the CU, and there are no significant residual coefficients, no coded motion vector delta or reference picture index. A merge mode is specified whereby the motion parameters for the current PU are obtained from neighbouring PUs, including spatial and temporal candidates. The merge mode can be applied to any inter-predicted PU, not only for skip mode. The alternative to merge mode is the explicit transmission of motion parameters, where motion vector (to be more precise, motion vector differences (MVD) compared to a motion vector predictor), corresponding reference picture index for each reference picture list and reference picture list usage are signalled explicitly per each PU. Such a mode is named advanced motion vector prediction (AMVP) in this disclosure.

[0066] When signalling indicates that one of the two reference picture lists is to be used, the PU is produced from one block of samples. This is referred to as ‘uni-prediction’. Uni-prediction is available both for P-slices and B-slices.

[0067] When signalling indicates that both of the reference picture lists are to be used, the PU is produced from two blocks of samples. This is referred to as ‘bi-prediction’. Bi-prediction is available for B-slices only.

[0068] The following text provides the details on the inter prediction modes specified in HEVC. The description will start with the merge mode.

2.1.1 Reference picture list

[0069] In HEVC, the term inter prediction is used to denote prediction derived from data elements (e.g., sample values or motion vectors) of reference pictures other than the current decoded picture. Like in H.264/AVC, a picture can be predicted from multiple reference pictures. The reference pictures that are used for inter prediction are organized in one or more reference picture lists. The reference index identifies which of the reference pictures in the list should be used for creating the prediction signal.

[0070] A single reference picture list, List 0, is used for a P slice and two reference picture lists, List 0 and List 1 are used for B slices. It should be noted reference pictures included in List 0/1 could be from past and future pictures in terms of capturing/display order.

2.1.2 Merge Mode

2.1.2.1 Derivation of candidates for merge mode

[0071] When a PU is predicted using merge mode, an index pointing to an entry in the *merge candidates list* is parsed from the bitstream and used to retrieve the motion information. The construction of this list is specified in the HEVC standard and can be summarized according to the following sequence of steps:

- Step 1: Initial candidates derivation
 - Step 1.1: Spatial candidates derivation
 - Step 1.2: Redundancy check for spatial candidates
 - Step 1.3: Temporal candidates derivation
- Step 2: Additional candidates insertion

- Step 2.1: Creation of bi-predictive candidates
- Step 2.2: Insertion of zero motion candidates

[0072] These steps are also schematically depicted in FIG. 1. For spatial merge candidate derivation, a maximum of four merge candidates are selected among candidates that are located in five different positions. For temporal merge candidate derivation, a maximum of one merge candidate is selected among two candidates. Since constant number of candidates for each PU is assumed at decoder, additional candidates are generated when the number of candidates obtained from step 1 does not reach the maximum number of merge candidate (MaxNumMergeCand) which is signalled in slice header. Since the number of candidates is constant, index of best merge candidate is encoded using truncated unary binarization (TU). If the size of CU is equal to 8, all the PUs of the current CU share a single merge candidate list, which is identical to the merge candidate list of the $2N \times 2N$ prediction unit.

[0073] In the following, the operations associated with the aforementioned steps are detailed.

2.1.2.2 Spatial candidates derivation

[0074] In the derivation of spatial merge candidates, a maximum of four merge candidates are selected among candidates located in the positions depicted in FIG. 2. The order of derivation is A_1, B_1, B_0, A_0 and B_2 . Position B_2 is considered only when any PU of position A_1, B_1, B_0, A_0 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. 3 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. Another source of duplicate motion information is the “*second PU*” associated with partitions different from $2N \times 2N$. As an example, FIGS. 4A-4B depicts the second PU for the case of $N \times 2N$ and $2N \times N$, respectively. When the current PU is partitioned as $N \times 2N$, candidate at position A_1 is not considered for list construction. In fact, by adding this candidate will lead to two prediction units having the same motion information, which is redundant to just have one PU in a coding unit. Similarly, position B_1 is not considered when the current PU is partitioned as $2N \times N$.

2.1.2.3 Temporal candidates derivation

[0075] 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 PU belonging to the picture which has the smallest POC difference with current picture within the given reference picture list. The reference picture list to be used for derivation of the co-located PU 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. 5, which is scaled from the motion vector of the co-located PU using the POC distances, t_b and t_d , where t_b is defined to be the POC difference between the reference picture of the current picture and the current picture and t_d is defined to be the POC difference between the reference picture of the co-located picture and the co-located picture. The reference picture index of temporal merge candidate is set equal to zero. A practical realization of the scaling process is described in the HEVC specification. For a B-slice, two motion vectors, one is for reference picture list 0 and the other is for reference picture list 1, are obtained and combined to make the bi-predictive merge candidate.

[0076] FIG. 5 is an illustration of motion vector scaling for temporal merge candidate.

[0077] In the co-located PU (Y) belonging to the reference frame, the position for the temporal candidate is selected between candidates C_0 and C_1 , as depicted in FIG. 6. If PU at position C_0 is not available, is intra coded, or is outside of the current coding tree unit (CTU aka. LCU, largest coding unit) row, position C_1 is used. Otherwise, position C_0 is used in the derivation of the temporal merge candidate.

[0078] FIG. 6 shows examples of candidate positions for temporal merge candidate, C_0 and C_1 .

2.1.2.4 Additional candidates insertion

[0079] Besides spatial and temporal merge candidates, there are two additional types of merge candidates: combined bi-predictive merge candidate and zero merge candidate. Combined bi-predictive merge candidates are generated by utilizing spatial and temporal merge candidates. Combined bi-predictive merge candidate is used for B-Slice only. The combined bi-predictive candidates are generated by combining the first reference picture list motion parameters of an initial candidate with the second reference picture list motion parameters of another. If these two tuples provide different motion hypotheses, they will form a new bi-predictive candidate. As an example, FIG. 7 depicts the case when two candidates in the original list (on the left), which

have mvL0 and refIdxL0 or mvL1 and refIdxL1, are used to create a combined bi-predictive merge candidate added to the final list (on the right). There are numerous rules regarding the combinations which are considered to generate these additional merge candidates.

[0080] Zero motion candidates are inserted to fill the remaining entries in the merge candidates list and therefore hit the MaxNumMergeCand capacity. These candidates have zero spatial displacement and a reference picture index which starts from zero and increases every time a new zero motion candidate is added to the list.

[0081] More specifically, the following steps are performed in order till the merge list is full:

1. Set variable numRef to either number of reference picture associated with list 0 for P slice, or the minimum number of reference pictures in two lists for B slice;
2. Add non-repeated zero motion candidates:
For variable i being 0...numRef-1, add a default motion candidate with MV set to (0, 0) and reference picture index set to i for list 0 (if P slice), or for both lists (if B slice).
3. Add repeated zero motion candidates with MV set to (0, 0), reference picture index of list 0 set to 0 (if P slice) and reference picture indices of both lists set to 0 (if B slice).

[0082] Finally, no redundancy check is performed on these candidates.

2.1.3 Advanced motion vector prediction (AMVP)

[0083] AMVP exploits spatio-temporal correlation of motion vector with neighbouring PUs, which is used for explicit transmission of motion parameters. For each reference picture list, a motion vector candidate list is constructed by firstly checking availability of left, above temporally neighbouring PU positions, removing redundant candidates and adding zero vector to make the candidate list to be constant length. Then, the encoder can select the best predictor from the candidate list and transmit the corresponding index indicating the chosen candidate. Similarly with merge index signalling, the index of the best motion vector candidate is encoded using truncated unary. The maximum value to be encoded in this case is 2 (see FIG. 8). In the following sections, details about derivation process of motion vector prediction candidate are provided.

2.1.3.1 Derivation of AMVP candidates

[0084] FIG. 8 summarizes derivation process for motion vector prediction candidate.

[0085] In motion vector prediction, two types of motion vector candidates are considered: spatial motion vector candidate and temporal motion vector candidate. For spatial motion vector candidate derivation, two motion vector candidates are eventually derived based on motion vectors of each PU located in five different positions as depicted in FIG. 2.

[0086] For temporal motion vector candidate derivation, one motion vector candidate is selected from two candidates, which are derived based on two different co-located positions. After the first list of spatio-temporal candidates is made, duplicated motion vector candidates in the list are removed. If the number of potential candidates is larger than two, motion vector candidates whose reference picture index within the associated reference picture list is larger than 1 are removed from the list. If the number of spatio-temporal motion vector candidates is smaller than two, additional zero motion vector candidates is added to the list.

2.1.3.2 Spatial motion vector candidates

[0087] In the derivation of spatial motion vector candidates, a maximum of two candidates are considered among five potential candidates, which are derived from PUs located in positions as depicted in FIG. 2, those positions being the same as those of motion merge. The order of derivation for the left side of the current PU is defined as A_0 , A_1 , and scaled A_0 , scaled A_1 . The order of derivation for the above side of the current PU is defined as B_0 , B_1 , B_2 , scaled B_0 , scaled B_1 , scaled B_2 . For each side there are therefore four cases that can be used as motion vector candidate, with two cases not required to use spatial scaling, and two cases where spatial scaling is used. The four different cases are summarized as follows.

- No spatial scaling
 - (1) Same reference picture list, and same reference picture index (same POC)
 - (2) Different reference picture list, but same reference picture (same POC)
- Spatial scaling
 - (3) Same reference picture list, but different reference picture (different POC)
 - (4) Different reference picture list, and different reference picture (different POC)

[0088] The no-spatial-scaling cases are checked first followed by the spatial scaling. Spatial scaling is considered when the POC is different between the reference picture of the neighbouring PU and that of the current PU regardless of reference picture list. If all PUs of left candidates are not available or are intra coded, scaling for the above motion vector is allowed to

help parallel derivation of left and above MV candidates. Otherwise, spatial scaling is not allowed for the above motion vector.

[0089] In a spatial scaling process, the motion vector of the neighbouring PU is scaled in a similar manner as for temporal scaling, as depicted as FIG. 9. The main difference is that the reference picture list and index of current PU is given as input; the actual scaling process is the same as that of temporal scaling.

2.1.3.3 Temporal motion vector candidates

[0090] Apart for the reference picture index derivation, all processes for the derivation of temporal merge candidates are the same as for the derivation of spatial motion vector candidates (see FIG. 6). The reference picture index is signalled to the decoder.

2.2 Sub-CU based motion vector prediction methods in JEM

[0091] In the JEM with QTBT, each CU can have at most one set of motion parameters for each prediction direction. Two sub-CU level motion vector prediction methods are considered in the encoder by splitting a large CU into sub-CUs and deriving motion information for all the sub-CUs of the large CU. Alternative temporal motion vector prediction (ATMVP) method allows each CU to fetch multiple sets of motion information from multiple blocks smaller than the current CU in the collocated reference picture. In spatial-temporal motion vector prediction (STMVP) method motion vectors of the sub-CUs are derived recursively by using the temporal motion vector predictor and spatial neighbouring motion vector.

[0092] To preserve more accurate motion field for sub-CU motion prediction, the motion compression for the reference frames is currently disabled.

[0093] FIG. 10 shows an example of ATMVP motion prediction for a CU.

2.2.1 Alternative temporal motion vector prediction

[0094] In the alternative temporal motion vector prediction (ATMVP) method, the motion vectors temporal motion vector prediction (TMVP) is modified by fetching multiple sets of motion information (including motion vectors and reference indices) from blocks smaller than the current CU. In some implementations, the sub-CUs are square $N \times N$ blocks (N is set to 4 by default).

[0095] ATMVP predicts the motion vectors of the sub-CUs within a CU in two steps. The first step is to identify the corresponding block in a reference picture with a so-called temporal vector. The reference picture is called the motion source picture. The second step is to split the current CU into sub-CUs and obtain the motion vectors as well as the reference indices of each sub-CU from the block corresponding to each sub-CU.

[0096] In the first step, a reference picture and the corresponding block is determined by the motion information of the spatial neighbouring blocks of the current CU. To avoid the repetitive scanning process of neighbouring blocks, the first merge candidate in the merge candidate list of the current CU is used. The first available motion vector as well as its associated reference index are set to be the **temporal vector** and the index to the motion source picture. This way, in ATMVP, the corresponding block may be more accurately identified, compared with TMVP, wherein the corresponding block (sometimes called collocated block) is always in a bottom-right or center position relative to the current CU.

[0097] In the second step, a corresponding block of the sub-CU is identified by the temporal vector in the motion source picture, by adding to the coordinate of the current CU the temporal vector. For each sub-CU, the motion information of its corresponding block (the smallest motion grid that covers the center sample) is used to derive the motion information for the sub-CU. After the motion information of a corresponding $N \times N$ block is identified, it is converted to the motion vectors and reference indices of the current sub-CU, in the same way as TMVP of HEVC, wherein motion scaling and other procedures apply. For example, the decoder checks whether the low-delay condition (i.e. the POCs of all reference pictures of the current picture are smaller than the POC of the current picture) is fulfilled and possibly uses motion vector MV_x (the motion vector corresponding to reference picture list X) to predict motion vector MV_y (with X being equal to 0 or 1 and Y being equal to 1-X) for each sub-CU.

2.2.2 Spatial-temporal motion vector prediction (STMVP)

[0098] In this method, the motion vectors of the sub-CUs are derived recursively, following raster scan order. FIG. 11 illustrates this concept. Let us consider an 8×8 CU which contains four 4×4 sub-CUs A, B, C, and D. The neighbouring 4×4 blocks in the current frame are labelled as a, b, c, and d.

[0099] The motion derivation for sub-CU A starts by identifying its two spatial neighbours. The first neighbour is the $N \times N$ block above sub-CU A (block c). If this block c is not available or is intra coded the other $N \times N$ blocks above sub-CU A are checked (from left to right, starting at block c). The second neighbour is a block to the left of the sub-CU A (block b). If block b is not available or is intra coded other blocks to the left of sub-CU A are checked (from top to bottom, starting at block b). The motion information obtained from the neighbouring blocks for each list is scaled to the first reference frame for a given list. Next, temporal motion vector predictor (TMVP) of sub-block A is derived by following the same procedure of TMVP derivation as specified in HEVC. The motion information of the collocated block at location D is fetched and scaled accordingly. Finally, after retrieving and scaling the motion information, all available motion vectors (up to 3) are averaged separately for each reference list. The averaged motion vector is assigned as the motion vector of the current sub-CU.

2.2.3 Sub-CU motion prediction mode signalling

[00100] The sub-CU modes are enabled as additional merge candidates and there is no additional syntax element required to signal the modes. Two additional merge candidates are added to merge candidates list of each CU to represent the ATMVP mode and STMVP mode. Up to seven merge candidates are used, if the sequence parameter set indicates that ATMVP and STMVP are enabled. The encoding logic of the additional merge candidates is the same as for the merge candidates in the HM, which means, for each CU in P or B slice, two more RD checks is needed for the two additional merge candidates.

[00101] In the JEM, all bins of merge index is context coded by CABAC. While in HEVC, only the first bin is context coded and the remaining bins are context by-pass coded.

2.3 Inter prediction methods in VVC

[00102] There are several new coding tools for inter prediction improvement, such as Adaptive motion vector difference resolution (AMVR) for signaling MVD, affine prediction mode, Triangular prediction mode (TPM), ATMVP, Generalized Bi-Prediction (GBI), Bi-directional Optical flow (BIO).

2.3.1 Adaptive motion vector difference resolution

[00103] In HEVC, motion vector differences (MVDs) (between the motion vector and predicted motion vector of a PU) are signalled in units of quarter luma samples when `use_integer_mv_flag` is equal to 0 in the slice header. In the VVC, a locally adaptive motion vector resolution (LAMVR) is introduced. In the VVC, MVD can be coded in units of quarter luma samples, integer luma samples or four luma samples (i.e., $\frac{1}{4}$ -pel, 1-pel, 4-pel). The MVD resolution is controlled at the coding unit (CU) level, and MVD resolution flags are conditionally signalled for each CU that has at least one non-zero MVD components.

[00104] For a CU that has at least one non-zero MVD components, a first flag is signalled to indicate whether quarter luma sample MV precision is used in the CU. When the first flag (equal to 1) indicates that quarter luma sample MV precision is not used, another flag is signalled to indicate whether integer luma sample MV precision or four luma sample MV precision is used.

[00105] When the first MVD resolution flag of a CU is zero, or not coded for a CU (meaning all MVDs in the CU are zero), the quarter luma sample MV resolution is used for the CU. When a CU uses integer-luma sample MV precision or four-luma-sample MV precision, the MVPs in the AMVP candidate list for the CU are rounded to the corresponding precision.

[00106] In the encoder, CU-level RD checks are used to determine which MVD resolution is to be used for a CU. That is, the CU-level RD check is performed three times for each MVD resolution. To accelerate encoder speed, the following encoding schemes are applied in the JEM.

- During RD check of a CU with normal quarter luma sample MVD resolution, the motion information of the current CU (integer luma sample accuracy) is stored. The stored motion information (after rounding) is used as the starting point for further small range motion vector refinement during the RD check for the same CU with integer luma sample and 4 luma sample MVD resolution so that the time-consuming motion estimation process is not duplicated three times.
- RD check of a CU with 4 luma sample MVD resolution is conditionally invoked. For a CU, when RD cost integer luma sample MVD resolution is much larger than that of quarter luma sample MVD resolution, the RD check of 4 luma sample MVD resolution for the CU is skipped.

[00107] The encoding process is shown in FIG. 12. First, 1/4 pel MV is tested and the RD cost is calculated and denoted as $RDCost0$, then integer MV is tested and the RD cost is denoted as $RDCost1$. If $RDCost1 < th * RDCost0$ (wherein th is a positive value), then 4-pel MV is tested; otherwise, 4-pel MV is skipped. Basically, motion information and RD cost etc. are already known for 1/4 pel MV when checking integer or 4-pel MV, which can be reused to speed up the encoding process of integer or 4-pel MV.

2.3.2 Triangular prediction mode

[00108] The concept of the triangular prediction mode (TPM) is to introduce a new triangular partition for motion compensated prediction. As shown in FIGS. 13A-13B, it splits a CU into two triangular prediction units, in either diagonal or inverse diagonal direction. Each triangular prediction unit in the CU is inter-predicted using its own uni-prediction motion vector and reference frame index which are derived from a single uni-prediction candidate list. An adaptive weighting process is performed to the diagonal edge after predicting the triangular prediction units. Then, the transform and quantization process are applied to the whole CU. It is noted that this mode is only applied to merge mode (note: skip mode is treated as a special merge mode).

[00109] FIG. 13A-13B are an illustration of splitting a CU into two triangular prediction units (two splitting patterns): FIG. 13A: 135 degree partition type (splitting from top-left corner to bottom-right corner); FIG. 13B: 45 degree splitting patterns.

2.3.2.1 Uni-prediction candidate list for TPM

[00110] The uni-prediction candidate list, named **TPM motion candidate list**, consists of five uni-prediction motion vector candidates. It is derived from seven neighboring blocks including five spatial neighboring blocks (1 to 5) and two temporal co-located blocks (6 to 7), as shown in FIG. 14. The motion vectors of the seven neighboring blocks are collected and put into the uni-prediction candidate list according in the order of uni-prediction motion vectors, L0 motion vector of bi-prediction motion vectors, L1 motion vector of bi-prediction motion vectors, and averaged motion vector of the L0 and L1 motion vectors of bi-prediction motion vectors. If the number of candidates is less than five, zero motion vector is added to the list. Motion candidates added in this list for TPM are called TPM candidates, motion information derived from spatial/temporal blocks are called regular motion candidates.

[00111] More specifically, the following steps are involved:

- 1) Obtain regular motion candidates from $A_1, B_1, B_0, A_0, B_2, Col$ and $Col2$ (corresponding to block 1-7 in FIG. 14) with full pruning operations when adding a regular motion candidate from spatial neighboring blocks.
- 2) Set variable $numCurrMergeCand = 0$
- 3) For each regular motion candidates derived from $A_1, B_1, B_0, A_0, B_2, Col$ and $Col2$, if not pruned and $numCurrMergeCand$ is less than 5, if the regular motion candidate is **uni-prediction** (either from List 0 or List 1), it is directly added to the merge list as an TPM candidate with $numCurrMergeCand$ increased by 1. Such a TPM candidate is named '**originally uni-predicted candidate**'.
Full pruning is applied.
- 4) For each motion candidates derived from $A_1, B_1, B_0, A_0, B_2, Col$ and $Col2$, if not pruned, and $numCurrMergeCand$ is less than 5, if the regular motion candidate is **bi-prediction**, the motion information from **List 0** is added to the TPM merge list (that is, modified to be uni-prediction from List 0) as a new TPM candidate and $numCurrMergeCand$ increased by 1. Such a TPM candidate is named '**Truncated List0-predicted candidate**'.
Full pruning is applied.
- 5) For each motion candidates derived from $A_1, B_1, B_0, A_0, B_2, Col$ and $Col2$, if not pruned, and $numCurrMergeCand$ is less than 5, if the regular motion candidate is **bi-prediction**, the motion information from **List 1** is added to the TPM merge list (that is, modified to be uni-prediction from List 1) and $numCurrMergeCand$ increased by 1. Such a TPM candidate is named '**Truncated List1-predicted candidate**'.
Full pruning is applied.
- 6) For each motion candidates derived from $A_1, B_1, B_0, A_0, B_2, Col$ and $Col2$, if not pruned, and $numCurrMergeCand$ is less than 5, if the regular motion candidate is bi-prediction,
 - **If List 0 reference picture's slice QP is smaller than List 1 reference picture's slice QP**, the motion information of List 1 is firstly scaled to List 0 reference picture, and the average of the two MVs (one is from original List 0, and the other is the scaled MV from List 1) is added to the TPM merge list, such a candidate is called **averaged uni-prediction from List 0 motion candidate** and $numCurrMergeCand$ increased by 1.

- Otherwise, the motion information of List 0 is firstly scaled to List 1 reference picture, and the average of the two MVs (one is from original List 1, and the other is the scaled MV from List 0) is added to the TPM merge list, such a TPM candidate is called **averaged uni-prediction from List 1 motion candidate** and numCurrMergeCand increased by 1.

Full pruning is applied.

- 7) If numCurrMergeCand is less than 5, zero motion vector candidates are added.

[00112] When inserting a candidate to the list, if it has to be compared to all previously added candidates to see whether it is identical to one of them, such a process is called full pruning.

2.3.2.2 Adaptive weighting process

[00113] After predicting each triangular prediction unit, an adaptive weighting process is applied to the diagonal edge between the two triangular prediction units to derive the final prediction for the whole CU. Two weighting factor groups are defined as follows:

- 1st weighting factor group: {7/8, 6/8, 4/8, 2/8, 1/8} and {7/8, 4/8, 1/8} are used for the luminance and the chrominance samples, respectively;
- 2nd weighting factor group: {7/8, 6/8, 5/8, 4/8, 3/8, 2/8, 1/8} and {6/8, 4/8, 2/8} are used for the luminance and the chrominance samples, respectively.

[00114] Weighting factor group is selected based on the comparison of the motion vectors of two triangular prediction units. The 2nd weighting factor group is used when the reference pictures of the two triangular prediction units are different from each other or their motion vector difference is larger than 16 pixels. Otherwise, the 1st weighting factor group is used.

2.3.2.3 Signaling of triangular prediction mode (TPM)

[00115] One bit flag to indicate whether TPM is used may be firstly signaled. Afterwards, the indications of two splitting patterns (as depicted in FIGS. 13A-13B), and selected merge indices for each of the two partitions are further signaled.

2.3.2.3.1 Signaling of TPM flag

[00116] Let's denote one luma block's width and height by W and H, respectively. If $W \cdot H < 64$, triangular prediction mode is disabled.

[00117] When one block is coded with affine mode, triangular prediction mode is also disabled.

[00118] When one block is coded with merge mode, one bit flag may be signaled to indicate whether the triangular prediction mode is enabled or disabled for the block.

[00119] The flag is coded with 3 contexts, based on the following equation:

$$\begin{aligned} \text{Ctx index} = & ((\text{left block L available \&\& L is coded with TPM?}) 1: 0) \\ & + ((\text{Above block A available \&\& A is coded with TPM?}) 1: 0); \end{aligned}$$

[00120] FIG. 15 shows examples of neighboring blocks (A and L) used for context selection in TPM flag coding.

2.3.2.3.2 Signaling of an indication of two splitting patterns (as depicted in FIG. 13), and selected merge indices for each of the two partitions

[00121] It is noted that splitting patterns, merge indices of two partitions are jointly coded. In an existing implementation, it is restricted that the two partitions couldn't use the same reference index. Therefore, there are 2 (splitting patterns) * N (maximum number of merge candidates) *(N-1) possibilities wherein N is set to 5. One indication is coded and the mapping between the splitting patterns, two merge indices and coded indication are derived from the array defined below:

```
const uint8_t g_TriangleCombination[TRIANGLE_MAX_NUM_CANDS][3] = {
    { 0, 1, 0 }, { 1, 0, 1 }, { 1, 0, 2 }, { 0, 0, 1 }, { 0, 2, 0 },
    { 1, 0, 3 }, { 1, 0, 4 }, { 1, 1, 0 }, { 0, 3, 0 }, { 0, 4, 0 },
    { 0, 0, 2 }, { 0, 1, 2 }, { 1, 1, 2 }, { 0, 0, 4 }, { 0, 0, 3 },
    { 0, 1, 3 }, { 0, 1, 4 }, { 1, 1, 4 }, { 1, 1, 3 }, { 1, 2, 1 },
    { 1, 2, 0 }, { 0, 2, 1 }, { 0, 4, 3 }, { 1, 3, 0 }, { 1, 3, 2 },
    { 1, 3, 4 }, { 1, 4, 0 }, { 1, 3, 1 }, { 1, 2, 3 }, { 1, 4, 1 },
    { 0, 4, 1 }, { 0, 2, 3 }, { 1, 4, 2 }, { 0, 3, 2 }, { 1, 4, 3 },
    { 0, 3, 1 }, { 0, 2, 4 }, { 1, 2, 4 }, { 0, 4, 2 }, { 0, 3, 4 };
```

[00122] splitting patterns (45 degree or 135 degree) = g_TriangleCombination[signaled indication][0];

Merge index of candidate A = g_TriangleCombination[signaled indication][1];

Merge index of candidate B = g_TriangleCombination[signaled indication][2];

[00123] Once the two motion candidates A and B are derived, the two partitions' (PU1 and PU2) motion information could be set either from A or B. Whether PU1 uses the motion information of merge candidate A or B is dependent on the prediction directions of the two motion candidates. Table 1 shows the relationship between two derived motion candidates A and B, with the two partitions.

Table 1: Derivation of partitions' motion information from derived two merge candidates (A, B)

Prediction direction of A	Prediction direction of B	PU1's motion information	PU2's motion information
L0	L0	A (L0)	B (L0)
L1	L1	B (L1)	A (L1)
L0	L1	A (L0)	B (L1)
L1	L0	B (L0)	A (L1)

2.3.2.3.3 Entropy coding of the indication (denoted by merge_triangle_idx)

[00124] merge_triangle_idx is within the range [0, 39], inclusively. K-th order Exponential Golomb (EG) code is used for binarization of merge_triangle_idx wherein K is set to 1.

K-th order EG

[00125] To encode larger numbers in fewer bits (at the expense of using more bits to encode smaller numbers), this can be generalized using a nonnegative integer parameter k. To encode a nonnegative integer x in an order-k exp-Golomb code:

1. Encode $\lfloor x/2^k \rfloor$ using order-0 exp-Golomb code described above, then
2. Encode $x \bmod 2^k$ in binary

Table 1: Exp-Golomb-k coding examples

x	k=0	k=1	k=2	x	k=0	k=1	k=2
0	1	10	100	10	0001011	001100	01110
1	010	11	101	11	0001100	001101	01111

2	011	0100	110	12	0001101	001110	0010000
3	00100	0101	111	13	0001110	001111	0010001
4	00101	0110	01000	14	0001111	00010000	0010010
5	00110	0111	01001	15	000010000	00010001	0010011
6	00111	001000	01010	16	000010001	00010010	0010100
7	0001000	001001	01011	17	000010010	00010011	0010101
8	0001001	001010	01100	18	000010011	00010100	0010110
9	0001010	001011	01101	19	000010100	00010101	0010111

2.3.3 Affine motion compensation prediction

[00126] In HEVC, only translation motion model is applied for motion compensation prediction (MCP). While in the real world, there are many kinds of motion, e.g. zoom in/out, rotation, perspective motions and the other irregular motions. In VVC, a simplified affine transform motion compensation prediction is applied with 4-parameter affine model and 6-parameter affine model. As shown FIGS. 16A-16B, the affine motion field of the block is described by two control point motion vectors (CPMVs) for the 4-parameter affine model (FIG. 16A) and 3 CPMVs for the 6-parameter affine model (FIG. 16B).

[00127] The motion vector field (MVF) of a block is described by the following equations with the 4-parameter affine model (wherein the 4-parameter are defined as the variables a , b , e and f) in equation (1) and 6-parameter affine model (wherein the 4-parameter are defined as the variables a , b , c , d , e and f) in equation (2) respectively:

$$\begin{cases} mv^h(x, y) = ax - by + e = \frac{(mv_1^h - mv_0^h)}{w}x - \frac{(mv_1^v - mv_0^v)}{w}y + mv_0^h \\ mv^v(x, y) = bx + ay + f = \frac{(mv_1^v - mv_0^v)}{w}x + \frac{(mv_1^h - mv_0^h)}{w}y + mv_0^v \end{cases} \quad (1)$$

$$\begin{cases} mv^h(x, y) = ax + cy + e = \frac{(mv_1^h - mv_0^h)}{w}x + \frac{(mv_2^h - mv_0^h)}{h}y + mv_0^h \\ mv^v(x, y) = bx + dy + f = \frac{(mv_1^v - mv_0^v)}{w}x + \frac{(mv_2^v - mv_0^v)}{h}y + mv_0^v \end{cases} \quad (2)$$

[00128] where (mv^h_0, mv^v_0) is motion vector of the top-left corner control point, and (mv^h_1, mv^v_1) is motion vector of the top-right corner control point and (mv^h_2, mv^v_2) is motion vector of the bottom-left corner control point, all of the three motion vectors are called **control point motion vectors (CPMV)**, (x, y) represents the coordinate of a **representative point** relative to the top-left sample within current block and $(mv^h(x,y), mv^v(x,y))$ is the motion vector derived for a sample located at (x, y) . The CP motion vectors may be signaled (like in the affine AMVP mode) or derived on-the-fly (like in the affine merge mode). w and h are the width and height of the current block. In practice, the division is implemented by right-shift with a rounding operation. In VTM, the **representative point** is defined to be the center position of a sub-block, e.g., when the coordinate of the left-top corner of a sub-block relative to the top-left sample within current block is (xs, ys) , the coordinate of the representative point is defined to be $(xs+2, ys+2)$. For each sub-block (i.e., 4×4 in VTM), the representative point is utilized to derive the motion vector for the whole sub-block.

[00129] In order to further simplify the motion compensation prediction, sub-block based affine transform prediction is applied. To derive motion vector of each $M \times N$ (both M and N are set to 4 in current VVC) sub-block, the motion vector of the center sample of each sub-block, as shown in FIG. 17, is calculated according to Equation (1) and (2), and rounded to $1/16$ fraction accuracy. Then the motion compensation interpolation filters for $1/16$ -pel are applied to generate the prediction of each sub-block with derived motion vector. The interpolation filters for $1/16$ -pel are introduced by the affine mode.

[00130] After MCP, the high accuracy motion vector of each sub-block is rounded and saved as the same accuracy as the normal motion vector.

2.3.3.1 Signaling of affine prediction

[00131] Similar to the translational motion model, there are also two modes for signaling the side information due affine prediction. They are AFFINE_INTER and AFFINE_MERGE modes.

2.3.3.2 AF_INTER mode

[00132] For CUs with both width and height larger than 8, AF_INTER mode can be applied. An affine flag in CU level is signalled in the bitstream to indicate whether AF_INTER mode is used.

[00133] In this mode, for each reference picture list (List 0 or List 1), an affine AMVP candidate list is constructed with three types of affine motion predictors in the following order, wherein each candidate includes the estimated CPMVs of the current block. The differences of the best CPMVs found at the encoder side (such as mv_0 mv_1 mv_2 in FIG. 20) and the estimated CPMVs are signalled. In addition, the index of affine AMVP candidate from which the estimated CPMVs are derived is further signalled.

1) Inherited affine motion predictors

[00134] The checking order is similar to that of spatial MVPs in HEVC AMVP list construction. First, a left inherited affine motion predictor is derived from the first block in $\{A1, A0\}$ that is affine coded and has the same reference picture as in current block. Second, an above inherited affine motion predictor is derived from the first block in $\{B1, B0, B2\}$ that is affine coded and has the same reference picture as in current block. The five blocks A1, A0, B1, B0, B2 are depicted in FIG. 19.

[00135] Once a neighboring block is found to be coded with affine mode, the CPMVs of the coding unit covering the neighboring block are used to derive predictors of CPMVs of current block. For example, if A1 is coded with non-affine mode and A0 is coded with 4-parameter affine mode, the left inherited affine MV predictor will be derived from A0. In this case, the CPMVs of a CU covering A0, as denoted by MV_0^N for the top-left CPMV and MV_1^N for the top-right CPMV in FIG. 21B are utilized to derive the estimated CPMVs of current block, denoted by MV_0^C , MV_1^C , MV_2^C for the top-left (with coordinate (x0, y0)), top-right (with coordinate (x1, y1)) and bottom-right positions (with coordinate (x2, y2)) of current block.

2) Constructed affine motion predictors

[00136] A constructed affine motion predictor consists of control-point motion vectors (CPMVs) that are derived from neighboring inter coded blocks, as shown in FIG. 20, that have the same reference picture. If the current affine motion model is 4-parameter affine, the number of CPMVs is 2, otherwise if the current affine motion model is 6-parameter affine, the number of CPMVs is 3. The top-left CPMV $\overline{mv}_0$ is derived by the MV at the first block in the group $\{A, B, C\}$ that is inter coded and has the same reference picture as in current block. The top-right CPMV $\overline{mv}_1$ is derived by the MV at the first block in the group $\{D, E\}$ that is inter coded and has the same reference picture as in current block. The bottom-left CPMV $\overline{mv}_2$ is derived by

the MV at the first block in the group {F, G} that is inter coded and has the same reference picture as in current block.

- If the current affine motion model is 4-parameter affine, then a constructed affine motion predictor is inserted into the candidate list only if both $\overline{mv}_0$ and $\overline{mv}_1$ are founded, that is, $\overline{mv}_0$ and $\overline{mv}_1$ are used as the estimated CPMVs for top-left (with coordinate (x0, y0)), top-right (with coordinate (x1, y1)) positions of current block.
- If the current affine motion model is 6-parameter affine, then a constructed affine motion predictor is inserted into the candidate list only if $\overline{mv}_0$, $\overline{mv}_1$ and $\overline{mv}_2$ are all founded, that is, $\overline{mv}_0$, $\overline{mv}_1$ and $\overline{mv}_2$ are used as the estimated CPMVs for top-left (with coordinate (x0, y0)), top-right (with coordinate (x1, y1)) and bottom-right (with coordinate (x2, y2)) positions of current block.

[00137] No pruning process is applied when inserting a constructed affine motion predictor into the candidate list.

3) Normal AMVP motion predictors

[00138] The following applies until the number of affine motion predictors reaches the maximum.

- 1) Derive an affine motion predictor by setting all CPMVs equal to $\overline{mv}_2$ if available.
- 2) Derive an affine motion predictor by setting all CPMVs equal to $\overline{mv}_1$ if available.
- 3) Derive an affine motion predictor by setting all CPMVs equal to $\overline{mv}_0$ if available.
- 4) Derive an affine motion predictor by setting all CPMVs equal to HEVC TMVP if available.
- 5) Derive an affine motion predictor by setting all CPMVs to zero MV.

[00139] Note that $\overline{mv}_i$ is already derived in constructed affine motion predictor.

[00140] FIG. 18A shows an example of a 4-parameter affine model. FIG. 18B shows an example of a 6-parameter affine model.

[00141] FIG. 19 shows an example of MVP for AF_INTER for inherited affine candidates.

[00142] FIG. 20 shows an example of MVP for AF_INTER for constructed affine candidates.

[00143] In AF_INTER mode, when 4/6-parameter affine mode is used, 2/3 control points are required, and therefore 2/3 MVD needs to be coded for these control points, as shown in FIG. 18.

In an existing implementation, it is proposed to derive the MV as follows, i.e., mvd_1 and mvd_2 are predicted from mvd_0 .

$$\begin{aligned} mv_0 &= \overline{mv}_0 + mvd_0 \\ mv_1 &= \overline{mv}_1 + mvd_1 + mvd_0 \\ mv_2 &= \overline{mv}_2 + mvd_2 + mvd_0 \end{aligned}$$

[00144] Wherein $\overline{mv}_i$, mvd_i and mv_i are the predicted motion vector, motion vector difference and motion vector of the top-left pixel ($i = 0$), top-right pixel ($i = 1$) or left-bottom pixel ($i = 2$) respectively, as shown in FIG. 18B. Please note that the addition of two motion vectors (e.g., $mvA(xA, yA)$ and $mvB(xB, yB)$) is equal to summation of two components separately, that is, $newMV = mvA + mvB$ and the two components of newMV is set to $(xA + xB)$ and $(yA + yB)$, respectively.

2.3.3.3 AF_MERGE mode

[00145] When a CU is applied in AF_MERGE mode, it gets the first block coded with affine mode from the valid neighbour reconstructed blocks. And the selection order for the candidate block is from left, above, above right, left bottom to above left as shown in FIG. 21A (denoted by A, B, C, D, E in order). For example, if the neighbour left bottom block is coded in affine mode as denoted by A0 in FIG. 21B, the Control Point (CP) motion vectors mv_0^N , mv_1^N and mv_2^N of the top left corner, above right corner and left bottom corner of the neighbouring CU/PU which contains the block A are fetched. And the motion vector mv_0^C , mv_1^C and mv_2^C (which is only used for the 6-parameter affine model) of the top left corner/top right/bottom left on the current CU/PU is calculated based on mv_0^N , mv_1^N and mv_2^N . It should be noted that in VTM-2.0, sub-block (e.g. 4×4 block in VTM) located at the top-left corner stores mv_0 , the sub-block located at the top-right corner stores mv_1 if the current block is affine coded. If the current block is coded with the 6-parameter affine model, the sub-block located at the bottom-left corner stores mv_2 ; otherwise (with the 4-parameter affine model), LB stores mv_2' . Other sub-blocks stores the MVs used for MC.

[00146] After the CPMV of the current CU mv_0^C , mv_1^C and mv_2^C are derived, according to the simplified affine motion model Equation (1) and (2), the MVF of the current CU is generated. In order to identify whether the current CU is coded with AF_MERGE mode, an affine flag is signalled in the bitstream when there is at least one neighbour block is coded in affine mode.

[00147] In some existing implementations, an affine merge candidate list is constructed with following steps:

1) Insert inherited affine candidates

[00148] Inherited affine candidate means that the candidate is derived from the affine motion model of its valid neighbor affine coded block. The maximum two inherited affine candidates are derived from affine motion model of the neighboring blocks and inserted into the candidate list. For the left predictor, the scan order is {A0, A1}; for the above predictor, the scan order is {B0, B1, B2}.

2) Insert constructed affine candidates

[00149] If the number of candidates in affine merge candidate list is less than MaxNumAffineCand (e.g., 5), constructed affine candidates are inserted into the candidate list. Constructed affine candidate means the candidate is constructed by combining the neighbor motion information of each control point.

- a) The motion information for the control points is derived firstly from the specified spatial neighbors and temporal neighbor shown in FIG. 22. CP_k (k=1, 2, 3, 4) represents the k-th control point. A0, A1, A2, B0, B1, B2 and B3 are spatial positions for predicting CP_k (k=1, 2, 3); T is temporal position for predicting CP4.

The coordinates of CP1, CP2, CP3 and CP4 is (0, 0), (W, 0), (H, 0) and (W, H), respectively, where W and H are the width and height of current block.

[00150] The motion information of each control point is obtained according to the following priority order:

- For CP1, the checking priority is B2->B3->A2. B2 is used if it is available. Otherwise, if B2 is available, B3 is used. If both B2 and B3 are unavailable, A2 is used. If all the three candidates are unavailable, the motion information of CP1 cannot be obtained.
 - For CP2, the checking priority is B1->B0.
 - For CP3, the checking priority is A1->A0.
 - For CP4, T is used.
- b) Secondly, the combinations of controls points are used to construct an affine merge candidate.

- I. Motion information of three control points are needed to construct a 6-parameter affine candidate. The three control points can be selected from one of the following four combinations ($\{CP1, CP2, CP4\}$, $\{CP1, CP2, CP3\}$, $\{CP2, CP3, CP4\}$, $\{CP1, CP3, CP4\}$). Combinations $\{CP1, CP2, CP3\}$, $\{CP2, CP3, CP4\}$, $\{CP1, CP3, CP4\}$ will be converted to a 6-parameter motion model represented by top-left, top-right and bottom-left control points.
- II. Motion information of two control points are needed to construct a 4-parameter affine candidate. The two control points can be selected from one of the two combinations ($\{CP1, CP2\}$, $\{CP1, CP3\}$). The two combinations will be converted to a 4-parameter motion model represented by top-left and top-right control points.
- III. The combinations of constructed affine candidates are inserted into to candidate list as following order:
 $\{CP1, CP2, CP3\}$, $\{CP1, CP2, CP4\}$, $\{CP1, CP3, CP4\}$, $\{CP2, CP3, CP4\}$, $\{CP1, CP2\}$, $\{CP1, CP3\}$
 - i. For each combination, the reference indices of list X for each CP are checked, if they are all the same, then this combination has valid CPMVs for list X. If the combination does not have valid CPMVs for both list 0 and list 1, then this combination is marked as invalid. Otherwise, it is valid, and the CPMVs are put into the sub-block merge list.

3) Padding with zero motion vectors

[00151] If the number of candidates in affine merge candidate list is less than 5, zero motion vectors with zero reference indices are insert into the candidate list, until the list is full.

[00152] More specifically, for the sub-block merge candidate list, a 4-parameter merge candidate with MVs set to (0, 0) and prediction direction set to uni-prediction from list 0 (for P slice) and bi-prediction (for B slice).

2.3.4 Current Picture Referencing

[00153] Intra block copy (a.k.a. IBC, or intra picture block compensation), also named current picture referencing (CPR) was adopted in HEVC screen content coding extensions (SCC). This tool is very efficient for coding of screen content video in that repeated patterns in text and graphics rich content occur frequently **within the same picture**. Having a previously

reconstructed block with equal or similar pattern as a predictor can effectively reduce the prediction error and therefore improve coding efficiency. An example of the intra block compensation is illustrated in FIG. 23.

[00154] Similar to the design of CRP in HEVC SCC, In VVC, The use of the IBC mode is signaled at both sequence and picture level. When the IBC mode is enabled at sequence parameter set (SPS), it can be enabled at picture level. When the IBC mode is enabled at picture level, the current reconstructed picture is treated as a reference picture. Therefore, no syntax change on block level is needed on top of the existing VVC inter mode to signal the use of the IBC mode.

[00155] Main features:

- It is treated as a normal inter mode. Therefore, merge and skip modes are also available for the IBC mode. The merge candidate list construction is unified, containing merge candidates from the neighboring positions that are either coded in the IBC mode or the HEVC inter mode. Depending on the selected merge index, the current block under merge or skip mode can merge into either an IBC mode coded neighbor or otherwise an normal inter mode coded one with different pictures as reference pictures.
- Block vector prediction and coding schemes for the IBC mode reuse the schemes used for motion vector prediction and coding in the HEVC inter mode (AMVP and MVD coding).
- The motion vector for the IBC mode, also referred as block vector, is coded with **integer-pel precision**, but stored in memory in 1/16-pel precision after decoding as quarter-pel precision is required in interpolation and deblocking stages. When used in motion vector prediction for the IBC mode, the stored vector predictor will be right shifted by 4.
- Search range: it is restricted to be within the current CTU.
- CPR is disallowed when affine mode/triangular mode/GBI/weighted prediction is enabled.

2.3.5 Merge list design in VVC

[00156] There are three different merge list construction processes supported in VVC:

- 1) **Sub-block merge candidate list:** it includes ATMVP and affine merge candidates. One merge list construction process is shared for both affine modes and ATMVP mode. Here, the ATMVP and affine merge candidates may be added in order. Sub-block merge list size is signaled in slice header, and maximum value is 5.

- 2) **Uni-Prediction TPM merge list:** For triangular prediction mode, one merge list construction process for the two partitions is shared even two partitions could select their own merge candidate index. When constructing this merge list, the spatial neighbouring blocks and two temporal blocks of the block are checked. The motion information derived from spatial neighbours and temporal blocks are called regular motion candidates in our IDF. These regular motion candidates are further utilized to derive multiple TPM candidates. Please note the transform is performed in the whole block level, even two partitions may use different motion vectors for generating their own prediction blocks. Uni-Prediction TPM merge list size is fixed to be 5.
- 3) **Regular merge list:** For remaining coding blocks, one merge list construction process is shared. Here, the spatial/temporal/HMVP, pairwise combined bi-prediction merge candidates and zero motion candidates may be inserted in order. Regular merge list size is signaled in slice header, and maximum value is 6.

2.3.5.1 Sub-block merge candidate list

[00157] It is suggested that all the sub-block related motion candidates are put in a separate merge list in addition to the regular merge list for non-sub block merge candidates.

[00158] The sub-block related motion candidates are put in a separate merge list is named as 'sub-block merge candidate list'.

[00159] In one example, the sub-block merge candidate list includes affine merge candidates, and ATMVP candidate, and/or sub-block based STMVP candidate.

2.3.5.1.1 Another ATMVP Embodiment

[00160] In this contribution, the ATMVP merge candidate in the normal merge list is moved to the first position of the affine merge list. Such that all the merge candidates in the new list (i.e., sub-block based merge candidate list) are based on sub-block coding tools.

2.3.5.1.2 ATMVP in VTM-3.0

[00161] In VTM-3.0, a special merge candidate list, known as sub-block merge candidate list (a.k.a affine merge candidate list) is added besides the regular merge candidate list. The sub-block merge candidate list is filled with candidates in the following order:

- b. ATMVP candidate (maybe available or unavailable);

- c. Inherited Affine candidates;
- d. Constructed Affine candidates;
- e. Padding as zero MV 4-parameter affine model

[00162] The maximum number of candidates (denoted as ML) in the sub-block merge candidate list derived as below:

- 1) If the ATMVP usage flag (e.g. the flag may be named as “**sps_sbtmvp_enabled_flag**”) is on (equal to 1), but the affine usage flag (e.g. the flag may be named as “**sps_affine_enabled_flag**”) is off (equal to 0), then ML is set equal to 1.
- 2) If the ATMVP usage flag is off (equal to 0), and the affine usage flag is off (equal to 0), then ML is set equal to 0. In this case, the sub-block merge candidate list is not used.
- 3) Otherwise (the affine usage flag is on (equal to 1), the ATMVP usage flag is on or off), ML is signaled from the encoder to the decoder. Valid ML is $0 \leq ML \leq 5$.

[00163] When construct the sub-block merge candidate list, ATMVP candidate is checked first. If any one of the following conditions is true, ATMVP candidate is skipped and not put into the sub-block merge candidate list.

- 1) The ATMVP usage flag is off;
- 2) Any TMVP usage flag (e.g. the flag may be named as “**slice_temporal_mvp_enabled_flag**” when signaled at slice level) is off;
- 3) The reference picture with reference index 0 in reference list 0 is identical to the current picture (It is a CPR)

[00164] ATMVP in VTM-3.0 is much simpler than in JEM. When an ATMVP merge candidate is generated, the following process is applied:

- a. Check neighbouring blocks A1, B1, B0, A0 as shown in FIG. 22 in order, to find the first inter-coded, but not CPR-coded block, denoted as block X;
- b. Initialize TMV=(0,0). If there is a MV (denoted as MV') of block X, referring to the collocated reference picture (as signaled in the slice header), TMV is set equal to MV'.
- c. Suppose the center point of the current block is (x0, y0), then locate a corresponding position of (x0,y0) as $M=(x0+MV'x, y0+MV'y)$ in the collocated picture. Find the block Z covering M.
 - i. If Z is intra-coded, then ATMVP is unavailable;
 - ii. If Z is inter-coded, MVZ_0 and MVZ_1 for the two lists of block Z are scaled to (Reflist 0 index 0) and (Reflist 1 index 0) as MVdefault0, MVdefault1, and stored.

d. For each 8x8 sub-block, suppose its center point is (x0S, y0S), then locate a corresponding position of (x0S, y0S) as $MS=(x0S+MV \cdot x, y0S+MV \cdot y)$ in the collocated picture. Find the block ZS covering MS.

- i. If ZS is intra-coded, MVdefault0, MVdefault1 are assigned to the sub-block;
- ii. If ZS is inter-coded, MVZS_0 and MVZS_1 for the two lists of block ZS are scaled to (Reflist 0 index 0) and (Reflist 1 index 0) and are assigned to the sub-block;

[00165] MV clipping and masking in ATMVP:

[00166] When locating a corresponding position such as M or MS in the collocated picture, it is clipped to be inside a predefined region. The CTU size is $S \times S$, $S=128$ in VTM-3.0. Suppose the top-left position of the collocated CTU is (xCTU, yCTU), then the corresponding position M or MS at (xN, yN) will be clipped into the valid region $xCTU \leq xN < xCTU+S+4$; $yCTU \leq yN < yCTU+S$.

[00167] Besides Clipping, (xN, yN) is also masked as $xN=xN \& MASK$, $yN=yN \& MASK$, where MASK is an integer equal to $\sim(2^N-1)$, and $N=3$, to set the lowest 3 bits to be 0. So xN and yN must be numbers which are times of 8. (“~” represents the bitwise complement operator).

[00168] FIG. 24 shows an example of a valid corresponding region in the collocated picture.

2.3.5.1.3 Syntax design in slice header

if(sps_temporal_mvp_enabled_flag)	
slice_temporal_mvp_enabled_flag	u(1)
if(slice_type == B)	
mvd_l1_zero_flag	u(1)
if(slice_temporal_mvp_enabled_flag) {	
if(slice_type == B)	
collocated_from_l0_flag	u(1)
}	
...	
if(!sps_affine_enabled_flag){	
if(!sps_sbtmvp_enabled_flag)	
MaxSubBlockMergeListSize = 0	

else	
MaxSubBlockMergeListSize = 1	
}	
else{	
five_minus_max_num_affine_merge_cand	ue(v)
MaxSubBlockMergeListSize = 5- five_minus_max_num_affine_merge_cand	
}	

2.3.5.2 Regular merge list

[00169] Different from the merge list design, in VVC, the history-based motion vector prediction (HMVP) method is employed.

[00170] In HMVP, the previously coded motion information is stored. The motion information of a previously coded block is defined as an HMVP candidate. Multiple HMVP candidates are stored in a table, named as the HMVP table, and this table is maintained during the encoding/decoding process on-the-fly. The HMVP table is emptied when starting coding/decoding a new slice. Whenever there is an inter-coded block, the associated motion information is added to the last entry of the table as a new HMVP candidate. The overall coding flow is depicted in FIG. 25.

[00171] HMVP candidates could be used in both AMVP and merge candidate list construction processes. FIG. 26 depicts the modified merge candidate list construction process (highlighted in blue). When the merge candidate list is not full after the TMVP candidate insertion, HMVP candidates stored in the HMVP table could be utilized to fill in the merge candidate list. Considering that one block usually has a higher correlation with the nearest neighbouring block in terms of motion information, the HMVP candidates in the table are inserted in a descending order of indices. The last entry in the table is firstly added to the list, while the first entry is added in the end. Similarly, redundancy removal is applied on the HMVP candidates. Once the total number of available merge candidates reaches the maximal number of merge candidates allowed to be signaled, the merge candidate list construction process is terminated.

[00172] 2.4 MV Rounding

[00173] In VVC, when MV is right shifted, it is asked to be rounded toward zero. In a formulated way, for a MV(MV_x, MV_y) to be right shifted with N bits, the result MV'(MV_x', MV_y') will be derived as:

[00174] $MV_x' = (MV_x + ((1 \ll N) >> 1) - (MV_x \geq 0 ? 1 : 0)) >> N;$

[00175] $MV_y' = (MV_y + ((1 \ll N) >> 1) - (MV_y \geq 0 ? 1 : 0)) >> N;$

2.5 Embodiments of reference picture resampling (RPR)

[00176] ARC, a.k.a. reference picture resampling (RPR) has been incorporated in some existing and upcoming video standards.

[00177] In some embodiments of RPR, TMVP is disabled if the collocated picture has a different resolution to the current picture. Besides, BDOF and DMVR are disabled when the reference picture has a different resolution to the current picture.

[00178] To handle the normal MC when the reference picture has a different resolution than the current picture, the interpolation section is defined as below:

8.5.6.3 Fractional sample interpolation process

8.5.6.3.1 General

Inputs to this process are:

- a luma location (xSb, ySb) specifying the top-left sample of the current coding subblock relative to the top-left luma sample of the current picture,
- a variable sbWidth specifying the width of the current coding subblock,
- a variable sbHeight specifying the height of the current coding subblock,
- a motion vector offset mvOffset,
- a refined motion vector refMvLX,
- the selected reference picture sample array refPicLX,
- the half sample interpolation filter index hpelIfIdx,
- the bi-directional optical flow flag bdofFlag,
- a variable cIdx specifying the colour component index of the current block.

Outputs of this process are:

- an $(sbWidth + brdExtSize) \times (sbHeight + brdExtSize)$ array `predSamplesLX` of prediction sample values.

The prediction block border extension size `brdExtSize` is derived as follows:

$$brdExtSize = (bdoFlag \quad || \quad (inter_affine_flag[xSb][ySb] \quad \&\& \quad sps_affine_prof_enabled_flag)) ? 2 : 0(8-752)$$

The variable `fRefWidth` is set equal to the `PicOutputWidthL` of the reference picture in luma samples.

The variable `fRefHeight` is set equal to `PicOutputHeightL` of the reference picture in luma samples.

The motion vector `mvLX` is set equal to $(refMvLX - mvOffset)$.

- If `cldx` is equal to 0, the following applies:
 - The scaling factors and their fixed-point representations are defined as

$$hori_scale_fp = ((fRefWidth \ll 14) + (PicOutputWidthL \gg 1)) / PicOutputWidthL(8-753)$$

$$vert_scale_fp = ((fRefHeight \ll 14) + (PicOutputHeightL \gg 1)) / PicOutputHeightL(8-754)$$

- Let $(xIntL, yIntL)$ be a luma location given in full-sample units and $(xFracL, yFracL)$ be an offset given in 1/16-sample units. These variables are used only in this clause for specifying fractional-sample locations inside the reference sample arrays `refPicLX`.
- The top-left coordinate of the bounding block for reference sample padding $(xSbIntL, ySbIntL)$ is set equal to $(xSb + (mvLX[0] \gg 4), ySb + (mvLX[1] \gg 4))$.
- For each luma sample location $(x_L = 0..sbWidth - 1 + brdExtSize, y_L = 0..sbHeight - 1 + brdExtSize)$ inside the

prediction luma sample array predSamplesLX, the corresponding prediction luma sample value predSamplesLX[x_L][y_L] is derived as follows:

- Let (refxSb_L, refySb_L) and (refx_L, refy_L) be luma locations pointed to by a motion vector (refMvLX[0], refMvLX[1]) given in 1/16-sample units. The variables refxSb_L, refx_L, refySb_L, and refy_L are derived as follows:

$$\text{refxSb}_L = ((xSb \ll 4) + \text{refMvLX}[0]) * \text{hori_scale_fp} \quad (8-755)$$

$$\begin{aligned} \text{refx}_L = & ((\text{Sign}(\text{refxSb}) * ((\text{Abs}(\text{refxSb}) + 128) \gg 8) \\ & + x_L * ((\text{hori_scale_fp} + 8) \gg 4)) + 32) \gg 6 \end{aligned} \quad (8-756)$$

$$\text{refySb}_L = ((ySb \ll 4) + \text{refMvLX}[1]) * \text{vert_scale_fp} \quad (8-757)$$

$$\begin{aligned} \text{refy}_L = & ((\text{Sign}(\text{refySb}) * ((\text{Abs}(\text{refySb}) + 128) \gg 8) + y_L * \\ & ((\text{vert_scale_fp} + 8) \gg 4)) + 32) \gg 6 \end{aligned} \quad (8-758)$$

- The variables xInt_L, yInt_L, xFrac_L and yFrac_L are derived as follows:

$$xInt_L = \text{refx}_L \gg 4 \quad (8-759)$$

$$yInt_L = \text{refy}_L \gg 4 \quad (8-760)$$

$$xFrac_L = \text{refx}_L \& 15 \quad (8-761)$$

$$yFrac_L = \text{refy}_L \& 15 \quad (8-762)$$

- If bdoFlag is equal to TRUE or (sps_affine_prof_enabled_flag is equal to TRUE and inter_affine_flag[xSb][ySb] is equal to TRUE), and one or more of the following conditions are true, the prediction luma sample value predSamplesLX[x_L][y_L] is derived by invoking the luma integer sample fetching process as specified in clause 8.5.6.3.3 with (xInt_L + (xFrac_L >> 3) - 1), yInt_L + (yFrac_L >> 3) - 1) and refPicLX as inputs.

1. x_L is equal to 0.
2. x_L is equal to sbWidth + 1.

3. y_L is equal to 0.
 4. y_L is equal to $sbHeight + 1$.
- Otherwise, the prediction luma sample value $predSamplesLX[xL][yL]$ is derived by invoking the luma sample 8-tap interpolation filtering process as specified in clause 8.5.6.3.2 with
 $(xIntL - (brdExtSize > 0 ? 1 : 0), yIntL - (brdExtSize > 0 ? 1 : 0)),$
 $(xFracL, yFracL), (xSbIntL, ySbIntL), refPicLX, hpelIdx, sbWidth, sbHeight$ and
 (xSb, ySb) as inputs.
 - Otherwise ($cIdx$ is not equal to 0), the following applies:
 - Let $(xIntC, yIntC)$ be a chroma location given in full-sample units and $(xFracC, yFracC)$ be an offset given in 1/32 sample units. These variables are used only in this clause for specifying general fractional-sample locations inside the reference sample arrays $refPicLX$.
 - The top-left coordinate of the bounding block for reference sample padding $(xSbIntC, ySbIntC)$ is set equal to $((xSb / SubWidthC) + (mvLX[0] >> 5), (ySb / SubHeightC) + (mvLX[1] >> 5))$.
 - For each chroma sample location $(xC = 0..sbWidth - 1, yC = 0..sbHeight - 1)$ inside the prediction chroma sample arrays $predSamplesLX$, the corresponding prediction chroma sample value $predSamplesLX[xC][yC]$ is derived as follows:
 - Let $(refxSbC, refySbC)$ and $(refxC, refyC)$ be chroma locations pointed to by a motion vector $(mvLX[0], mvLX[1])$ given in 1/32-sample units. The variables $refxSbC, refySbC, refxC$ and $refyC$ are derived as follows:

$$refxSbC = ((xSb / SubWidthC << 5) + mvLX[0]) * hori_scale_fp \quad (8-763)$$

$$refxC = ((Sign(refxSbC) * ((Abs(refxSbC) + 256) >> 9) + xC * ((hori_scale_fp + 8) >> 4) + 16) >> 5) \quad (8-764)$$

$$refySbC = ((ySb / SubHeightC << 5) + mvLX[1]) * vert_scale_fp \quad (8-765)$$

$$\begin{aligned} \text{refy}_C = & ((\text{Sign}(\text{refySb}_C) * (\text{Abs}(\text{refySb}_C) + 256)) \gg 9) \\ & + yC * ((\text{vert_scale_fp} + 8) \gg 4) + 16) \gg 5 \end{aligned} \quad (8-766)$$

- The variables xInt_C , yInt_C , xFrac_C and yFrac_C are derived as follows:

$$\text{xInt}_C = \text{refx}_C \gg 5 \quad (8-767)$$

$$\text{yInt}_C = \text{refy}_C \gg 5 \quad (8-768)$$

$$\text{xFrac}_C = \text{refy}_C \& 31 \quad (8-769)$$

$$\text{yFrac}_C = \text{refy}_C \& 31 \quad (8-770)$$

- The prediction sample value $\text{predSamplesLX}[\text{x}_C][\text{y}_C]$ is derived by invoking the process specified in clause 8.5.6.3.4 with $(\text{xInt}_C, \text{yInt}_C)$, $(\text{xFrac}_C, \text{yFrac}_C)$, $(\text{xSbInt}_C, \text{ySbInt}_C)$, sbWidth , sbHeight and refPicLX as inputs.

8.5.6.3.2 Luma sample interpolation filtering process

Inputs to this process are:

- a luma location in full-sample units $(\text{xInt}_L, \text{yInt}_L)$,
- a luma location in fractional-sample units $(\text{xFrac}_L, \text{yFrac}_L)$,
- a luma location in full-sample units $(\text{xSbInt}_L, \text{ySbInt}_L)$ specifying the top-left sample of the bounding block for reference sample padding relative to the top-left luma sample of the reference picture,
- the luma reference sample array refPicLXL ,
- the half sample interpolation filter index hpelIdx ,
- a variable sbWidth specifying the width of the current subblock,
- a variable sbHeight specifying the height of the current subblock,
- a luma location (xSb, ySb) specifying the top-left sample of the current subblock relative to the top-left luma sample of the current picture,

Output of this process is a predicted luma sample value predSampleLXL

The variables shift1, shift2 and shift3 are derived as follows:

- The variable shift1 is set equal to $\text{Min}(4, \text{BitDepth}_Y - 8)$, the variable shift2 is set equal to 6 and the variable shift3 is set equal to $\text{Max}(2, 14 - \text{BitDepth}_Y)$.
- The variable picW is set equal to pic_width_in_luma_samples and the variable picH is set equal to pic_height_in_luma_samples.

The luma interpolation filter coefficients $f_L[p]$ for each 1/16 fractional sample position p equal to x_{Frac_L} or y_{Frac_L} are derived as follows:

- If $\text{MotionModelIdc}[x_{\text{Sb}}][y_{\text{Sb}}]$ is greater than 0, and sbWidth and sbHeight are both equal to 4, the luma interpolation filter coefficients $f_L[p]$ are specified in Table 8-12.
- Otherwise, the luma interpolation filter coefficients $f_L[p]$ are specified in Table 8-11 depending on hpelIdx.

The luma locations in full-sample units (x_{Int_i} , y_{Int_i}) are derived as follows for $i = 0..7$:

- If subpic_treated_as_pic_flag[SubPicIdx] is equal to 1, the following applies:

$$x_{\text{Int}_i} = \text{Clip3}(\text{SubPicLeftBoundaryPos}, \text{SubPicRightBoundaryPos}, x_{\text{Int}_L} + i - 3) \quad (8-771)$$

$$y_{\text{Int}_i} = \text{Clip3}(\text{SubPicTopBoundaryPos}, \text{SubPicBotBoundaryPos}, y_{\text{Int}_L} + i - 3) \quad (8-772)$$

- Otherwise (subpic_treated_as_pic_flag[SubPicIdx] is equal to 0), the following applies:

$$x_{\text{Int}_i} = \text{Clip3}(0, \text{picW} - 1, \text{sps_ref_wraparound_enabled_flag} \quad ?$$

$$\begin{aligned} & \text{ClipH}((\text{sps_ref_wraparound_offset_minus1} + 1) * \text{MinCbSizeY}, \text{picW}, x_{\text{Int}_L} + i - 3) \\ & : \quad (8-773) \\ & x_{\text{Int}_L} + i - 3 \end{aligned}$$

$$\begin{aligned} y_{\text{Int}_i} &= \text{Clip3}(0, \text{picH} - 1, y_{\text{Int}_L} + i - 3) \\ & (8-774) \end{aligned}$$

The luma locations in full-sample units are further modified as follows for $i = 0..7$:

$$x_{\text{Int}_i} = \text{Clip3}(x_{\text{SbInt}_L} - 3, x_{\text{SbInt}_L} + \text{sbWidth} + 4, x_{\text{Int}_i}) \quad (8-775)$$

$$yInt_i = \text{Clip3}(ySbInt_L - 3, ySbInt_L + sbHeight + 4, yInt_i) \quad (8-776)$$

The predicted luma sample value predSampleLX_L is derived as follows:

- If both $x\text{Frac}_L$ and $y\text{Frac}_L$ are equal to 0, the value of predSampleLX_L is derived as follows:

$$\text{predSampleLX}_L = \text{refPicLX}_L[xInt_3][yInt_3] \ll \text{shift3} \quad (8-777)$$

- Otherwise, if $x\text{Frac}_L$ is not equal to 0 and $y\text{Frac}_L$ is equal to 0, the value of predSampleLX_L is derived as follows:

$$\text{predSampleLX}_L = \left(\sum_{i=0}^7 f_L[x\text{Frac}_L][i] * \text{refPicLX}_L[xInt_i][yInt_3] \right) \gg \text{shift1} \quad (8-778)$$

- Otherwise, if $x\text{Frac}_L$ is equal to 0 and $y\text{Frac}_L$ is not equal to 0, the value of predSampleLX_L is derived as follows:

$$\text{predSampleLX}_L = \left(\sum_{i=0}^7 f_L[y\text{Frac}_L][i] * \text{refPicLX}_L[xInt_3][yInt_i] \right) \gg \text{shift1} \quad (8-779)$$

- Otherwise, if $x\text{Frac}_L$ is not equal to 0 and $y\text{Frac}_L$ is not equal to 0, the value of predSampleLX_L is derived as follows:

- The sample array $\text{temp}[n]$ with $n = 0..7$, is derived as follows:

$$\text{temp}[n] = \left(\sum_{i=0}^7 f_L[x\text{Frac}_L][i] * \text{refPicLX}_L[xInt_i][yInt_n] \right) \gg \text{shift1} \quad (8-780)$$

- The predicted luma sample value predSampleLX_L is derived as follows:

$$\text{predSampleLX}_L = \left(\sum_{i=0}^7 f_L[y\text{Frac}_L][i] * \text{temp}[i] \right) \gg \text{shift2} \quad (8-781)$$

Table 8-11 – Specification of the luma interpolation filter coefficients $f_L[p]$ for each 1/16 fractional sample position p .

Fractional sample position p	interpolation filter coefficients							
	$f_L[p][0]$	$f_L[p][1]$	$f_L[p][2]$	$f_L[p][3]$	$f_L[p][4]$	$f_L[p][5]$	$f_L[p][6]$	$f_L[p][7]$

1	0	1	-3	63	4	-2	1	0
2	-1	2	-5	62	8	-3	1	0
3	-1	3	-8	60	13	-4	1	0
4	-1	4	-10	58	17	-5	1	0
5	-1	4	-11	52	26	-8	3	-1
6	-1	3	-9	47	31	-10	4	-1
7	-1	4	-11	45	34	-10	4	-1
8	(hpelldx == 0)							
8		-1	4	-11	40	40	-11	4
8	(hpelldx == 1)							
8		0	3	9	20	20	9	3
9		-1	4	-10	34	45	-11	4
10		-1	4	-10	31	47	-9	3
11		-1	3	-8	26	52	-11	4
12		0	1	-5	17	58	-10	4
13		0	1	-4	13	60	-8	3
14		0	1	-3	8	62	-5	2
15		0	1	-2	4	63	-3	1

Table 8-12 – Specification of the luma interpolation filter coefficients $f_L[p]$ for each $1/16$ fractional sample position p for affine motion mode.

Fractional sample position p	interpolation filter coefficients							
	$f_L[p][0]$	$f_L[p][1]$	$f_L[p][2]$	$f_L[p][3]$	$f_L[p][4]$	$f_L[p][5]$	$f_L[p][6]$	$f_L[p][7]$
1	0	1	-3	63	4	-2	1	0
2	0	1	-5	62	8	-3	1	0
3	0	2	-8	60	13	-4	1	0
4	0	3	-10	58	17	-5	1	0
5	0	3	-11	52	26	-8	2	0
6	0	2	-9	47	31	-10	3	0
7	0	3	-11	45	34	-10	3	0
8	0	3	-11	40	40	-11	3	0
9	0	3	-10	34	45	-11	3	0
10	0	3	-10	31	47	-9	2	0
11	0	2	-8	26	52	-11	3	0
12	0	1	-5	17	58	-10	3	0
13	0	1	-4	13	60	-8	2	0
14	0	1	-3	8	62	-5	1	0
15	0	1	-2	4	63	-3	1	0

8.5.6.3.3 Luma integer sample fetching process

Inputs to this process are:

- a luma location in full-sample units (x_{IntL} , y_{IntL}),
- the luma reference sample array $refPicLXL$,

Output of this process is a predicted luma sample value $predSampleLXL$

The variable shift is set equal to $\text{Max}(2, 14 - \text{BitDepth}_Y)$.

The variable $picW$ is set equal to $pic_width_in_luma_samples$ and the variable $picH$ is set equal to $pic_height_in_luma_samples$.

The luma locations in full-sample units (x_{Int} , y_{Int}) are derived as follows:

$$x_{Int} = \text{Clip3}(0, picW - 1, \text{sps_ref_wraparound_enabled_flag} ? \text{ClipH}((\text{sps_ref_wraparound_offset_minus1} + 1) * \text{MinCbSizeY}, picW, x_{IntL}) : x_{IntL}) \quad (8-782)$$

$$y_{Int} = \text{Clip3}(0, picH - 1, y_{IntL}) \quad (8-783)$$

The predicted luma sample value $predSampleLXL$ is derived as follows:

$$predSampleLXL = refPicLXL[x_{Int}][y_{Int}] \ll shift3 \quad (8-784)$$

8.5.6.3.4 Chroma sample interpolation process

Inputs to this process are:

- a chroma location in full-sample units (x_{IntC} , y_{IntC}),
- a chroma location in 1/32 fractional-sample units (x_{FracC} , y_{FracC}),
- a chroma location in full-sample units (x_{SbIntC} , y_{SbIntC}) specifying the top-left sample of the bounding block for reference sample padding relative to the top-left chroma sample of the reference picture,
- a variable $sbWidth$ specifying the width of the current subblock,

- a variable sbHeight specifying the height of the current subblock,
- the chroma reference sample array refPicLXC.

Output of this process is a predicted chroma sample value predSampleLXC

The variables shift1, shift2 and shift3 are derived as follows:

- The variable shift1 is set equal to $\text{Min}(4, \text{BitDepth}_C - 8)$, the variable shift2 is set equal to 6 and the variable shift3 is set equal to $\text{Max}(2, 14 - \text{BitDepth}_C)$.
- The variable picWC is set equal to $\text{pic_width_in_luma_samples} / \text{SubWidth}_C$ and the variable picHC is set equal to $\text{pic_height_in_luma_samples} / \text{SubHeight}_C$.

The chroma interpolation filter coefficients $f_C[p]$ for each 1/32 fractional sample position p equal to xFracC or yFracC are specified in Table 8-13.

The variable xOffset is set equal to

$(\text{sps_ref_wraparound_offset_minus1} + 1) * \text{MinCbSize}_Y / \text{SubWidth}_C$.

The chroma locations in full-sample units ($xInt_i, yInt_i$) are derived as follows for $i = 0..3$:

- If subpic_treated_as_pic_flag[SubPicIdx] is equal to 1, the following applies:

$$xInt_i = \text{Clip3}(\text{SubPicLeftBoundaryPos} / \text{SubWidth}_C, \text{SubPicRightBoundaryPos} / \text{SubWidth}_C, xInt_L + i) \quad (8-785)$$

$$yInt_i = \text{Clip3}(\text{SubPicTopBoundaryPos} / \text{SubHeight}_C, \text{SubPicBotBoundaryPos} / \text{SubHeight}_C, yInt_L + i) \quad (8-786)$$

- Otherwise (subpic_treated_as_pic_flag[SubPicIdx] is equal to 0), the following applies:

$$xInt_i = \text{Clip3}(0, \text{picWC} - 1, \text{sps_ref_wraparound_enabled_flag} ? \text{ClipH}(xOffset, \text{picWC}, xInt_C + i - 1) : xInt_C + i - 1) \quad (8-787)$$

$$yInt_i = \text{Clip3}(0, \text{picHC} - 1, yInt_C + i - 1) \quad (8-788)$$

The chroma locations in full-sample units ($xInt_i, yInt_i$) are further modified as follows for $i = 0..3$:

$$xInt_i = \text{Clip3}(xSbIntC - 1, xSbIntC + sbWidth + 2, xInt_i) \quad (8-789)$$

$$yInt_i = \text{Clip3}(ySbIntC - 1, ySbIntC + sbHeight + 2, yInt_i) \quad (8-790)$$

The predicted chroma sample value predSampleLXC is derived as follows:

- If both $xFracc$ and $yFracc$ are equal to 0, the value of predSampleLXC is derived as follows:

$$\text{predSampleLXC} = \text{refPicLXC}[xInt_i][yInt_i] \ll \text{shift3} \quad (8-791)$$

- Otherwise, if $xFracc$ is not equal to 0 and $yFracc$ is equal to 0, the value of predSampleLXC is derived as follows:

$$\text{predSampleLXC} = \left(\sum_{i=0}^3 fc[xFracc][i] * \text{refPicLXC}[xInt_i][yInt_i] \right) \gg \text{shift1} \quad (8-792)$$

- Otherwise, if $xFracc$ is equal to 0 and $yFracc$ is not equal to 0, the value of predSampleLXC is derived as follows:

$$\text{predSampleLXC} = \left(\sum_{i=0}^3 fc[yFracc][i] * \text{refPicLXC}[xInt_i][yInt_i] \right) \gg \text{shift1} \quad (8-793)$$

- Otherwise, if $xFracc$ is not equal to 0 and $yFracc$ is not equal to 0, the value of predSampleLXC is derived as follows:

- The sample array $\text{temp}[n]$ with $n = 0..3$, is derived as follows:

$$\text{temp}[n] = \left(\sum_{i=0}^3 fc[xFracc][i] * \text{refPicLXC}[xInt_i][yInt_n] \right) \gg \text{shift1} \quad (8-794)$$

- The predicted chroma sample value predSampleLXC is derived as follows:

$$\begin{aligned} \text{predSampleLXC} &= (fc[yFracc][0] * \text{temp}[0] + \\ &\quad fc[yFracc][1] * \text{temp}[1] + \\ &\quad fc[yFracc][2] * \text{temp}[2] + \\ &\quad fc[yFracc][3] * \text{temp}[3]) \gg \text{shift2} \end{aligned} \quad (8-795)$$

Table 8-13 – Specification of the chroma interpolation filter coefficients $fc[p]$ for each 1/32 fractional sample position p .

Fractional sample position p	interpolation filter coefficients			
	fc[p][0]	fc[p][1]	fc[p][2]	fc[p][3]
1	-1	63	2	0
2	-2	62	4	0
3	-2	60	7	-1
4	-2	58	10	-2
5	-3	57	12	-2
6	-4	56	14	-2
7	-4	55	15	-2
8	-4	54	16	-2
9	-5	53	18	-2
10	-6	52	20	-2
11	-6	49	24	-3
12	-6	46	28	-4
13	-5	44	29	-4
14	-4	42	30	-4
15	-4	39	33	-4
16	-4	36	36	-4
17	-4	33	39	-4
18	-4	30	42	-4
19	-4	29	44	-5
20	-4	28	46	-6
21	-3	24	49	-6
22	-2	20	52	-6
23	-2	18	53	-5
24	-2	16	54	-4
25	-2	15	55	-4
26	-2	14	56	-4
27	-2	12	57	-3
28	-2	10	58	-2
29	-1	7	60	-2
30	0	4	62	-2
31	0	2	63	-1

2.6 Embodiments employing sub-pictures

[00179] With the current syntax design of sub-pictures in an existing implementation, the locations and dimensions of sub-pictures are derived as below:

subpics_present_flag	u(1)
if(subpics_present_flag) {	
max_subpics_minus1	u(8)
subpic_grid_col_width_minus1	u(v)
subpic_grid_row_height_minus1	u(v)
for(i = 0; i < NumSubPicGridRows; i++)	
for(j = 0; j < NumSubPicGridCols; j++)	
subpic_grid_idx[i][j]	u(v)
for(i = 0; i <= NumSubPics; i++) {	
subpic_treated_as_pic_flag[i]	u(1)
loop_filter_across_subpic_enabled_flag[i]	u(1)
}	
}	

subpics_present_flag equal to 1 indicates that subpicture parameters are present in the present in the SPS RBSP syntax. **subpics_present_flag** equal to 0 indicates that subpicture parameters are not present in the present in the SPS RBSP syntax.

NOTE 2 – When a bitstream is the result of a sub-bitstream extraction process and contains only a subset of the subpictures of the input bitstream to the sub-bitstream extraction process, it might be required to set the value of **subpics_present_flag** equal to 1 in the RBSP of the SPSs.

max_subpics_minus1 plus 1 specifies the maximum number of subpictures that may be present in the CVS. **max_subpics_minus1** shall be in the range of 0 to 254. The value of 255 is reserved for future use by ITU-T | ISO/IEC.

subpic_grid_col_width_minus1 plus 1 specifies the width of each element of the subpicture identifier grid in units of 4 samples. The length of the syntax element is

$\text{Ceil}(\text{Log2}(\text{pic_width_max_in_luma_samples} / 4))$ bits.

The variable NumSubPicGridCols is derived as follows:

$$\text{NumSubPicGridCols} = \frac{(\text{pic_width_max_in_luma_samples} + \text{subpic_grid_col_width_minus1} * 4 + 3)}{(\text{subpic_grid_col_width_minus1} * 4 + 4)} \quad (7-5)$$

subpic_grid_row_height_minus1 plus 1 specifies the height of each element of the subpicture identifier grid in units of 4 samples. The length of the syntax element is

$\text{Ceil}(\text{Log2}(\text{pic_height_max_in_luma_samples} / 4))$ bits.

The variable NumSubPicGridRows is derived as follows:

$$\text{NumSubPicGridRows} = \frac{(\text{pic_height_max_in_luma_samples} + \text{subpic_grid_row_height_minus1} * 4 + 3)}{(\text{subpic_grid_row_height_minus1} * 4 + 4)} \quad (7-6)$$

subpic_grid_idx[i][j] specifies the subpicture index of the grid position (i, j). The length of the syntax element is $\text{Ceil}(\text{Log2}(\text{max_subpics_minus1} + 1))$ bits.

The variables SubPicTop[subpic_grid_idx[i][j]], SubPicLeft[subpic_grid_idx[i][j]], SubPicWidth[subpic_grid_idx[i][j]], SubPicHeight[subpic_grid_idx[i][j]], and NumSubPics are derived as follows:

```

NumSubPics = 0
for( i = 0; i < NumSubPicGridRows; i++ ) {
    for( j = 0; j < NumSubPicGridCols; j++ ) {
        if ( i == 0 )
            SubPicTop[ subpic_grid_idx[ i ][ j ] ] = 0
        else if( subpic_grid_idx[ i ][ j ] != subpic_grid_idx[ i - 1 ][ j ] ) {
            SubPicTop[ subpic_grid_idx[ i ][ j ] ] = i
            SubPicHeight[ subpic_grid_idx[ i - 1 ][ j ] ] =
i - SubPicTop[ subpic_grid_idx[ i - 1 ][ j ] ]
        }
        if ( j == 0 )
            SubPicLeft[ subpic_grid_idx[ i ][ j ] ] = 0
        else if( subpic_grid_idx[ i ][ j ] != subpic_grid_idx[ i ][ j - 1 ] ) {
            SubPicLeft[ subpic_grid_idx[ i ][ j ] ] = j
            SubPicWidth[ subpic_grid_idx[ i ][ j ] ] =
j - SubPicLeft[ subpic_grid_idx[ i ][ j - 1 ] ]
        }
        if ( i == NumSubPicGridRows - 1 )
            SubPicHeight[ subpic_grid_idx[ i ][ j ] ] =
i - SubPicTop[ subpic_grid_idx[ i - 1 ][ j ] ] + 1
    }
}

```

(7-7)

if (j

```

== NumSubPicGridRows - 1)
    SubPicWidth[ subpic_grid_idx[ i ][ j ] ] =
j - SubPicLeft[ subpic_grid_idx[ i ][ j - 1 ] ] + 1
    if( subpic_grid_idx[ i ][ j ] > NumSubPics)
        NumSubPics = subpic_grid_idx[ i ][ j ]
    }
}

```

subpic_treated_as_pic_flag[i] equal to 1 specifies that the i-th subpicture of each coded picture in the CVS is treated as a picture in the decoding process excluding in-loop filtering operations.

subpic_treated_as_pic_flag[i] equal to 0 specifies that the i-th subpicture of each coded picture in the CVS is not treated as a picture in the decoding process excluding in-loop filtering operations. When not present, the value of **subpic_treated_as_pic_flag[i]** is inferred to be equal to 0.

2.7 Combined Inter-Intra Prediction (CIIP)

[00180] Combined Inter-Intra Prediction (CIIP) is adopted into VVC as a special merge candidate. It can only be enabled for a WxH block with $W \leq 64$ and $H \leq 64$.

3. Drawbacks of Existing Implementations

[00181] In the current design of VVC, ATMVP has following problems:

- 1) Whether to apply ATMVP is mismatched in slice level and in CU level;
- 2) In slice header, ATMVP may be enabled even if TMVP is disabled. Meanwhile, ATMVP flag is signaled before TMVP flag.
- 3) Masking is always done without considering whether MV is compressed;
- 4) The valid corresponding region may be too large;
- 5) The derivation of TMV is too complicated;
- 6) ATMVP may be unavailable in some cases and a better default MV is desirable.
- 7) The MV scaling methods in ATMVP may not be efficient;
- 8) ATMVP should consider the CPR cases;
- 9) A default zero affine merge candidate may be put into the list even when affine prediction is disabled.
- 10) Current picture is treated as a long-term reference picture while other pictures are treated as short-reference picture. For both ATMVP and TMVP candidates, the motion information

from a temporal block in the collocated picture will be scaled to a reference picture with fixed reference index (i.e., 0 for each reference picture list in the current design). However, when CPR mode is enabled, the current picture is also treated as a reference picture and the current picture may be added to the reference picture list 0 (RefPicList0) with index equal to 0.

- a. For TMVP, if the temporal block is coded with CPR mode and reference picture of RefPicList0 is a short reference picture, the TMVP candidate is set to be unavailable.
- b. If the reference picture of the RefPicList0 with index equal to 0 is the current picture and current picture is the Intra Random Access Point (IRAP) picture, ATMVP candidate is set to be unavailable.
- c. For a ATMVP sub-block within a block, when deriving the sub-block's motion information from a temporal block, if the temporal block is coded with CPR mode, default ATMVP candidate (derived from a temporal block identified by the starting TMV and center position of the current block) is used to fill in the sub-block's motion information.

11) MV is right shifted to the integer precision, but not following the rounding rule in VVC.

12) The MV(MV_x, MV_y) used in ATMVP to locate a corresponding block in a different picture (e.g., the TMV in 0) is directly used since it points to the collocated picture. This is based on the assumption that all pictures are in the same resolution. However, when RPR is enabled, different picture resolutions may be utilized. Similar issues are existing for identification of corresponding blocks in a collocate picture to derive sub-block motion information.

13) If one block width or height is greater than 32, and max transform block size is 32, for a CIIP coded block, the intra prediction signal is generated with the CU size; while the inter prediction signal is generated with the TU size (recursively splitting current block to multiple 32x32 blocks). Using the CU to derive intra prediction signal results in less efficiency.

[00182] There are some problems with current design. Firstly, if the reference picture of the RefPicList0 with index equal to 0 is the current picture and current picture is not an IRAP picture, ATMVP procedure is still invoked, but it couldn't find any available ATMVP candidates since none of temporal motion vectors could be scaled to the current picture.

4. Examples of embodiments and techniques

[00183] The itemized list of techniques and embodiments below should be considered as examples to explain general concepts. These techniques should not be interpreted in a narrow way. Furthermore, these techniques can be combined in any manner in encoder or decoder embodiments.

1. Whether TMVP is allowed and/or whether CPR is used should be considered to decide/parse the maximum number of candidates in the sub-block merge candidate list and/or to decide whether ATMVP candidate should be added to a candidate list. The maximum number of candidates in the sub-block merge candidate list is denoted as ML.
 - a) In one example, ATMVP is inferred to be not applicable if either ATMVP usage flag is off (equal to 0), or TMVP is disabled, in the determination of or parsing the maximum number of candidates in the sub-block merge candidate list.
 - i. In one example, the ATMVP usage flag is on (equal to 1) and TMVP is disabled, the ATMVP candidate is not added to the sub-block merge candidate list or the ATMVP candidate list.
 - ii. In one example, the ATMVP usage flag is on (equal to 1) and TMVP is disabled, and the affine usage flag is off (equal to 0), then ML is set equal to 0, meaning that sub-block merge is not applicable.
 - iii. In one example, the ATMVP usage flag is on (equal to 1) and TMVP is enabled, and the affine usage flag is off (equal to 0), then ML is set equal to 1.
 - b) In one example, ATMVP is inferred to be not applicable if either ATMVP usage flag is off (equal to 0), or the collocated reference picture of the current picture is the current picture itself, when decide or parse the maximum number of candidates in the sub-block merge candidate list.
 - i. In one example, the ATMVP usage flag is on (equal to 1) and the collocated reference picture of the current picture is the current picture itself, the ATMVP candidate is not added to the sub-block merge candidate list or the ATMVP candidate list.

- ii. In one example, the ATMVP usage flag is on (equal to 1), and the collocated reference picture of the current picture is the current picture itself, and the affine usage flag is off (equal to 0), then ML is set equal to 0, meaning that sub-block merge is not applicable.
 - iii. In one example, the ATMVP usage flag is on (equal to 1), and the collocated reference picture of the current picture is not the current picture itself, and the affine usage flag is off (equal to 0), then ML is set equal to 1.
- c) In one example, ATMVP is inferred to be not applicable if ATMVP usage flag is off (equal to 0), or the reference picture with reference picture index 0 in reference list 0 is the current picture itself, when decide or parse the maximum number of candidates in the sub-block merge candidate list.
- i. In one example, the ATMVP usage flag is on (equal to 1) and the collocated reference picture with reference picture index 0 in reference list 0 is the current picture itself, the ATMVP candidate is not added to the sub-block merge candidate list or the ATMVP candidate list.
 - ii. In one example, the ATMVP usage flag is on (equal to 1), and the reference picture with reference picture index 0 in reference list 0 is the current picture itself, and the affine usage flag is off (equal to 0), then ML is set equal to 0, meaning that sub-block merge is not applicable.
 - iii. In one example, the ATMVP usage flag is on (equal to 1) and the reference picture with reference picture index 0 in reference list 0 is not the current picture itself, and the affine usage flag is off (equal to 0), then ML is set equal to 1.
- d) In one example, ATMVP is inferred to be not applicable if ATMVP usage flag is off (equal to 0), or the reference picture with reference picture index 0 in reference list 1 is the current picture itself, when decide or parse the maximum number of candidates in the sub-block merge candidate list.
- i. In one example, the ATMVP usage flag is on (equal to 1) and the collocated reference picture with reference picture index 0 in reference list 1 is the current picture itself, the ATMVP candidate is not added to the sub-block merge candidate list or the ATMVP candidate list.

- ii. In one example, the ATMVP usage flag is on (equal to 1), and the reference picture with reference picture index 0 in reference list 1 is the current picture itself, and the affine usage flag is off (equal to 0), then ML is set equal to 0, meaning that sub-block merge is not applicable.
 - iii. In one example, the ATMVP usage flag is on (equal to 1) and the reference picture with reference picture index 0 in reference list 1 is not the current picture itself, and the affine usage flag is off (equal to 0), then ML is set equal to 1.
- 2. It is proposed that ATMVP is disabled implicitly and no ATMVP flag is signaled if TMVP is disabled at slice/tile/picture level.
 - a) In one example, ATMVP flag is signaled after the TMVP flag in slice header/tile header/PPS.
 - b) In one example, ATMVP or/and TMVP flag may be not signaled in slice header/tile header/PPS and is only signaled in SPS header.
- 3. Whether and how to do mask the corresponding position in ATMVP depends on whether and how MVs are compressed. Suppose (x_N, y_N) is a corresponding position calculated with the coordinator of current block/sub-block and a starting motion vector (e.g., TMV) in the collocated picture
 - a) In one example, (x_N, y_N) is not masked if MVs are not required to be compressed (e.g. `sps_disable_motioncompression` signaled in SPS is 1); Otherwise, (MVs are required to be compressed) (x_N, y_N) is masked as $x_N = x_N \& \text{MASK}$, $y_N = y_N \& \text{MASK}$, where MASK is an integer equal to $\sim(2^M - 1)$, and M can be integers such as 3 or 4.
 - b) Assume the MV compression method for MV storage result in each $2^K \times 2^K$ block share the same motion information and the mask in the ATMVP process is defined as $\sim(2^M - 1)$. It is proposed that K may be unequal to M, e.g., $M = K + 1$.
 - c) The MASK used in ATMVP and TMVP may be the same, or they may be different.
- 4. In one example, the MV compression method can be flexible.

- a) In one example, the MV compression method can be selected between no compression, 8×8 compression ($M=3$ in Bullet 3.a), or 16×16 compression ($M=4$ in Bullet 3.a)
 - b) In one example, the MV compression method may be signaled in VPS/SPS/PPS/slice header/tile group header.
 - c) In one example, the MV compression method can be set differently in different standard profiles/levels/tiers.
5. The valid corresponding region in ATMVP may adaptive;
- a) For example, the valid corresponding region may depend on the width and height of the current block;
 - b) For example, the valid corresponding region may depend on the MV compression method;
 - i. In one example, the valid corresponding region is smaller if MV compression method is not used; the valid corresponding region is larger if MV compression method is used.
6. The valid corresponding region in ATMVP may be based on a basic region with size $M \times N$ smaller than a CTU region. For example, the CTU size in VTM-3.0 is 128×128 , and the basic region size may be 64×64 . Suppose the width and height of the current block is W and H .
- a) In one example, if $W \leq M$ and $H \leq N$, meaning that the current block is inside a basic region, then the valid corresponding region in ATMVP is the collocated basic region and the extension in the collocated picture. FIG. 27 shows an example.
 - i. For example, suppose the top-left position of the collocated basic region is (x_{BR}, y_{BR}) , then the corresponding position at (x_N, y_N) will be clipped into the valid region $x_{BR} \leq x_N < x_{BR} + M + 4$; $y_{BR} \leq y_N < y_{BR} + N$.

[00184] FIG. 27 shows an example embodiment of the proposed valid region when the current block is inside a Basic Region (BR).

[00185] FIG. 28 shows an example embodiment of a valid region when the current block is not inside a basic region.

- b) In one example, if $W > M$ and $H > N$, meaning that the current block is not inside a basic region, then the current block is divided into several parts. Each part has an individual valid corresponding region in ATMVP. For a position A in the current block, its corresponding position B in the collocated block should be within the valid corresponding region of the part in which the position A locates.
 - i. For example, the current block is divided into non-overlapped basic regions. The valid corresponding region for one basic region is its collocated basic region and the extension in the collocated picture. FIG. 28 shows an example.
 - 1. For example, suppose position A in the current block is in one basic region R . The collocated basic region of R in the collocated picture is denoted as CR . The corresponding position of A in the collocated block is position B . The top-left position of CR is (x_{CR}, y_{CR}) , then position B at (x_N, y_N) will be clipped into the valid region $x_{CR} \leq x_N < x_{CR} + M + 4$; $y_{CR} \leq y_N < y_{CR} + N$.
- 7. It is proposed the motion vector used in ATMVP to locate a corresponding block in a different picture (e.g., the TMV in 2.3.5.1.2) can be derived as:
 - a) In one example, TMV is always set equal to a default MV such as $(0, 0)$.
 - i. In one example, the default MV is signaled in VPS/SPS/PPS/slice header/tile group header/CTU/CU.
 - b) In one example, TMV is set to be one MV stored in the HMVP table with the following methods;
 - i. If HMVP list is empty, TMV is set equal to a default MV such as $(0, 0)$
 - ii. Otherwise (HMVP list is not empty),
 - 1. TMV may be set to equal to the first element stored in the HMVP table;
 - 2. Alternatively, TMV may be set to equal to the last element stored in the HMVP table;
 - 3. Alternatively, TMV may only be set equal to a specific MV stored in the HMVP table;
 - a. In one example, the specific MV refers to reference list 0.

- b. In one example, the specific MV refers to reference list 1.
 - c. In one example, the specific MV refers to a specific reference picture in reference list 0, such as the reference picture with index 0.
 - d. In one example, the specific MV refers to a specific reference picture in reference list 1, such as the reference picture with index 0.
 - e. In one example, the specific MV refers to the collocated picture.
 - 4. Alternatively, TMV may be set equal to the default MV if the specific MV (e.g., mentioned in bullet 3.) stored in the HMVP table cannot be found;
 - a. In one example, only the first element stored in the HMVP table is searched to find the specific MV.
 - b. In one example, only the last element stored in the HMVP table is searched to find the specific MV.
 - c. In one example, some or all elements stored in the HMVP table is searched to find the specific MV.
 - 5. Alternatively, furthermore, TMV obtained from the HMVP cannot refer to the current picture itself.
 - 6. Alternatively, furthermore, TMV obtained from the HMVP table may be scaled to the collocated picture if it does not refer to it.
- c) In one example, TMV is set to be one MV of one specific neighbouring block. No other neighbouring blocks are involved.
- i. The specific neighbouring block may be block A0, A1, B0, B1, B2 in FIG. 22.
 - ii. TMV may be set equal to the default MV if
 - 1. The specific neighbouring block does not exist;
 - 2. The specific neighbouring block is not inter-coded;
 - iii. TMV may only be set equal to a specific MV stored in the specific neighbouring block;

1. In one example, the specific MV refers to reference list 0.
 2. In one example, the specific MV refers to reference list 1.
 3. In one example, the specific MV refers to a specific reference picture in reference list 0, such as the reference picture with index 0.
 4. In one example, the specific MV refers to a specific reference picture in reference list 1, such as the reference picture with index 0.
 5. In one example, the specific MV refers to the collocated picture.
 6. TMV may be set equal to the default MV if the specific MV stored in the specific neighbouring block cannot be found;
 - iv. TMV obtained from the specific neighbouring block may be scaled to the collocated picture if it does not refer to it.
 - v. TMV obtained from the specific neighbouring block cannot refer to the current picture itself.
8. MVdefault0 and MVdefault1 used in ATMVP as disclosed in 2.3.5.1.2 may be derived as
- a) In one example, MVdefault0 and MVdefault1 are set equal to (0,0);
 - b) In one example, MVdefaultX (X=0 or 1) is derived from HMVP,
 - i. If HMVP list is empty, MVdefaultX is set equal to a predefined default MV such as (0, 0).
 1. The predefined default MV may be signaled in VPS/SPS/PPS/slice header/tile group header/CTU/CU.
 - ii. Otherwise (HMVP list is not empty),
 1. MVdefaultX may be set to equal to the first element stored in the HMVP table;
 2. MVdefaultX may be set to equal to the last element stored in the HMVP table;
 3. MVdefaultX may only be set equal to a specific MV stored in the HMVP table;
 - a. In one example, the specific MV refers to reference list X.

- b. In one example, the specific MV refers to a specific reference picture in reference list X, such as the reference picture with index 0.
 - 4. MVdefaultX may be set equal to the predefined default MV if the specific MV stored in the HMVP table cannot be found;
 - a. In one example, only the first element stored in the HMVP table is searched.
 - b. In one example, only the last element stored in the HMVP table is searched.
 - c. In one example, some or all elements stored in the HMVP table is searched.
 - 5. MVdefaultX obtained from the HMVP table may be scaled to the collocated picture if it does not refer to it.
 - 6. MVdefaultX obtained from the HMVP cannot refer to the current picture itself.
- c) In one example, MVdefaultX (X=0 or 1) is derived from a neighbouring block.
 - i. The neighbouring blocks may include block A0, A1, B0, B1, B2 in FIG. 22.
 - 1. For example, only one of these blocks is used to derive MVdefaultX.
 - 2. Alternatively, some or all of these blocks are used to derive MVdefaultX.
 - a. These blocks are checked in order until a valid MVdefaultX is found.
 - 3. If no valid MVdefaultX can be found from selected one or more neighbouring blocks, it is set equal to a predefined default MV such as (0, 0).
 - a. The predefined default MV may be signaled in VPS/SPS/PPS/slice header/tile group header/CTU/CU.
 - ii. No valid MVdefaultX can be found from a specific neighbouring block if
 - 1. The specific neighbouring block does not exist;
 - 2. The specific neighbouring block is not inter-coded;

- iii. MVdefaultX may only be set equal to a specific MV stored in the specific neighbouring block;
 - 1. In one example, the specific MV refers to reference list X.
 - 2. In one example, the specific MV refers to a specific reference picture in reference list X., such as the reference picture with index 0
 - iv. MVdefaultX obtained from the specific neighbouring block may be scaled to a specific reference picture, such as the reference picture with index 0 in reference list X.
 - v. MVdefaultX obtained from the specific neighbouring block cannot refer to the current picture itself.
9. For either sub-block or non-sub-block ATMVP candidate, if a temporal block for a sub-block/whole block in the collocated picture is coded with CPR mode, a default motion candidate may be utilized instead.
- a) In one example, the default motion candidate may be defined as the motion candidate associated with the center position of current block (e.g., MVdefault0 and/or MVdefault1 used in ATMVP as disclosed in 2.3.5.1.2).
 - b) In one example, the default motion candidate may be defined as (0, 0) motion vector and reference picture index equal to 0 for both reference picture lists, if available.
10. It is proposed that the default motion information for the ATMVP process (e.g., MVdefault0 and MVdefault1 used in ATMVP as disclosed in 2.3.5.1.2) may be derived based on the location of a position that is used in the sub-block motion information derivation process. With this proposed method, for that sub-block, there is no need to further derive motion information since the default motion information will be directly assigned.
- a) In one example, instead of using the center position of the current block, the center position of a sub-block (e.g., a center sub-block) within the current block may be utilized.
 - b) An example of existing and proposed implementations are shown in FIGS. 29A and 29B, respectively.

11. It is proposed that the ATMVP candidate is always available with the following methods:

- a) Suppose the center point of the current block is (x_0, y_0) , then locate a corresponding position of (x_0, y_0) as $M = (x_0 + MV_x, y_0 + MV_y)$ in the collocated picture. Find the block Z covering M. If Z is intra-coded, then MVdefault0, MVdefault1 are derived by some methods proposed in Item 6.
- b) Alternatively, Block Z is not located to get the motion information, some methods proposed in Item 8 are directly applied to get MVdefault0 and MVdefault1.
- c) Alternatively, the default motion candidate used in ATMVP process is always available. If it is set to unavailable based on current design (e.g., the temporal block is intra coded), other motion vectors may be utilized instead for the default motion candidate.
 - i. In one example, the solutions in international application PCT/CN2018/124639, incorporated by reference herein, may be applied.
- d) Alternatively, furthermore, whether ATMVP candidate is always available depend on other high-level syntax information.
 - i. In one example, only when the ATMVP enabling flag at slice/tile/picture header or other video units is set to true is inferred to be true, the ATMVP candidate may be always set to be available.
 - ii. In one example, the above methods may be only applicable when ATMVP enabling flag in slice header/picture header or other video units is set to true and current picture is not an IRAP picture and current picture is not inserted to RefPicList0 with reference index equal to 0.
- e) A fixed index or a fixed group of indices is assigned to ATMVP candidates. When ATMVP candidates are always unavailable, the fixed index/group index may be inferred to other kinds of motion candidates (such as affine candidates).

12. It is proposed that whether the Zero motion affine merge candidate is put into the sub-block merge candidate list should depend on whether affine prediction is enabled.

- a) For example, if affine usage flag is off (**sps_affine_enabled_flag** is equal to 0), the Zero motion affine merge candidate is not put into the sub-block merge candidate list.

- b) Alternatively, furthermore, default motion vector candidates which are non-affine candidate are added instead.
13. It is proposed that non-affine padding candidate may be put into the sub-block merge candidate list.
- a) Zero motion non-affine padding candidate may be added if the sub-block merge candidate list is not fulfilled.
 - b) When such a padding candidate is chosen, the `affine_flag` of the current block should be set to be 0.
 - c) Alternatively, Zero motion non-affine padding candidate is put into the sub-block merge candidate list if the sub-block merge candidate list is not fulfilled and `affine_usage` flag is off.
14. Suppose MV0 and MV1 represent the MVs in reference list 0 and reference list 1 of the block covering a corresponding position (e.g., MV0 and MV1 may be MVZ_0 and MVZ_1 or MVZS_0 and MVZS_1 described in section 2.3.5.1.2). MV0' and MV1' represent the MVs in reference list 0 and reference list 1 to be derived for the current block or sub-block. Then the MV0' and MV1' should be derived by scaling.
- a) MV0, if the collocated picture is in reference list 1;
 - b) MV1, if the collocated picture is in reference list 0.
15. The ATMVP and/or TMVP enabling/disabling flag may be inferred to be false for a slice/tile or other kinds of video units when current picture is treated as a reference picture with index set to M (e.g., 0) in a reference picture list X (`PicRefListX`, e.g., $X = 0$). Here, M may be equal to the target reference picture index that motion information of a temporal block shall be scaled to for `PicRefListX` during ATMVP/TMVP processes.
- a) Alternatively, furthermore, the above method is only applicable when the current picture is an Intra Random Access Point (IRAP) picture.
 - b) In one example, ATMVP and/or TMVP enabling/disabling flag may be inferred to be false when current picture is treated as a reference picture with index set to M (e.g., 0) in a `PicRefListX` and/or current picture is treated as a reference picture with index set to N (e.g., 0) in a `PicRefListY`. The variable M and N represent the target reference picture index used in TMVP or ATMVP process.

- c) For the ATMVP process, it is restricted that a confirming bitstream shall follow the rule that the collocated picture wherein motion information of current block is derived from shall not be the current picture.
 - d) Alternatively, when the above conditions are true, the ATMVP or TMVP process is not invoked.
16. It is proposed that if a reference picture with index set to M (e.g., 0) in a reference picture list X (PicRefList X , e.g., $X = 0$) for the current block is the current picture, ATMVP may be still enabled for this block.
- a) In one example, all sub-blocks' motion information are point to the current picture.
 - b) In one example, when obtaining the sub-block's motion information from a temporal block, the temporal block shall be coded with at least one reference picture pointing to the current picture of the temporal block.
 - c) In one example, when obtaining the sub-block's motion information from a temporal block, no scaling operations are applied.
17. Coding methods of sub-block merge index are aligned regardless the usage of ATMVP or not.
- a) In one example, for the first L bins, they are context coded. For the remaining bins, they are bypass coded. In one example, L is set to 1.
 - b) Alternatively, for all bins, they are context coded.
18. The $MV(MV_x, MV_y)$ used in ATMVP to locate a corresponding block in a different picture (e.g., the TMV in 0) may be right-shifted to the integer precision (denoted as (MV'_x, MV'_y)) with the same rounding method as in the MV scaling process.
- a) Alternatively, the MV used in ATMVP to locate a corresponding block in a different picture (e.g., the TMV in 0) may be right-shifted to the integer precision with the same rounding method as in the MV averaging process.
 - b) Alternatively, the MV used in ATMVP to locate a corresponding block in a different picture (e.g., the TMV in 0) may be right-shifted to the integer precision with the same rounding method as in the adaptive MV resolution (AMVR) process.
19. The $MV(MV_x, MV_y)$ used in ATMVP to locate a corresponding block in a different picture (e.g., the TMV in 0) may be right-shifted to the integer precision (denoted as (MV'_x, MV'_y)) by rounding toward zero.

- a) For example, $MV_x' = (MV_x + ((1 < N) >> 1) - (MV_x \geq 0 ? 1 : 0)) >> N$; N is an integer presenting the MV resolution, e.g. N=4.
 - i. For example, $MV_x' = (MV_x + (MV_x \geq 0 ? 7 : 8)) >> 4$.
 - b) For example, $MV_y' = (MV_y + ((1 < N) >> 1) - (MV_y \geq 0 ? 1 : 0)) >> N$; N is an integer presenting the MV resolution, e.g. N=4.
 - i. For example, $MV_y' = (MV_y + (MV_y \geq 0 ? 7 : 8)) >> 4$.
20. In one example, MV(MV_x, MV_y) in bullet 18 and bullet 19 is used to locate a corresponding block to derive the default motion information used in ATMVP, such as using the center position of the sub-block and the shifted MV, or using the top-left position of current block and the shifted MV.
- a) In one example, MV(MV_x, MV_y) is used to locate a corresponding block to derive the motion information of a sub-block in current block during the ATMVP process, such as using the center position of the sub-block and the shifted MV.
21. The proposed methods in bullet 18, 19, 20 may be also applied to other coding tools that require to locate a reference block in a different picture or current picture with a motion vector.
22. The MV(MV_x, MV_y) used in ATMVP to locate a corresponding block in a different picture (e.g., the TMV in 0) may be scaled even it points to the collocated picture.
- a) In one example, if the width and/or height of the collocated picture (or the conformance window in it) is different from that of the current picture (or the conformance window in it), the MV may be scaled.
 - b) Suppose the width and height of (the conformance window) of the collocated picture are denoted as W1 and H1, respectively. The width and height of (the conformance window of) the current picture are denoted as W2 and H2, respectively. Then MV(MV_x, MV_y) may be scaled as $MV_x' = MV_x * W1 / W2$ and $MV_y' = MV_y * H1 / H2$.
23. The center point of the current block (such as position (x0, y0) in 2.3.5.1.2) used to derive motion information in the ATMVP process may be further modified by scaling and/or adding offsets.

- a) In one example, if the width and/or height of the collocated picture (or the conformance window in it) is different from that of the current picture (or the conformance window in it), the center point may be further modified.
 - b) Suppose the top-left position of the conformance window in the collocated picture are denoted as $X1$ and $Y1$. The top-left position of the conformance window defined in the current picture are denoted as $X2$ and $Y2$. The width and height of (the conformance window of) the collocated picture are denoted as $W1$ and $H1$, respectively. The width and height of (the conformance window) of the current picture are denoted as $W2$ and $H2$, respectively. Then $(x0, y0)$ may be modified as $x0' = (x0 - X2) * W1 / W2 + X1$ and $y0' = (y0 - Y2) * H1 / H2 + Y1$.
 - i. Alternatively, $x0' = x0 * W1 / W2$, $y0' = y0 * H1 / H2$.
24. The corresponding position (such as position M in 2.3.5.1.2) used to derive motion information in the ATMVP process may be further modified by scaling and/or adding offsets
- a) In one example, if the width and/or height of the collocated picture (or the conformance window in it) is different from that of the current picture (or the conformance window in it), the corresponding position may be further modified.
 - b) Suppose the top-left position of the conformance window in the collocated picture are denoted as $X1$ and $Y1$. The top-left position of the conformance window defined in the current picture are denoted as $X2$ and $Y2$. The width and height of (the conformance window of) the collocated picture are denoted as $W1$ and $H1$, respectively. The width and height of (the conformance window) of the current picture are denoted as $W2$ and $H2$, respectively. Then $M(x, y)$ may be modified as $x' = (x - X2) * W1 / W2 + X1$ and $y' = (y - Y2) * H1 / H2 + Y1$.
 - i. Alternatively, $x' = x * W1 / W2$, $y' = y * H1 / H2$.

Sub-picture related

25. In one example, the width of a sub-picture S ending at the $(j-1)$ column may be set equal to j minus the left-most column of the sub-picture S, if the position (i, j) and $(i, j-1)$ belong to different sub-pictures.

a) An embodiment based on an existing implementation is highlighted as below.

```

NumSubPics = 0
for( i = 0; i < NumSubPicGridRows; i++ ) {
    for( j = 0; j < NumSubPicGridCols; j++ ) {
        if ( i == 0 )
            SubPicTop[ subpic_grid_idx[ i ][ j ] ] = 0
        else if( subpic_grid_idx[ i ][ j ] != subpic_grid_idx[ i - 1 ][ j ] ) {
            SubPicTop[ subpic_grid_idx[ i ][ j ] ] = i
            SubPicHeight[ subpic_grid_idx[ i - 1 ][ j ] ] =
i - SubPicTop[ subpic_grid_idx[ i - 1 ][ j ] ]
        }
        if ( j == 0 )
            SubPicLeft[ subpic_grid_idx[ i ][ j ] ] = 0
        else if ( subpic_grid_idx[ i ][ j ] != subpic_grid_idx[ i ][ j - 1 ] ) {
            SubPicLeft[ subpic_grid_idx[ i ][ j ] ] = j
            SubPicWidth[ subpic_grid_idx[ i ][ j - 1 ] ] =
j - SubPicLeft[ subpic_grid_idx[ i ][ j - 1 ] ]
        }
        if ( i == NumSubPicGridRows - 1 )
            SubPicHeight[ subpic_grid_idx[ i ][ j ] ] =
i - SubPicTop[ subpic_grid_idx[ i - 1 ][ j ] ] + 1
        == NumSubPicGridRows - 1 )
            SubPicWidth[ subpic_grid_idx[ i ][ j ] ] =
j - SubPicLeft[ subpic_grid_idx[ i ][ j - 1 ] ] + 1
        if( subpic_grid_idx[ i ][ j ] > NumSubPics )
            NumSubPics = subpic_grid_idx[ i ][ j ]
    }
}

```

26. In one example, the height of a sub-picture S ending at the (NumSubPicGridRows – 1) row may be set equal to (NumSubPicGridRows – 1) minus the top-most row of the sub-picture S then plus one.

a) An embodiment based on an existing implementation is highlighted as below.

```

NumSubPics = 0
for( i = 0; i < NumSubPicGridRows; i++ ) {
    for( j = 0; j < NumSubPicGridCols; j++ ) {
        if ( i == 0 )
            SubPicTop[ subpic_grid_idx[ i ][ j ] ] = 0
        else if( subpic_grid_idx[ i ][ j ] != subpic_grid_idx[ i - 1 ][ j ] ) {
            SubPicTop[ subpic_grid_idx[ i ][ j ] ] = i
            SubPicHeight[ subpic_grid_idx[ i - 1 ][ j ] ] =
i - SubPicTop[ subpic_grid_idx[ i - 1 ][ j ] ]
        }
        if ( j == 0 )
            SubPicLeft[ subpic_grid_idx[ i ][ j ] ] = 0    (7-7)
        else if( subpic_grid_idx[ i ][ j ] != subpic_grid_idx[ i ][ j - 1 ] ) {
            SubPicLeft[ subpic_grid_idx[ i ][ j ] ] = j
            SubPicWidth[ subpic_grid_idx[ i ][ j ] ] =
j - SubPicLeft[ subpic_grid_idx[ i ][ j - 1 ] ]
        }
        if ( i == NumSubPicGridRows - 1 )
            SubPicHeight[ subpic_grid_idx[ i ][ j ] ] =
i - SubPicTop[ subpic_grid_idx[ i ][ j ] ] + 1    if (j ==
NumSubPicGridRows - 1)
            SubPicWidth[ subpic_grid_idx[ i ][ j ] ] =
j - SubPicLeft[ subpic_grid_idx[ i ][ j - 1 ] ] + 1
            if( subpic_grid_idx[ i ][ j ] > NumSubPics)
                NumSubPics = subpic_grid_idx[ i ][ j ]
    }
}

```

27. In one example, the width of a sub-picture S ending at the (NumSubPicGridColumns – 1) column may be set equal to (NumSubPicGridColumns – 1) minus the left-most column of the sub-picture S then plus 1.

a) An embodiment based on an existing implementation is highlighted as below.

```

NumSubPics = 0
for( i = 0; i < NumSubPicGridRows; i++ ) {
    for( j = 0; j < NumSubPicGridCols; j++ ) {
        if ( i == 0 )
            SubPicTop[ subpic_grid_idx[ i ][ j ] ] = 0
        else if( subpic_grid_idx[ i ][ j ] != subpic_grid_idx[ i - 1 ][ j ] ) {
            SubPicTop[ subpic_grid_idx[ i ][ j ] ] = i
            SubPicHeight[ subpic_grid_idx[ i - 1 ][ j ] ] =
i - SubPicTop[ subpic_grid_idx[ i - 1 ][ j ] ]
        }
        if ( j == 0 )
            SubPicLeft[ subpic_grid_idx[ i ][ j ] ] = 0 (7-7)
        else if( subpic_grid_idx[ i ][ j ] != subpic_grid_idx[ i ][ j - 1 ] ) {
            SubPicLeft[ subpic_grid_idx[ i ][ j ] ] = j
            SubPicWidth[ subpic_grid_idx[ i ][ j ] ] =
j - SubPicLeft[ subpic_grid_idx[ i ][ j - 1 ] ]
        }
        if ( i == NumSubPicGridRows - 1 )
            SubPicHeight[ subpic_grid_idx[ i ][ j ] ] =
i - SubPicTop[ subpic_grid_idx[ i - 1 ][ j ] ] + 1      if (j ==
NumSubPicGridColumns - 1)
            SubPicWidth[ subpic_grid_idx[ i ][ j ] ] =
j - SubPicLeft[ subpic_grid_idx[ i ][ j ] ] + 1
        if( subpic_grid_idx[ i ][ j ] > NumSubPics)
            NumSubPics = subpic_grid_idx[ i ][ j ]
    }
}

```

28. The sub-picture grid must be integer times of the CTU size.

a) An embodiment based on an existing implementation is highlighted as below.

subpic_grid_col_width_minus1 plus 1 specifies the width of each element of the subpicture identifier grid in units of **CtbSizeY**. The length of the syntax element is $\text{Ceil}(\text{Log2}(\text{pic_width_max_in_luma_samples} / \text{CtbSizeY}))$ bits.

The variable NumSubPicGridCols is derived as follows:

$$\begin{aligned} \text{NumSubPicGridCols} &= \left(\frac{\text{pic_width_max_in_luma_samples} + \text{subpic_grid_col_width_minus1} * \text{CtbSizeY} + \text{CtbSizeY} - 1}{(\text{subpic_grid_col_width_minus1} * \text{CtbSizeY} + \text{CtbSizeY})} \right) \quad (7-5) \end{aligned}$$

subpic_grid_row_height_minus1 plus 1 specifies the height of each element of the subpicture identifier grid in units of 4 samples. The length of the syntax element is $\text{Ceil}(\text{Log2}(\text{pic_height_max_in_luma_samples} / \text{CtbSizeY}))$ bits.

The variable NumSubPicGridRows is derived as follows:

$$\begin{aligned} \text{NumSubPicGridRows} &= \left(\frac{\text{pic_height_max_in_luma_samples} + \text{subpic_grid_row_height_minus1} * \text{CtbSizeY} + \text{CtbSizeY} - 1}{(\text{subpic_grid_row_height_minus1} * \text{CtbSizeY} + \text{CtbSizeY})} \right) \quad (7-6) \end{aligned}$$

29. A conformance constrain is added to guarantee that sub-pictures cannot be overlapped with each other, and all the sub-pictures must cover the whole picture.

a) An embodiment based on an existing implementation is highlighted as below

Any **subpic_grid_idx[i][j]** must be equal to **idx** if the following conditions are both satisfied:

i $\geq$ **SubPicTop[idx]** and **i** $<$ **SubPicTop[idx]** + **SubPicHeight[idx]**.
j $\geq$ **SubPicLeft[idx]** and **j** $<$ **SubPicLeft[idx]** + **SubPicWidth[idx]**.

Any **subpic_grid_idx[i][j]** must be different to **idx** if the following conditions are not both satisfied:

i $\geq$ **SubPicTop[idx]** and **i** $<$ **SubPicTop[idx]** + **SubPicHeight[idx]**.
j $\geq$ **SubPicLeft[idx]** and **j** $<$ **SubPicLeft[idx]** + **SubPicWidth[idx]**.

RPR related

30. A syntax element (such as a flag), denoted as RPR_flag, is signaled to indicate whether RPR may be used or not in a video unit (e.g., sequence). RPR_flag may be signaled in SPS, VPS, or DPS.
- a) In one example, if RPR is signaled not to be used (e.g. RPR_flag is 0) all width/height signaled in PPS must be the same to the maximum width/maximum height signaled in SPS.
 - b) In one example, if RPR is signaled not to be used (e.g. RPR_flag is 0) all width/height in PPS is not signaled and inferred to be the maximum width/maximum height signaled in SPS.
 - c) In one example, if RPR is signaled not to be used (e.g. RPR_flag is 0), conformance window information is not used in the decoding process. Otherwise (RPR is signaled to be used), conformance window information may be used in the decoding process.
31. It is proposed that the interpolation filters used in the motion compensation process to derive the prediction block of a current block may be selected depending on whether the resolution of the reference picture is different to the current picture, or whether the width and/or height of the reference picture is larger than that of the current picture.
- a. In one example, the interpolation filters with less taps may be applied when condition A is satisfied, wherein condition A depends on the dimensions of the current picture and/or the reference picture.
 - i. In one example, condition A is the resolution of the reference picture is different to the current picture.
 - ii. In one example, condition A is the width and/or height of the reference picture is larger than that of the current picture.
 - iii. In one example, condition A is $W1 > a * W2$ and/or $H1 > b * H2$, wherein (W1, H1) represents the width and height of the reference picture and (W2, H2) represents the width and height of the current picture, a and b are two factors, e.g. $a = b = 1.5$.
 - iv. In one example, condition A may also depend on whether bi-prediction is used.

- 1) Condition A is satisfied only when bi-prediction is used for the current block.
- v. In one example, condition A may depend on M and N , where M and N represent the width and height of the current block.
 - 1) For example, condition A is satisfied only when $M*N \leq T$, where T is an integer such as 64.
 - 2) For example, condition A is satisfied only when $M \leq T1$ or $N \leq T2$, where $T1$ and $T2$ are integers, e.g. $T1=T2 = 4$.
 - 3) For example, condition A is satisfied only when $M \leq T1$ and $N \leq T2$, where $T1$ and $T2$ are integers, e.g. $T1=T2 = 4$.
 - 4) For example, condition A is satisfied only when $M*N \leq T$, or $M \leq T1$ or $N \leq T2$, where T , $T1$ and $T2$ are integers, e.g. $T=64$, $T1=T2 = 4$.
 - 5) In one example, the smaller condition in above sub-bullets may be replaced by greater.
- vi. In one example, 1-tap filters are applied. In other words, an integer pixel without filtering is output as the interpolation result.
- vii. In one example, bi-linear filters are applied when the resolution of the reference picture is different to the current picture.
- viii. In one example, 4-tap filters or 6-tap filters are applied when the resolution of the reference picture is different to the current picture, or the width and/or height of the reference picture is larger than that of the current picture.
 - 1) The 6-tap filters may also be used for the affine motion compensation.
 - 2) The 4-tap filters may also be used for interpolation for chroma samples.
- b. Whether to and/or how to apply the methods disclosed in bullet 31 may depend on the color components.
 - i. For example, the methods are only applied on the luma component.
- c. Whether to and/or how to apply the methods disclosed in bullet 31 may depend on the interpolation filtering direction.

- i. For example, the methods are only applied on horizontal filtering.
- ii. For example, the methods are only applied on vertical filtering.

CIIP related

32. Intra prediction signal used in CIIP process may be done in TU level instead of CU level (e.g., using reference samples outside the TU instead of CU).

- a) In one example, if either CU width or height is greater than the max transform block size, the CU may be split to multiple TUs and intra/inter prediction may be generated for each TU, e.g., using reference samples outside the TU.
- b) In one example, if the maximum transform size K is smaller than 64 (such as K = 32), then the intra-prediction used in CIIP is performed in a recursive way like in a normal intra-code block.
- c) For example, a $KM \times KN$ CIIP coded block where M and N are integers are split into MN of $K \times K$ blocks, Intra-prediction is done for each $K \times K$ block. The intra-prediction for a later coded/decoded $K \times K$ block may depend on the reconstruction samples of a previously coded/decoded $K \times K$ block.

5. Additional example embodiments

5.1 Embodiment #1: An example of syntax design in SPS/PPS/Slice header/Tile group header

[00186] The changes compared to the VTM3.0.1rc1 reference software is highlighted in large size bold face font as follow:

if(sps_temporal_mvp_enabled_flag)	
slice_temporal_mvp_enabled_flag	u(1)
if(slice_type == B)	
mvd_l1_zero_flag	u(1)
if(slice_temporal_mvp_enabled_flag) {	
if(slice_type == B)	
collocated_from_l0_flag	u(1)
}	

...	
if(!sps_affine_enabled_flag){	
if(!(sps_sbtmvp_enabled_flag && slice_temporal_mvp_enabled_flag))	
MaxSubBlockMergeListSize = 0	
else	
MaxSubBlockMergeListSize = 1	
}	
else{	
five_minus_max_num_affine_merge_cand	ue(v)
MaxSubBlockMergeListSize = 5- five_minus_max_num_affine_merge_cand	
}	

5.2 Embodiment #2: An example of syntax design in SPS/PPS/Slice header/Tile group header

7.3.2.1 Sequence parameter set RBSP syntax

seq_parameter_set_rbsp() {	Descript or
sps_seq_parameter_set_id	ue(v)
chroma_format_idc	ue(v)
...	
sps_ref_wraparound_enabled_flag	u(1)
if(sps_ref_wraparound_enabled_flag)	
sps_ref_wraparound_offset	ue(v)
sps_temporal_mvp_enabled_flag	u(1)
if(sps_temporal_mvp_enabled_flag)	

sps_sbtmvp_enabled_flag	u(1)
sps_amvr_enabled_flag	u(1)
...	
}	
rbsp_trailing_bits()	
}	

[00187] **sps_sbtmvp_enabled_flag** equal to 1 specifies that subblock-based temporal motion vector predictors may be used in decoding of pictures with all slices having **slice_type** not equal to I in the CVS. **sps_sbtmvp_enabled_flag** equal to 0 specifies that subblock-based temporal motion vector predictors are not used in the CVS. When **sps_sbtmvp_enabled_flag** is not present, it is inferred to be equal to 0.

[00188] **five_minus_max_num_subblock_merge_cand** specifies the maximum number of subblock-based merging motion vector prediction (MVP) candidates supported in the slice subtracted from 5. When **five_minus_max_num_subblock_merge_cand** is not present, it is inferred to be equal to $5 - \text{sps_sbtmvp_enabled_flag}$. The maximum number of subblock-based merging MVP candidates, **MaxNumSubblockMergeCand** is derived as follows:

[00189] $\text{MaxNumSubblockMergeCand} = 5 - \text{five_minus_max_num_subblock_merge_cand}$ (7-45)

[00190] The value of **MaxNumSubblockMergeCand** shall be in the range of 0 to 5, inclusive.

[00191] 8.3.4.2 Derivation process for motion vectors and reference indices in subblock merge mode

[00192] Inputs to this process are:

[00193] ..[No changes to the current VVC specification draft].

[00194] Outputs of this process are:

[00195] ... [No changes to the current VVC specification draft].

[00196] The variables **numSbX**, **numSbY** and the subblock merging candidate list, **subblockMergeCandList** are derived by the following ordered steps:

[00197] When **sps_sbtmvp_enabled_flag** is equal to 1 and (current picture is an IRAP and index 0 of reference picture list 0 is the current picture) is not true, the following applies:

[00198] The derivation process for merging candidates from neighbouring coding units as specified in clause 8.3.2.3 is invoked with the luma coding block location (x_{Cb} , y_{Cb}), the luma coding block width $cbWidth$, the luma coding block height $cbHeight$ and the luma coding block width as inputs, and the output being the availability flags $availableFlagA0$, $availableFlagA1$, $availableFlagB0$, $availableFlagB1$ and $availableFlagB2$, the reference indices $refIdxLXA0$, $refIdxLXA1$, $refIdxLXB0$, $refIdxLXB1$ and $refIdxLXB2$, the prediction list utilization flags $predFlagLXA0$, $predFlagLXA1$, $predFlagLXB0$, $predFlagLXB1$ and $predFlagLXB2$, and the motion vectors $mvLXA0$, $mvLXA1$, $mvLXB0$, $mvLXB1$ and $mvLXB2$, with X being 0 or 1.

[00199] The derivation process for subblock-based temporal merging candidates as specified in clause 8.3.4.3 is invoked with the luma location (x_{Cb} , y_{Cb}), the luma coding block width $cbWidth$, the luma coding block height $cbHeight$, the availability flags $availableFlagA0$, $availableFlagA1$, $availableFlagB0$, $availableFlagB1$, the reference indices $refIdxLXA0$, $refIdxLXA1$, $refIdxLXB0$, $refIdxLXB1$, the prediction list utilization flags $predFlagLXA0$, $predFlagLXA1$, $predFlagLXB0$, $predFlagLXB1$ and the motion vectors $mvLXA0$, $mvLXA1$, $mvLXB0$, $mvLXB1$ as inputs and the output being the availability flag $availableFlagSbCol$, the number of luma coding subblocks in horizontal direction $numSbX$ and in vertical direction $numSbY$, the reference indices $refIdxLXSbCol$, the luma motion vectors $mvLXSbCol[xSbIdx][ySbIdx]$ and the prediction list utilization flags $predFlagLXSbCol[xSbIdx][ySbIdx]$ with $xSbIdx = 0..numSbX - 1$, $ySbIdx = 0 .. numSbY - 1$ and X being 0 or 1.

[00200] When $sps_affine_enabled_flag$ is equal to 1, the sample locations (x_{NbA0} , y_{NbA0}), (x_{NbA1} , y_{NbA1}), (x_{NbA2} , y_{NbA2}), (x_{NbB0} , y_{NbB0}), (x_{NbB1} , y_{NbB1}), (x_{NbB2} , y_{NbB2}), (x_{NbB3} , y_{NbB3}), and the variables $numSbX$ and $numSbY$ are derived as follows:

[00201] [No changes to the current VVC specification draft].

[00202] 5.3 Embodiment # 3 An example of MV rounding

The syntax changes are based on an existing implementation.

8.5.5.3 Derivation process for subblock-based temporal merging candidates

...

- The location (x_{ColSb} , y_{ColSb}) of the collocated subblock inside $ColPic$ is derived as follows.

1. The following applies:

$yColSb = Clip3(yCtb, Min(CurPicHeightInSamplesY - 1, yCtb + (1 \ll CtbLog2SizeY) - 1), ySb + ((tempMv[1] + 8 - (tempMv[1] \geq 0)) \gg 4))$

- If `subpic_treated_as_pic_flag[ SubPicIdx ]` is equal to 1, the following applies:
 $xColSb = Clip3(xCtb, Min(SubPicRightBoundaryPos, xCtb + (1 \ll CtbLog2SizeY) + 3), xSb + ((tempMv[0] + 8 + (tempMV[0] \geq 0)) \gg 4))$
- Otherwise (`subpic_treated_as_pic_flag[ SubPicIdx ]` is equal to 0), the following applies:
 $xColSb = Clip3(xCtb, Min(CurPicWidthInSamplesY - 1, xCtb + (1 \ll CtbLog2SizeY) + 3), xSb + (tempMv[0] + 8 + (tempMV[0] \geq 0)) \gg 4))$

...

8.5.5.4 Derivation process for subblock-based temporal merging base motion data

...

The location (`xColCb`, `yColCb`) of the collocated block inside `ColPic` is derived as follows.

- The following applies:
 $yColCb = Clip3(yCtb, Min(CurPicHeightInSamplesY - 1, yCtb + (1 \ll CtbLog2SizeY) - 1), yColCtrCb + ((tempMv[1] + 8 - (tempMv[1] \geq 0)) \gg 4))$
- If `subpic_treated_as_pic_flag[ SubPicIdx ]` is equal to 1, the following applies:
 $xColCb = Clip3(xCtb, Min(SubPicRightBoundaryPos, xCtb + (1 \ll CtbLog2SizeY) + 3), xColCtrCb + ((tempMv[0] + 8 + (tempMV[0] \geq 0)) \gg 4))$
- Otherwise (`subpic_treated_as_pic_flag[ SubPicIdx ]` is equal to 0), the following applies:
 $xColCb = Clip3(xCtb, Min(CurPicWidthInSamplesY - 1, xCtb + (1 \ll CtbLog2SizeY) + 3), xColCtrCb + ((tempMv[0] + 8 + (tempMV[0] \geq 0)) \gg 4))$

5.3 Embodiment #3: An example of MV rounding.

The syntax changes are based on an existing implementation.

8.5.5.3 Derivation process for subblock-based temporal merging candidates

...

- The location (xColSb, yColSb) of the collocated subblock inside ColPic is derived as follows.

1. The following applies:

— $yColSb = Clip3(yCtb, Min(CurPicHeightInSamplesY - 1, yCtb + (1 < CtbLog2SizeY) - 1), ySb + ((tempMv[1] + 8 - (tempMv[1] \geq 0)) \gg 4))$

- If subpic_treated_as_pic_flag[SubPicIdx] is equal to 1, the following applies:

— $xColSb = Clip3(xCtb, Min(SubPicRightBoundaryPos, xCtb + (1 \ll CtbLog2SizeY) + 3), xSb + ((tempMv[0] + 8 + (tempMV[0] \geq 0)) \gg 4))$

- Otherwise (subpic_treated_as_pic_flag[SubPicIdx] is equal to 0), the following applies:

— $xColSb = Clip3(xCtb, Min(CurPicWidthInSamplesY - 1, xCtb + (1 \ll CtbLog2SizeY) + 3), xSb + (tempMv[0] + 8 + (tempMV[0] \geq 0)) \gg 4))$

...

8.5.5.4 Derivation process for subblock-based temporal merging base motion data

...

The location (xColCb, yColCb) of the collocated block inside ColPic is derived as follows.

- The following applies:

— $yColCb = Clip3(yCtb, Min(CurPicHeightInSamplesY - 1, yCtb + (1 < CtbLog2SizeY) - 1), yColCtrCb + ((tempMv[1] + 8 - (tempMv[1] \geq 0)) \gg 4))$

- If subpic_treated_as_pic_flag[SubPicIdx] is equal to 1, the following applies:

— $xColCb = Clip3(xCtb, Min(SubPicRightBoundaryPos, xCtb + (1 \ll CtbLog2SizeY) + 3), xColCtrCb + ((tempMv[0] + 8 + (tempMV[0] \geq 0)) \gg 4))$

- Otherwise (subpic_treated_as_pic_flag[SubPicIdx] is equal to 0, the following applies:

—
$$\text{xColCb} = \text{Clip3}(\text{xCtb}, \text{Min}(\text{CurPicWidthInSamplesY} - 1, \text{xCtb} + (1 \ll \text{CtbLog2SizeY}) + 3), \text{xColCtrCb} + ((\text{tempMv}[0] + 8 + (\text{tempMV}[0] \gg 0)) \gg 4))$$

5.4 Embodiment #4: A second example of MV rounding.

8.5.5.3 Derivation process for subblock-based temporal merging candidates

Inputs to this process are:

- a luma location (xCb, yCb) of the top-left sample of the current luma 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 availability flag availableFlagA₁ of the neighbouring coding unit,
- the reference index refIdxLXA₁ of the neighbouring coding unit with X being 0 or 1,
- the prediction list utilization flag predFlagLXA₁ of the neighbouring coding unit with X being 0 or 1,
- the motion vector in 1/16 fractional-sample accuracy mvLXA₁ of the neighbouring coding unit with X being 0 or 1.

Outputs of this process are:

- the availability flag availableFlagSbCol,
- the number of luma coding subblocks in horizontal direction numSbX and in vertical direction numSbY,
- the reference indices refIdxL0SbCol and refIdxL1SbCol,
- the luma motion vectors in 1/16 fractional-sample accuracy mvL0SbCol[xSbIdx][ySbIdx] and mvL1SbCol[xSbIdx][ySbIdx] with xSbIdx = 0..numSbX - 1, ySbIdx = 0 .. numSbY - 1,
- the prediction list utilization flags predFlagL0SbCol[xSbIdx][ySbIdx] and predFlagL1SbCol[xSbIdx][ySbIdx] with xSbIdx = 0..numSbX - 1, ySbIdx = 0 .. numSbY - 1.

The availability flag availableFlagSbCol is derived as follows.

- If one or more of the following conditions is true, availableFlagSbCol is set equal to 0.
- slice_temporal_mvp_enabled_flag is equal to 0.
- sps_sbtmvp_enabled_flag is equal to 0.
- cbWidth is less than 8.
- cbHeight is less than 8.
- Otherwise, the following ordered steps apply:

1. The location (x_{Ctb} , y_{Ctb}) of the top-left sample of the luma coding tree block that contains the current coding block and the location (x_{Ctr} , y_{Ctr}) of the below-right center sample of the current luma coding block are derived as follows:

$$\text{---}x_{Ctb} = (x_{Cb} \gg CtuLog2Size) \ll CtuLog2Size \quad (8-542)$$

$$\text{---}y_{Ctb} = (y_{Cb} \gg CtuLog2Size) \ll CtuLog2Size \quad (8-543)$$

$$\text{---}x_{Ctr} = x_{Cb} + (cbWidth / 2) \quad (8-544)$$

$$\text{---}y_{Ctr} = y_{Cb} + (cbHeight / 2) \quad (8-545)$$
2. The luma location ($x_{ColCtrCb}$, $y_{ColCtrCb}$) is set equal to the top-left sample of the collocated luma coding block covering the location given by (x_{Ctr} , y_{Ctr}) inside ColPic relative to the top-left luma sample of the collocated picture specified by ColPic.
3. The derivation process for subblock-based temporal merging base motion data as specified in clause 8.5.5.4 is invoked with the location (x_{Ctb} , y_{Ctb}), the location ($x_{ColCtrCb}$, $y_{ColCtrCb}$), the availability flag availableFlagA₁, and the prediction list utilization flag predFlagLXA₁, and the reference index refIdxLXA₁, and the motion vector mvLXA₁, with X being 0 and 1 as inputs and the motion vectors ctrMvLX, and the prediction list utilization flags ctrPredFlagLX of the collocated block, with X being 0 and 1, and the temporal motion vector tempMv as outputs.
4. The variable availableFlagSbCol is derived as follows:
 - If both ctrPredFlagL0 and ctrPredFlagL1 are equal to 0, availableFlagSbCol is set equal to 0.
 - Otherwise, availableFlagSbCol is set equal to 1.

When availableFlagSbCol is equal to 1, the following applies:

- The variables numSbX, numSbY, sbWidth, sbHeight and refIdxLXSbCol are derived as follows:

$$\text{---} \quad \text{numSbX} = cbWidth \gg 3 \quad (8-546)$$

$$\text{---} \quad \text{numSbY} = cbHeight \gg 3 \quad (8-547)$$

$$\text{---} \quad \text{sbWidth} = cbWidth / \text{numSbX} \quad (8-548)$$

$$\text{---} \quad \text{sbHeight} = cbHeight / \text{numSbY} \quad (8-549)$$

$$\text{---} \quad \text{refIdxLXSbCol} = 0 \quad (8-550)$$

- For $xSbIdx = 0..numSbX - 1$ and $ySbIdx = 0..numSbY - 1$, the motion vectors $mvLXSbCol[xSbIdx][ySbIdx]$ and prediction list utilization flags $predFlagLXSbCol[xSbIdx][ySbIdx]$ are derived as follows:
 - The luma location (xSb, ySb) specifying the top-left sample of the current coding subblock relative to the top-left luma sample of the current picture is derived as follows:
 - $$xSb = xCb + xSbIdx * sbWidth + sbWidth / 2 \quad (8-551)$$
 - $$ySb = yCb + ySbIdx * sbHeight + sbHeight / 2 \quad (8-552)$$
 - The location ($xColSb, yColSb$) of the collocated subblock inside ColPic is derived as follows.
 1. The following applies:
 - $$yColSb = Clip3(yCtb, Min(CurPicHeightInSamplesY - 1, yCtb + (1 \ll CtbLog2SizeY) - 1), ySb + ((tempMv[1] + 8 - (tempMv[1] \geq 0 ? 1 : 0)) \gg 4)) \quad (8-553)$$
 - If `subpic_treated_as_pic_flag[ SubPicIdx ]` is equal to 1, the following applies:
 - $$xColSb = Clip3(xCtb, Min(SubPicRightBoundaryPos, xCtb + (1 \ll CtbLog2SizeY) + 3), xSb + ((tempMv[0] + 8 - (tempMv[0] \geq 0 ? 1 : 0)) \gg 4)) \quad (8-554)$$
 - Otherwise (`subpic_treated_as_pic_flag[ SubPicIdx ]` is equal to 0), the following applies:
 - $$xColSb = Clip3(xCtb, Min(CurPicWidthInSamplesY - 1, xCtb + (1 \ll CtbLog2SizeY) + 3), xSb + ((tempMv[0] + 8 - (tempMv[0] \geq 0 ? 1 : 0)) \gg 4)) \quad (8-555)$$
- The variable `currCb` specifies the luma coding block covering the current coding subblock inside the current picture.
- The variable `colCb` specifies the luma coding block covering the modified location given by $((xColSb \gg 3) \ll 3, (yColSb \gg 3) \ll 3)$ inside the ColPic.
- The luma location ($xColCb, yColCb$) is set equal to the top-left sample of the collocated luma coding block specified by `colCb` relative to the top-left luma sample of the collocated picture specified by ColPic.
- The derivation process for collocated motion vectors as specified in clause 8.5.2.12 is invoked with `currCb`, `colCb`, ($xColCb, yColCb$), `refIdxL0` set equal to 0 and `sbFlag` set equal to 1 as

inputs and the output being assigned to the motion vector of the subblock $mvL0SbCol[xSbIdx][ySbIdx]$ and $availableFlagL0SbCol$.

- The derivation process for collocated motion vectors as specified in clause 8.5.2.12 is invoked with $currCb$, $colCb$, $(xColCb, yColCb)$, $refIdxL1$ set equal to 0 and $sbFlag$ set equal to 1 as inputs and the output being assigned to the motion vector of the subblock $mvL1SbCol[xSbIdx][ySbIdx]$ and $availableFlagL1SbCol$.
- When $availableFlagL0SbCol$ and $availableFlagL1SbCol$ are both equal to 0, the following applies for X being 0 and 1:

$$\begin{aligned} &— mvLXSbCol[xSbIdx][ySbIdx] = ctrMvLX \\ &(8-556) \end{aligned}$$

$$— predFlagLXSbCol[xSbIdx][ySbIdx] = ctrPredFlagLX \quad (8-557)$$

8.5.5.4 Derivation process for subblock-based temporal merging base motion data

Inputs to this process are:

- the location (xCb, yCb) of the top-left sample of the luma coding tree block that contains the current coding block,
- the location $(xColCtrCb, yColCtrCb)$ of the top-left sample of the collocated luma coding block that covers the below-right center sample.
- the availability flag $availableFlagA_1$ of the neighbouring coding unit,
- the reference index $refIdxLXA_1$ of the neighbouring coding unit,
- the prediction list utilization flag $predFlagLXA_1$ of the neighbouring coding unit,
- the motion vector in 1/16 fractional-sample accuracy $mvLXA_1$ of the neighbouring coding unit.

Outputs of this process are:

- the motion vectors $ctrMvL0$ and $ctrMvL1$,
- the prediction list utilization flags $ctrPredFlagL0$ and $ctrPredFlagL1$,
- the temporal motion vector $tempMv$.

The variable $tempMv$ is set as follows:

$$— tempMv[0] = 0 \quad (8-558)$$

$$— tempMv[1] = 0 \quad (8-559)$$

The variable $currPic$ specifies the current picture.

When $availableFlagA_1$ is equal to TRUE, the following applies:

- If all of the following conditions are true, $tempMv$ is set equal to $mvL0A_1$:
 - $predFlagL0A_1$ is equal to 1,
 - $DiffPicOrderCnt(ColPic, RefPicList[0][refIdxL0A_1])$ is equal to 0,

- Otherwise, if all of the following conditions are true, tempMv is set equal to mvL1A₁:
 - slice_type is equal to B,
 - predFlagL1A₁ is equal to 1,
 - DiffPicOrderCnt(ColPic, RefPicList[1][refIdxL1A₁]) is equal to 0.

The location (xColCb, yColCb) of the collocated block inside ColPic is derived as follows.

- The following applies:

$$\begin{aligned} \text{yColCb} &= \text{Clip3}(\text{yCtb}, \\ &\quad \text{Min}(\text{CurPicHeightInSamplesY} - 1, \text{yCtb} + (1 \ll \text{CtbLog2SizeY}) - 1), \\ &\quad \text{yColCtCb} + ((\text{tempMv}[1] + 8 - (\text{tempMv}[1] \geq 0 ? 1 : 0)) \gg 4)) \quad (8-560) \end{aligned}$$
- If subpic_treated_as_pic_flag[SubPicIdx] is equal to 1, the following applies:

$$\begin{aligned} \text{xColCb} &= \text{Clip3}(\text{xCtb}, \\ &\quad \text{Min}(\text{SubPicRightBoundaryPos}, \text{xCtb} + (1 \ll \text{CtbLog2SizeY}) + 3), \\ &\quad \text{xColCtCb} + ((\text{tempMv}[0] + 8 - (\text{tempMv}[0] \geq 0 ? 1 : 0)) \gg 4)) \quad (8-561) \end{aligned}$$
- Otherwise (subpic_treated_as_pic_flag[SubPicIdx] is equal to 0, the following applies:

$$\begin{aligned} \text{xColCb} &= \text{Clip3}(\text{xCtb}, \\ &\quad \text{Min}(\text{CurPicWidthInSamplesY} - 1, \text{xCtb} + (1 \ll \text{CtbLog2SizeY}) + 3), \\ &\quad \text{xColCtCb} + ((\text{tempMv}[0] + 8 - (\text{tempMv}[0] \geq 0 ? 1 : 0)) \gg 4)) \quad (8-562) \end{aligned}$$

The array colPredMode is set equal to the prediction mode array CuPredMode[0] of the collocated picture specified by ColPic.

The motion vectors ctrMvL0 and ctrMvL1, and the prediction list utilization flags ctrPredFlagL0 and ctrPredFlagL1 are derived as follows:

- If colPredMode[xColCb][yColCb] is equal to MODE_INTER, the following applies:
 - The variable currCb specifies the luma coding block covering (xCtCb, yCtCb) inside the current picture.
 - The variable colCb specifies the luma coding block covering the modified location given by ((xColCb >> 3) << 3, (yColCb >> 3) << 3) inside the ColPic.
 - The luma location (xColCb, yColCb) is set equal to the top-left sample of the collocated luma coding block specified by colCb relative to the top-left luma sample of the collocated picture specified by ColPic.

- The derivation process for collocated motion vectors specified in clause 8.5.2.12 is invoked with currCb, colCb, (xCb, yCb), refIdxL0 set equal to 0, and sbFlag set equal to 1 as inputs and the output being assigned to ctrMvL0 and ctrPredFlagL0.
- The derivation process for collocated motion vectors specified in clause 8.5.2.12 is invoked with currCb, colCb, (xCb, yCb), refIdxL1 set equal to 0, and sbFlag set equal to 1 as inputs and the output being assigned to ctrMvL1 and ctrPredFlagL1.
- Otherwise, the following applies:
 - ctrPredFlagL0 = 0 (8-563)
 - ctrPredFlagL1 = 0 (8-564)

5.5 Embodiment #5: A third example of MV rounding.

8.5.5.3 Derivation process for subblock-based temporal merging candidates

Inputs to this process are:

- a luma location (xCb, yCb) of the top-left sample of the current luma 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 availability flag availableFlagA₁ of the neighbouring coding unit,
- the reference index refIdxLXA₁ of the neighbouring coding unit with X being 0 or 1,
- the prediction list utilization flag predFlagLXA₁ of the neighbouring coding unit with X being 0 or 1,
- the motion vector in 1/16 fractional-sample accuracy mvLXA₁ of the neighbouring coding unit with X being 0 or 1.

Outputs of this process are:

- the availability flag availableFlagSbCol,
- the number of luma coding subblocks in horizontal direction numSbX and in vertical direction numSbY,
- the reference indices refIdxL0SbCol and refIdxL1SbCol,
- the luma motion vectors in 1/16 fractional-sample accuracy mvL0SbCol[xSbIdx][ySbIdx] and mvL1SbCol[xSbIdx][ySbIdx] with xSbIdx = 0..numSbX – 1, ySbIdx = 0 .. numSbY – 1,
- the prediction list utilization flags predFlagL0SbCol[xSbIdx][ySbIdx] and predFlagL1SbCol[xSbIdx][ySbIdx] with xSbIdx = 0..numSbX – 1, ySbIdx = 0 .. numSbY – 1.

The availability flag availableFlagSbCol is derived as follows.

- If one or more of the following conditions is true, availableFlagSbCol is set equal to 0.

- slice_temporal_mvp_enabled_flag is equal to 0.
- sps_sbtmvp_enabled_flag is equal to 0.
- cbWidth is less than 8.
- cbHeight is less than 8.
- Otherwise, the following ordered steps apply:
 5. The location (xCtb, yCtb) of the top-left sample of the luma coding tree block that contains the current coding block and the location (xCtr, yCtr) of the below-right center sample of the current luma coding block are derived as follows:
 - $x_{Ctb} = (x_{Cb} \gg CtuLog2Size) \ll CtuLog2Size$ (8-542)
 - $y_{Ctb} = (y_{Cb} \gg CtuLog2Size) \ll CtuLog2Size$ (8-543)
 - $x_{Ctr} = x_{Cb} + (cbWidth / 2)$ (8-544)
 - $y_{Ctr} = y_{Cb} + (cbHeight / 2)$ (8-545)
 6. The luma location (xColCtrCb, yColCtrCb) is set equal to the top-left sample of the collocated luma coding block covering the location given by (xCtr, yCtr) inside ColPic relative to the top-left luma sample of the collocated picture specified by ColPic.
 7. The derivation process for subblock-based temporal merging base motion data as specified in clause 8.5.5.4 is invoked with the location (xCtb, yCtb), the location (xColCtrCb, yColCtrCb), the availability flag availableFlagA₁, and the prediction list utilization flag predFlagLXA₁, and the reference index refIdxLXA₁, and the motion vector mvLXA₁, with X being 0 and 1 as inputs and the motion vectors ctrMvLX, and the prediction list utilization flags ctrPredFlagLX of the collocated block, with X being 0 and 1, and the temporal motion vector tempMv as outputs.
 8. The variable availableFlagSbCol is derived as follows:
 - If both ctrPredFlagL0 and ctrPredFlagL1 are equal to 0, availableFlagSbCol is set equal to 0.
 - Otherwise, availableFlagSbCol is set equal to 1.

When availableFlagSbCol is equal to 1, the following applies:

- The variables numSbX, numSbY, sbWidth, sbHeight and refIdxLXSbCol are derived as follows:
 - $numSbX = cbWidth \gg 3$ (8-546)
 - $numSbY = cbHeight \gg 3$ (8-547)
 - $sbWidth = cbWidth / numSbX$ (8-548)

- $$\text{sbHeight} = \text{cbHeight} / \text{numSbY}$$
 (8-549)
- $$\text{refIdxLXSbCol} = 0$$
 (8-550)
- For $\text{xSbIdx} = 0.. \text{numSbX} - 1$ and $\text{ySbIdx} = 0.. \text{numSbY} - 1$, the motion vectors $\text{mvLXSbCol}[\text{xSbIdx}][\text{ySbIdx}]$ and prediction list utilization flags $\text{predFlagLXSbCol}[\text{xSbIdx}][\text{ySbIdx}]$ are derived as follows:
- The luma location (xSb, ySb) specifying the top-left sample of the current coding subblock relative to the top-left luma sample of the current picture is derived as follows:
- $$\text{xSb} = \text{xCb} + \text{xSbIdx} * \text{sbWidth} + \text{sbWidth} / 2$$
 (8-551)
- $$\text{ySb} = \text{yCb} + \text{ySbIdx} * \text{sbHeight} + \text{sbHeight} / 2$$
 (8-552)
- The location $(\text{xColSb}, \text{yColSb})$ of the collocated subblock inside ColPic is derived as follows.
1. The following applies:
- $$\begin{aligned} \text{yColSb} &= \text{Clip3}(\text{yCtb}, \\ &\text{Min}(\text{CurPicHeightInSamplesY} - 1, \text{yCtb} + (1 \ll \text{CtbLog2SizeY}) - 1), \\ &\text{ySb} + ((\text{tempMv}[1] + (\text{tempMv}[1] \geq 0 ? 7 : 8)) \gg 4)) \end{aligned}$$
 (8-553)
- If $\text{subpic_treated_as_pic_flag}[\text{SubPicIdx}]$ is equal to 1, the following applies:
- $$\begin{aligned} \text{xColSb} &= \text{Clip3}(\text{xCtb}, \\ &\text{Min}(\text{SubPicRightBoundaryPos}, \text{xCtb} + (1 \ll \text{CtbLog2SizeY}) + 3), \\ &\text{xSb} + ((\text{tempMv}[0] + (\text{tempMv}[0] \geq 0 ? 7 : 8)) \gg 4)) \end{aligned}$$
 (8-554)
- Otherwise ($\text{subpic_treated_as_pic_flag}[\text{SubPicIdx}]$ is equal to 0), the following applies:
- $$\begin{aligned} \text{xColSb} &= \text{Clip3}(\text{xCtb}, \\ &\text{Min}(\text{CurPicWidthInSamplesY} - 1, \text{xCtb} + (1 \ll \text{CtbLog2SizeY}) + 3), \\ &\text{xSb} + ((\text{tempMv}[0] + (\text{tempMv}[0] \geq 0 ? 7 : 8)) \gg 4)) \end{aligned}$$
 (8-555)
- The variable currCb specifies the luma coding block covering the current coding subblock inside the current picture.
- The variable colCb specifies the luma coding block covering the modified location given by $((\text{xColSb} \gg 3) \ll 3, (\text{yColSb} \gg 3) \ll 3)$ inside the ColPic.

- The luma location (x_{ColCb} , y_{ColCb}) is set equal to the top-left sample of the collocated luma coding block specified by $colCb$ relative to the top-left luma sample of the collocated picture specified by $ColPic$.
- The derivation process for collocated motion vectors as specified in clause 8.5.2.12 is invoked with $currCb$, $colCb$, (x_{ColCb} , y_{ColCb}), $refIdxL0$ set equal to 0 and $sbFlag$ set equal to 1 as inputs and the output being assigned to the motion vector of the subblock $mvL0SbCol[xSbIdx][ySbIdx]$ and $availableFlagL0SbCol$.
- The derivation process for collocated motion vectors as specified in clause 8.5.2.12 is invoked with $currCb$, $colCb$, (x_{ColCb} , y_{ColCb}), $refIdxL1$ set equal to 0 and $sbFlag$ set equal to 1 as inputs and the output being assigned to the motion vector of the subblock $mvL1SbCol[xSbIdx][ySbIdx]$ and $availableFlagL1SbCol$.
- When $availableFlagL0SbCol$ and $availableFlagL1SbCol$ are both equal to 0, the following applies for X being 0 and 1:
 - $mvLXSbCol[xSbIdx][ySbIdx] = ctrMvLX$
(8-556)
 - $predFlagLXSbCol[xSbIdx][ySbIdx] = ctrPredFlagLX$ (8-557)

8.5.5.4 Derivation process for subblock-based temporal merging base motion data

Inputs to this process are:

- the location (x_{Ctb} , y_{Ctb}) of the top-left sample of the luma coding tree block that contains the current coding block,
- the location ($x_{ColCtrCb}$, $y_{ColCtrCb}$) of the top-left sample of the collocated luma coding block that covers the below-right center sample.
- the availability flag $availableFlagA_1$ of the neighbouring coding unit,
- the reference index $refIdxLXA_1$ of the neighbouring coding unit,
- the prediction list utilization flag $predFlagLXA_1$ of the neighbouring coding unit,
- the motion vector in 1/16 fractional-sample accuracy $mvLXA_1$ of the neighbouring coding unit.

Outputs of this process are:

- the motion vectors $ctrMvL0$ and $ctrMvL1$,
- the prediction list utilization flags $ctrPredFlagL0$ and $ctrPredFlagL1$,
- the temporal motion vector $tempMv$.

The variable $tempMv$ is set as follows:

$$\text{—tempMv}[0] = 0 \quad (8-558)$$

$$\text{---tempMv}[1] = 0 \quad (8-559)$$

The variable currPic specifies the current picture.

When availableFlagA₁ is equal to TRUE, the following applies:

- If all of the following conditions are true, tempMv is set equal to mvL0A₁:
 - predFlagL0A₁ is equal to 1,
 - DiffPicOrderCnt(ColPic, RefPicList[0][refIdxL0A₁]) is equal to 0,
- Otherwise, if all of the following conditions are true, tempMv is set equal to mvL1A₁:
 - slice_type is equal to B,
 - predFlagL1A₁ is equal to 1,
 - DiffPicOrderCnt(ColPic, RefPicList[1][refIdxL1A₁]) is equal to 0.

The location (xColCb, yColCb) of the collocated block inside ColPic is derived as follows.

- The following applies:
 - $$\begin{aligned} yColCb &= Clip3(yCtb, \\ &\quad Min(CurPicHeightInSamplesY - 1, yCtb + (1 \ll CtbLog2SizeY) - 1), \\ &\quad yColCtCb + ((tempMv[1] + (tempMv[1] \geq 0 ? 7 : 8)) \gg 4)) \end{aligned} \quad (8-560)$$
- If subpic_treated_as_pic_flag[SubPicIdx] is equal to 1, the following applies:
 - $$\begin{aligned} xColCb &= Clip3(xCtb, \\ &\quad Min(SubPicRightBoundaryPos, xCtb + (1 \ll CtbLog2SizeY) + 3), \\ &\quad xColCtCb + ((tempMv[0] + (tempMv[0] \geq 0 ? 7 : 8)) \gg 4)) \end{aligned} \quad (8-561)$$
- Otherwise (subpic_treated_as_pic_flag[SubPicIdx] is equal to 0, the following applies:
 - $$\begin{aligned} xColCb &= Clip3(xCtb, \\ &\quad Min(CurPicWidthInSamplesY - 1, xCtb + (1 \ll CtbLog2SizeY) + 3), \\ &\quad xColCtCb + ((tempMv[0] + (tempMv[0] \geq 0 ? 7 : 8)) \gg 4)) \end{aligned} \quad (8-562)$$

The array colPredMode is set equal to the prediction mode array CuPredMode[0] of the collocated picture specified by ColPic.

The motion vectors ctrMvL0 and ctrMvL1, and the prediction list utilization flags ctrPredFlagL0 and ctrPredFlagL1 are derived as follows:

- If colPredMode[xColCb][yColCb] is equal to MODE_INTER, the following applies:
 - The variable currCb specifies the luma coding block covering (xCtCb, yCtCb) inside the current picture.

- The variable colCb specifies the luma coding block covering the modified location given by $((xColCb \gg 3) \ll 3, (yColCb \gg 3) \ll 3)$ inside the ColPic.
- The luma location $(xColCb, yColCb)$ is set equal to the top-left sample of the collocated luma coding block specified by colCb relative to the top-left luma sample of the collocated picture specified by ColPic.
- The derivation process for collocated motion vectors specified in clause 8.5.2.12 is invoked with currCb, colCb, $(xColCb, yColCb)$, refIdxL0 set equal to 0, and sbFlag set equal to 1 as inputs and the output being assigned to ctrMvL0 and ctrPredFlagL0.
- The derivation process for collocated motion vectors specified in clause 8.5.2.12 is invoked with currCb, colCb, $(xColCb, yColCb)$, refIdxL1 set equal to 0, and sbFlag set equal to 1 as inputs and the output being assigned to ctrMvL1 and ctrPredFlagL1.
- Otherwise, the following applies:
 - ctrPredFlagL0 = 0 (8-563)
 - ctrPredFlagL1 = 0 (8-564)

8.5.6.3 Fractional sample interpolation process

8.5.6.3.1 General

Inputs to this process are:

- a luma location (xSb, ySb) specifying the top-left sample of the current coding subblock relative to the top-left luma sample of the current picture,
- a variable sbWidth specifying the width of the current coding subblock,
- a variable sbHeight specifying the height of the current coding subblock,
- a motion vector offset mvOffset,
- a refined motion vector refMvLX,
- the selected reference picture sample array refPicLX,
- the half sample interpolation filter index hpellIdx,
- the bi-directional optical flow flag bdofFlag,
- a variable cIdx specifying the colour component index of the current block.

Outputs of this process are:

- an $(sbWidth + brdExtSize) \times (sbHeight + brdExtSize)$ array predSamplesLX of prediction sample values.

The prediction block border extension size brdExtSize is derived as follows:

— $\text{brdExtSize} = (\text{bdoFlag} \mid (\text{inter_affine_flag}[\text{xSb}][\text{ySb}] \&\& \text{sps_affine_prof_enabled_flag})) ? 2 : 0$ (8-752)

The variable fRefWidth is set equal to the PicOutputWidthL of the reference picture in luma samples.

The variable fRefHeight is set equal to PicOutputHeightL of the reference picture in luma samples.

The motion vector mvLX is set equal to $(\text{refMvLX} - \text{mvOffset})$.

– If cIdx is equal to 0, the following applies:

– The scaling factors and their fixed-point representations are defined as

— $\text{hori_scale_fp} = ((\text{fRefWidth} \ll 14) + (\text{PicOutputWidthL} \gg 1)) / \text{PicOutputWidthL}$ (8-753)

— $\text{vert_scale_fp} = ((\text{fRefHeight} \ll 14) + (\text{PicOutputHeightL} \gg 1)) / \text{PicOutputHeightL}$ (8-754)

– Let $(\text{xIntL}, \text{yIntL})$ be a luma location given in full-sample units and $(\text{xFracL}, \text{yFracL})$ be an offset given in 1/16-sample units. These variables are used only in this clause for specifying fractional-sample locations inside the reference sample arrays refPicLX .

– The top-left coordinate of the bounding block for reference sample padding $(\text{xSbIntL}, \text{ySbIntL})$ is set equal to $(\text{xSb} + (\text{mvLX}[0] \gg 4), \text{ySb} + (\text{mvLX}[1] \gg 4))$.

– For each luma sample location

$(\text{xL} = 0.. \text{sbWidth} - 1 + \text{brdExtSize}, \text{yL} = 0.. \text{sbHeight} - 1 + \text{brdExtSize})$ inside the prediction luma sample array predSamplesLX , the corresponding prediction luma sample value $\text{predSamplesLX}[\text{xL}][\text{yL}]$ is derived as follows:

– Let $(\text{refxSbL}, \text{refySbL})$ and $(\text{refxL}, \text{refyL})$ be luma locations pointed to by a motion vector $(\text{refMvLX}[0], \text{refMvLX}[1])$ given in 1/16-sample units. The variables refxSbL , refxL , refySbL , and refyL are derived as follows:

— $\text{refxSbL} = ((\text{xSb} \ll 4) + \text{refMvLX}[0]) * \text{hori_scale_fp}$ (8-755)

— $\text{refxL} = ((\text{Sign}(\text{refxSb}) * ((\text{Abs}(\text{refxSb}) + 128) \gg 8)$

$+ \text{xL} * ((\text{hori_scale_fp} + 8) \gg 4) + 32) \gg 6$ (8-756)

— $\text{refySbL} = ((\text{ySb} \ll 4) + \text{refMvLX}[1]) * \text{vert_scale_fp}$ (8-757)

— $\text{refyL} = ((\text{Sign}(\text{refySb}) * ((\text{Abs}(\text{refySb}) + 128) \gg 8) + \text{yL} * ((\text{vert_scale_fp} + 8) \gg 4) + 32) \gg 6$ (8-758)

—The variables x_{IntL} , y_{IntL} , x_{FracL} and y_{FracL} are derived as follows:

$$\text{---} \quad x_{IntL} = \text{refx}_L \gg 4$$

(8-759)

$$\text{---} \quad y_{IntL} = \text{refy}_L \gg 4$$

(8-760)

$$\text{---} \quad x_{FracL} = \text{refx}_L \& 15$$

(8-761)

$$\text{---} \quad y_{FracL} = \text{refy}_L \& 15$$

(8-762)

- **using6TapFlag** is set to be 1 if all the conditions below are satisfied:
 - **cbWidth[0][xSb][ySb] <=4 || cbHeight[0][xSb][ySb] <=4 || cbWidth[0][xSb][ySb] * cbHeight[0][xSb][ySb] <=64.**
 - **PredFlagL0[xSb][ySb] == 1 && PredFlagL1[xSb][ySb] == 1.**
- If **bdoFlag** is equal to TRUE or (**sps_affine_prof_enabled_flag** is equal to TRUE and **inter_affine_flag[xSb][ySb]** is equal to TRUE), and one or more of the following conditions are true, the prediction luma sample value **predSamplesLX[xL][yL]** is derived by invoking the luma integer sample fetching process as specified in clause 8.5.6.3.3 with ($x_{IntL} + (x_{FracL} \gg 3) - 1$), $y_{IntL} + (y_{FracL} \gg 3) - 1$) and **refPicLX** as inputs.
 1. x_L is equal to 0.
 2. x_L is equal to **sbWidth** + 1.
 3. y_L is equal to 0.
 4. y_L is equal to **sbHeight** + 1.
- Otherwise, the prediction luma sample value **predSamplesLX[xL][yL]** is derived by invoking the luma sample 8-tap interpolation filtering process as specified in clause 8.5.6.3.2 with ($x_{IntL} - (brdExtSize > 0 ? 1 : 0)$), $y_{IntL} - (brdExtSize > 0 ? 1 : 0)$), (x_{FracL} , y_{FracL}), (x_{SbIntL} , y_{SbIntL}), **refPicLX**, **hpellIdx**, **sbWidth**, **sbHeight** and (x_{Sb} , y_{Sb}) **and using6TapFlag** as inputs.
- Otherwise (**cIdx** is not equal to 0), the following applies:
 - Let (x_{IntC} , y_{IntC}) be a chroma location given in full-sample units and (x_{FracC} , y_{FracC}) be an offset given in 1/32 sample units. These variables are used only in this clause for specifying general fractional-sample locations inside the reference sample arrays **refPicLX**.

- The top-left coordinate of the bounding block for reference sample padding (x_{SbIntC} , y_{SbIntC}) is set equal to $((x_{Sb} / SubWidthC) + (mvLX[0] \gg 5), (y_{Sb} / SubHeightC) + (mvLX[1] \gg 5))$.
- For each chroma sample location ($x_C = 0..sbWidth - 1$, $y_C = 0..sbHeight - 1$) inside the prediction chroma sample arrays $predSamplesLX$, the corresponding prediction chroma sample value $predSamplesLX[x_C][y_C]$ is derived as follows:
 - Let $(refx_{SbC}, refy_{SbC})$ and $(refx_C, refy_C)$ be chroma locations pointed to by a motion vector $(mvLX[0], mvLX[1])$ given in 1/32-sample units. The variables $refx_{SbC}$, $refy_{SbC}$, $refx_C$ and $refy_C$ are derived as follows:
 - $refx_{SbC} = ((x_{Sb} / SubWidthC \ll 5) + mvLX[0]) * hori_scale_fp$ (8-763)
 - $refx_C = ((Sign(refx_{SbC}) * (Abs(refx_{SbC}) + 256)) \gg 9) + x_C * ((hori_scale_fp + 8) \gg 4) + 16 \gg 5$ (8-764)
 - $refy_{SbC} = ((y_{Sb} / SubHeightC \ll 5) + mvLX[1]) * vert_scale_fp$ (8-765)
 - $refy_C = ((Sign(refy_{SbC}) * (Abs(refy_{SbC}) + 256)) \gg 9) + y_C * ((vert_scale_fp + 8) \gg 4) + 16 \gg 5$ (8-766)
 - The variables x_{IntC} , y_{IntC} , x_{FracC} and y_{FracC} are derived as follows:
 - $x_{IntC} = refx_C \gg 5$ (8-767)
 - $y_{IntC} = refy_C \gg 5$ (8-768)
 - $x_{FracC} = refx_C \& 31$ (8-769)
 - $y_{FracC} = refy_C \& 31$ (8-770)
 - The prediction sample value $predSamplesLX[x_C][y_C]$ is derived by invoking the process specified in clause 8.5.6.3.4 with (x_{IntC}, y_{IntC}) , (x_{FracC}, y_{FracC}) , (x_{SbIntC}, y_{SbIntC}) , $sbWidth$, $sbHeight$ and $refPicLX$ as inputs.

8.5.6.3.2 Luma sample interpolation filtering process

Inputs to this process are:

- a luma location in full-sample units (x_{Int_L} , y_{Int_L}),
- a luma location in fractional-sample units (x_{Frac_L} , y_{Frac_L}),
- a luma location in full-sample units (x_{SbInt_L} , y_{SbInt_L}) specifying the top-left sample of the bounding block for reference sample padding relative to the top-left luma sample of the reference picture,
- the luma reference sample array $refPicLXL$,
- the half sample interpolation filter index $hpelIdx$,
- a variable $sbWidth$ specifying the width of the current subblock,
- a variable $sbHeight$ specifying the height of the current subblock,
- a luma location (x_{Sb} , y_{Sb}) specifying the top-left sample of the current subblock relative to the top-left luma sample of the current picture,
- **a flag $using6TapFlag$ specifying whether the 6-tap interpolation filter used.**

Output of this process is a predicted luma sample value $predSampleLXL$.

The variables $shift1$, $shift2$ and $shift3$ are derived as follows:

- The variable $shift1$ is set equal to $\text{Min}(4, \text{BitDepth}_Y - 8)$, the variable $shift2$ is set equal to 6 and the variable $shift3$ is set equal to $\text{Max}(2, 14 - \text{BitDepth}_Y)$.
- The variable $picW$ is set equal to $pic_width_in_luma_samples$ and the variable $picH$ is set equal to $pic_height_in_luma_samples$.

The luma interpolation filter coefficients $f_L[p]$ for each 1/16 fractional sample position p equal to x_{Frac_L} or y_{Frac_L} are derived as follows:

- **If at least one of the following conditions are satisfied, the luma interpolation filter coefficients $f_L[p]$ are specified in Table 8-12.**
 - **MotionModelIdx[x_{Sb}][y_{Sb}] is greater than 0, and $sbWidth$ and $sbHeight$ are both equal to 4,**
 - **$using6TapFlag$ is equal to 1.**
- Otherwise, the luma interpolation filter coefficients $f_L[p]$ are specified in Table 8-11 depending on $hpelIdx$.

The luma locations in full-sample units (x_{Int_i} , y_{Int_i}) are derived as follows for $i = 0..7$:

- If $subpic_treated_as_pic_flag[SubPicIdx]$ is equal to 1, the following applies:
 - $x_{Int_i} = \text{Clip3}(SubPicLeftBoundaryPos, SubPicRightBoundaryPos, x_{Int_L} + i - 3)$ (8-771)
 - $y_{Int_i} = \text{Clip3}(SubPicTopBoundaryPos, SubPicBotBoundaryPos, y_{Int_L} + i - 3)$ (8-772)
- Otherwise ($subpic_treated_as_pic_flag[SubPicIdx]$ is equal to 0), the following applies:
 - $x_{Int_i} = \text{Clip3}(0, picW - 1, sps_ref_wraparound_enabled_flag ? \text{ClipH}((sps_ref_wraparound_offset_minus1 + 1) * \text{MinCbSizeY}, picW, x_{Int_L} + i - 3) : 0)$

(8-773)

$$xInt_L + i - 3)$$

$$— yInt_i = Clip3(0, picH - 1, yInt_L + i - 3)$$

(8-774)

The luma locations in full-sample units are further modified as follows for $i = 0..7$:

$$— xInt_i = Clip3(xSbInt_L - 3, xSbInt_L + sbWidth + 4, xInt_i) \quad (8-775)$$

$$— yInt_i = Clip3(ySbInt_L - 3, ySbInt_L + sbHeight + 4, yInt_i) \quad (8-776)$$

The predicted luma sample value $predSampleLX_L$ is derived as follows:

- If both $xFrac_L$ and $yFrac_L$ are equal to 0, the value of $predSampleLX_L$ is derived as follows:
 - $predSampleLX_L = refPicLX_L[xInt_3][yInt_3] << shift3 \quad (8-777)$
- Otherwise, if $xFrac_L$ is not equal to 0 and $yFrac_L$ is equal to 0, the value of $predSampleLX_L$ is derived as follows:
 - $predSampleLX_L = (\sum_{i=0}^7 f_L[xFrac_L][i] * refPicLX_L[xInt_3][yInt_3]) >> shift1 \quad (8-778)$
- Otherwise, if $xFrac_L$ is equal to 0 and $yFrac_L$ is not equal to 0, the value of $predSampleLX_L$ is derived as follows:
 - $predSampleLX_L = (\sum_{i=0}^7 f_L[yFrac_L][i] * refPicLX_L[xInt_3][yInt_i]) >> shift1 \quad (8-779)$
- Otherwise, if $xFrac_L$ is not equal to 0 and $yFrac_L$ is not equal to 0, the value of $predSampleLX_L$ is derived as follows:
 - The sample array $temp[n]$ with $n = 0..7$, is derived as follows:
 - $temp[n] = (\sum_{i=0}^7 f_L[xFrac_L][i] * refPicLX_L[xInt_i][yInt_n]) >> shift1 \quad (8-780)$
 - The predicted luma sample value $predSampleLX_L$ is derived as follows:
 - $predSampleLX_L = (\sum_{i=0}^7 f_L[yFrac_L][i] * temp[i]) >> shift2 \quad (8-781)$

Table 8-11 – Specification of the luma interpolation filter coefficients $f_L[p]$ for each $1/16$ fractional sample position p .

Fractional sample position p	interpolation filter coefficients							
	$f_L[p][0]$	$f_L[p][1]$	$f_L[p][2]$	$f_L[p][3]$	$f_L[p][4]$	$f_L[p][5]$	$f_L[p][6]$	$f_L[p][7]$

1	0	1	-3	63	4	-2	1	0
2	-1	2	-5	62	8	-3	1	0
3	-1	3	-8	60	13	-4	1	0
4	-1	4	-10	58	17	-5	1	0
5	-1	4	-11	52	26	-8	3	-1
6	-1	3	-9	47	31	-10	4	-1
7	-1	4	-11	45	34	-10	4	-1
8 (hpelIfIdx == 0)	-1	4	-11	40	40	-11	4	-1
8 (hpelIfIdx == 1)	0	3	9	20	20	9	3	0
9	-1	4	-10	34	45	-11	4	-1
10	-1	4	-10	31	47	-9	3	-1
11	-1	3	-8	26	52	-11	4	-1
12	0	1	-5	17	58	-10	4	-1
13	0	1	-4	13	60	-8	3	-1
14	0	1	-3	8	62	-5	2	-1
15	0	1	-2	4	63	-3	1	0

Table 8-12 – Specification of the luma interpolation filter coefficients $f_L[p]$ for each 1/16 fractional sample position p for affine motion mode.

Fractional sample position p	interpolation filter coefficients							
	$f_L[p][0]$	$f_L[p][1]$	$f_L[p][2]$	$f_L[p][3]$	$f_L[p][4]$	$f_L[p][5]$	$f_L[p][6]$	$f_L[p][7]$

1	0	1	-3	63	4	-2	1	0
2	0	1	-5	62	8	-3	1	0
3	0	2	-8	60	13	-4	1	0
4	0	3	-10	58	17	-5	1	0
5	0	3	-11	52	26	-8	2	0
6	0	2	-9	47	31	-10	3	0
7	0	3	-11	45	34	-10	3	0
8	0	3	-11	40	40	-11	3	0
9	0	3	-10	34	45	-11	3	0
10	0	3	-10	31	47	-9	2	0
11	0	2	-8	26	52	-11	3	0
12	0	1	-5	17	58	-10	3	0
13	0	1	-4	13	60	-8	2	0
14	0	1	-3	8	62	-5	1	0
15	0	1	-2	4	63	-3	1	0

[00203] FIG. 30 is a flowchart for a method 3000 for video processing. The method 3000 includes, at operation 3010, determining, for a conversion between a current block of video and a bitstream representation of the video, a maximum number of candidates (ML) in a sub-block based merge candidate list and/or whether to add sub-block based temporal motion vector prediction (SbTMVP) candidates to the sub-block based merge candidate list based on whether temporal motion vector prediction (TMVP) is enabled for use during the conversion or whether a current picture referencing (CPR) coding mode is used for the conversion.

[00204] The method 3000 includes, at operation 3020, performing, based on the determining, the conversion.

[00205] FIG. 31 is a flowchart for a method 3100 for video processing. The method 3100 includes, at operation 3110, determining, for a conversion between a current block of video and a bitstream representation of the video, a maximum number of candidates (ML) in a sub-block based merge candidate list based on whether a temporal motion vector prediction (TMVP), a

sub-block based temporal motion vector prediction (SbTMVP), and an affine coding mode are enabled for use during the conversion.

[00206] The method 3100 includes, at operation 3120, performing, based on the determining, the conversion.

[00207] FIG. 32 is a flowchart for a method 3200 for video processing. The method 3200 includes, at operation 3210, determining, for a conversion between a current block of a first video segment of a video and a bitstream representation of the video, that a sub-block based motion vector prediction (SbTMVP) mode is disabled for the conversion due to a temporal motion vector prediction (TMVP) mode being disabled at a first video segment level.

[00208] The method 3200 includes, at operation 3220, performing the conversion based on the determining, the bitstream representation complying with a format that specifies whether an indication of the SbTMVP mode is included and/or a position of the indication of the SbTMVP mode in a merge candidate list, with respect to an indication of the TMVP mode.

[00209] FIG. 33 is a flowchart for a method 3300 for video processing. The method 3300 includes, at operation 3310, performing a conversion between a current block of a video that is coded using a sub-block based temporal motion vector prediction (SbTMVP) tool or a temporal motion vector prediction (TMVP) tool and a bitstream representation of the video, a coordinate of a corresponding position of the current block or a sub-block of the current block being selectively masked using a mask based on a compression of motion vectors associated with the SbTMVP tool or the TMVP tool, and an application of the mask comprising computing a bitwise AND operation between a value of the coordinate and a value of the mask.

[00210] FIG. 34 is a flowchart for a method 3400 for video processing. The method 3400 includes, at operation 3410, determining, based on one or more characteristics of a current block of a video segment of a video, a valid corresponding region of the current block for an application of a sub-block based motion vector prediction (SbTMVP) tool on the current block.

[00211] The method 3400 includes, at operation 3420, performing, based on the determining, a conversion between the current block and a bitstream representation of the video.

[00212] FIG. 35 is a flowchart for a method 3500 for video processing. The method 3500 includes, at operation 3510, determining, for a current block of a video that is coded using a sub-block based temporal motion vector prediction (SbTMVP) tool, a default motion vector.

[00213] The method 3500 includes, at operation 3520, performing, based on the determining, a conversion between the current block and a bitstream representation of the video, the default motion vector being determined in case that a motion vector is not obtained from a block covering a corresponding position in the collocated picture that is associated with a center position of the current block.

[00214] FIG. 36 is a flowchart for a method 3600 for video processing. The method 3600 includes, at operation 3610, inferring, for a current block of a video segment of a video, that a sub-block based temporal motion vector prediction (SbTMVP) tool or a temporal motion vector prediction (TMVP) tool is disabled for the video segment in case that a current picture of the current block is a reference picture with an index set to M in a reference picture list X, M and X being integers, and $X = 0$ or $X = 1$.

[00215] The method 3600 includes, at operation 3620, performing, based on the inferring, a conversion between the current block and a bitstream representation of the video.

[00216] FIG. 37 is a flowchart for a method 3700 for video processing. The method 3700 includes, at operation 3710, determining, for a current block of a video, that an application of an sub-block based temporal motion vector prediction (SbTMVP) tool is enabled in case that a current picture of the current block is a reference picture with an index set to M in a reference picture list X, M and X being integers.

[00217] The method 3700 includes, at operation 3720, performing, based on the determining, a conversion between the current block and a bitstream representation of the video.

[00218] FIG. 38 is a flowchart for a method 3800 for video processing. The method 3800 includes, at operation 3810, performing a conversion between a current block of a video and a bitstream representation of the video, the current block being coded using a sub-block based coding tool, wherein performing the conversion comprises using a plurality of bins (N) to code a sub-block merge index with a unified method in case that a sub-block based temporal motion vector prediction (SbTMVP) tool is enabled or disabled.

[00219] FIG. 39 is a flowchart for a method 3900 for video processing. The method 3900 includes, at operation 3910, determining, for a current block of a video coded using a sub-block based temporal motion vector prediction (SbTMVP) tool, a motion vector used by the SbTMVP tool to locate a corresponding block in a picture different from a current picture comprising the current block.

[00220] The method 3900 includes, at operation 3920, performing, based on the determining, a conversion between the current block and a bitstream representation of the video.

[00221] FIG. 40 is a flowchart for a method 4000 for video processing. The method 4000 includes, at operation 4010, determining, for a conversion between a current block of a video and a bitstream representation of the video, whether a zero motion affine merge candidate is inserted into a sub-block merge candidate list based on whether affine prediction is enabled for the conversion of the current block.

[00222] The method 4000 includes, at operation 4020, performing, based on the determining, the conversion.

[00223] FIG. 41 is a flowchart for a method 4100 for video processing. The method 4100 includes, at operation 4110, inserting, for a conversion between a current block of a video and a bitstream representation of the video that uses a sub-block merge candidate list, zero motion non-affine padding candidates into the sub-block merge candidate list in case that the sub-block merge candidate list is not fulfilled.

[00224] The method 4100 includes, at operation 4120, performing, subsequent to the inserting, the conversion.

[00225] FIG. 42 is a flowchart for a method 4200 for video processing. The method 4200 includes, at operation 4210, determining, for a conversion between a current block of a video and a bitstream representation of the video, motion vectors using a rule that determines that the motion vectors are derived from one or more motion vectors of a block covering a corresponding position in a collocated picture.

[00226] The method 4200 includes, at operation 4220, performing, based on the motion vectors, the conversion.

[00227] FIG. 43 is a flowchart for a method 4300 for video processing. The method 4300 includes, at operation 4310, determining, for a conversion between a current block of video and a bitstream representation of the video, to use an sub-block based temporal motion vector prediction (sbTMVP) tool with a default motion candidate for the conversion in case that a temporal block associated with the current block or a sub-block of the current block in a collocated picture is coded using a coding mode in which a video unit is reconstructed based on previously coded samples in a same picture.

[00228] The method 4300 includes, at operation 4320, performing, based on the default motion candidate, the conversion.

[00229] FIG. 44 is a flowchart for a method 4400 for video processing. The method 4400 includes, at operation 4410, deriving, for an sub-block based temporal motion vector prediction (sbTMVP) process that is part of a conversion between a current block of video and a bitstream representation of the video, default motion information for the sbTMVP process based on a location of a position that is used in a sub-block motion information derivation process for a sub-block of the current block.

[00230] The method 4400 includes, at operation 4420, performing, based on the default motion information, the conversion.

[00231] FIG. 45 is a flowchart for a method 4500 for video processing. The method 4500 includes, at operation 4510, determining, for a conversion between a current block of a video coded using an sub-block based temporal motion vector prediction (sbTMVP) tool and a bitstream representation of the video, a modified motion vector used to locate a corresponding block in a different picture from the current picture of the current block, wherein the modified motion vector is generated by right-shifting a motion vector, used for prediction in the sbTMVP tool, to an integer precision.

[00232] The method 4500 includes, at operation 4520, performing, based on the determining, the conversion.

[00233] FIG. 46 is a block diagram of a video processing apparatus 4600. The apparatus 4600 may be used to implement one or more of the methods described herein. The apparatus 4600 may be embodied in a smartphone, tablet, computer, Internet of Things (IoT) receiver, and so on. The apparatus 4600 may include one or more processors 4602, one or more memories 4604 and video processing hardware 4606. The processor(s) 4602 may be configured to implement one or more methods described in the present document. The memory (memories) 4604 may be used for storing data and code used for implementing the methods and techniques described herein. The video processing hardware 4606 may be used to implement, in hardware circuitry, some techniques described in the present document. In some embodiments, the video processing hardware 4606 may be at least partly internal to the processor 4602, e.g., a graphics co-processor.

[00234] In some embodiments, the video coding methods may be implemented using an apparatus that is implemented on a hardware platform as described with respect to FIG. 46.

[00235] 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.

[00236] 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.

[00237] FIG. 47 is a block diagram showing an example video processing system 4700 in which various techniques disclosed herein may be implemented. Various implementations may include some or all of the components of the system 4700. The system 4700 may include input 4702 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 4702 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.

[00238] The system 4700 may include a coding component 4704 that may implement the various coding or encoding methods described in the present document. The coding component 4704 may reduce the average bitrate of video from the input 4702 to the output of the coding component 4704 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 4704 may be either stored, or transmitted via a communication connected, as represented by the component 4706. The stored or communicated bitstream (or coded) representation of the video received at the input 4702 may be used by the component 4708 for generating pixel values or displayable video that is sent to a display interface 4710. 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.

[00239] 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.

[00240] In some embodiments, the following technical solutions can be implemented:

[00241] A1. A method of video processing, comprising: determining, for a conversion between a current block of video and a bitstream representation of the video, a maximum number of candidates (ML) in a sub-block based merge candidate list and/or whether to add sub-block based temporal motion vector prediction (SbTMVP) candidates to the sub-block based merge candidate list based on whether temporal motion vector prediction (TMVP) is enabled for use during the conversion or whether a current picture referencing (CPR) coding mode is used for the conversion; and performing, based on the determining, the conversion.

[00242] A2. The method of solution A1, wherein the use of the SbTMVP candidates is disabled due to a determination that the TMVP tool is disabled or an SbTMVP tool is disabled.

[00243] A3. The method of solution A2, wherein determining ML comprises: excluding the SbTMVP candidates from the sub-block based merge candidate list based on whether the SbTMVP tool or the TMVP tool are disabled.

[00244] A4. A method of video processing, comprising: determining, for a conversion between a current block of video and a bitstream representation of the video, a maximum number of candidates (ML) in a sub-block based merge candidate list based on whether a temporal motion

vector prediction (TMVP), a sub-block based temporal motion vector prediction (SbTMVP), and an affine coding mode are enabled for use during the conversion; and performing, based on the determining, the conversion.

[00245] A5. The method of solution A4, wherein ML is set on-the-fly and signaled in the bitstream representation due to a determination that the affine coding mode is enabled.

[00246] A6. The method of solution A4, wherein ML is predefined due to a determination that the affine coding mode is disabled.

[00247] A7. The method of solution A2 or A6, wherein determining ML comprises: setting ML to zero due to a determination that the TMVP tool is disabled, an SbTMVP tool is enabled and the affine coding mode for the current block is disabled.

[00248] A8. The method of solution A2 or A6, wherein determining ML comprises: setting ML to one due to a determination that an SbTMVP tool is enabled, the TMVP tool is enabled and the affine coding mode for the current block is disabled.

[00249] A9. The method of solution A1, wherein the use of the SbTMVP candidates is disabled due to a determination that a SbTMVP tool is disabled or a collocated reference picture of a current picture of the current block is the current picture.

[00250] A10. The method of solution A9, wherein determining ML comprises: excluding the SbTMVP candidates from the sub-block based merge candidate list based on whether the SbTMVP tool is disabled or the collocated reference picture of the current picture is the current picture.

[00251] A11. The method of solution A9, wherein determining ML comprises: setting ML to zero due to a determination that the collocated reference picture of the current picture is the current picture, and affine coding for the current block is disabled.

[00252] A12. The method of solution A9, wherein determining ML comprises: setting ML to one due to a determination that the SbTMVP tool is enabled, the collocated reference picture of the current picture is not the current picture, and affine coding for the current block is disabled.

[00253] A13. The method of solution A1, wherein the use of the SbTMVP candidates is disabled due to a determination that an SbTMVP tool is disabled or a reference picture with reference picture index 0 in reference picture list 0 (L0) is a current picture of the current block.

[00254] A14. The method of solution A13, wherein determining ML comprises: excluding the SbTMVP candidates from the sub-block based merge candidate list based on whether the SbTMVP tool is disabled or the reference picture with reference picture index 0 in L0 is the current picture.

[00255] A15. The method of solution A10 or A13 wherein determining ML comprises: setting ML to zero due to a determination that the SbTMVP tool is enabled, the reference picture with reference picture index 0 in L0 is the current picture, and affine coding for the current block is disabled.

[00256] A16. The method of solution A13, wherein determining ML comprises: setting ML to one due to a determination that the SbTMVP tool is enabled, the reference picture with reference picture index 0 in L0 is not the current picture, and affine coding for the current block is disabled.

[00257] A17. The method of solution A1, wherein the use of the SbTMVP candidates is disabled due to a determination that a SbTMVP tool is disabled or a reference picture with reference picture index 0 in reference picture list 1 (L1) is a current picture of the current block.

[00258] A18. The method of solution A17, wherein determining ML comprises: excluding the SbTMVP candidates from the sub-block based merge candidate list based on whether the SbTMVP tool is disabled or the reference picture with reference picture index 0 in L1 is the current picture.

[00259] A19. The method of solution A17, wherein determining ML comprises: setting ML to zero due to a determination that the SbTMVP tool is enabled, the reference picture with reference picture index 0 in L1 is the current picture, and affine coding for the current block is disabled.

[00260] A20. The method of solution A17, wherein determining ML comprises: setting ML to one due to a determination that the SbTMVP tool is enabled, the reference picture with reference picture index 0 in L1 is not the current picture, and affine coding for the current block is disabled.

[00261] A21. A method of video processing, comprising: determining, for a conversion between a current block of a first video segment of a video and a bitstream representation of the video, that a sub-block based motion vector prediction (SbTMVP) mode is disabled for the conversion due to a temporal motion vector prediction (TMVP) mode being disabled at a first video segment level; and performing the conversion based on the determining, wherein the bitstream representation complies with a format that specifies whether an indication of the SbTMVP mode is included and/or a position of the indication of the SbTMVP mode in a merge candidate list, with respect to an indication of the TMVP mode.

[00262] A22. The method of solution A21, wherein the first video segment is a sequence, slice, a tile or a picture.

[00263] A23. The method of solution A21, wherein the format specifies an omission of the indication of the SbTMVP mode due to an inclusion of the indication of the TMVP mode at the first video segment level.

[00264] A24. The method of solution A21, wherein the format specifies that the indication of the SbTMVP mode is at the first video segment level after the indication of the TMVP mode in a decoding order.

[00265] A25. The method of any of solutions A21 to A24, wherein the format specifies that the indication of the SbTMVP mode is omitted due to a determination that the TMVP mode is indicated as being disabled.

[00266] A26. The method of solution A21, wherein the format specifies that the indication of the SbTMVP mode is included at a sequence level for the video and omitted at a second video segment level.

[00267] A27. The method of solution A26, wherein a second video segment at the second video segment level is a slice, a tile or a picture.

[00268] A28. The method of any of solutions A1 to A27, the conversion generates the current block from the bitstream representation.

[00269] A29. The method of any of solutions A1 to A27, wherein the conversion generates the bitstream representation from the current block.

[00270] A30. The method of any of solutions A1 to A27, wherein performing the conversion comprises parsing the bitstream representation based on one or more decoding rules.

[00271] A31. An apparatus in a video system comprising a processor and a non-transitory memory with instructions thereon, wherein the instructions upon execution by the processor, cause the processor to implement the method in any one of solutions A1 to A30.

[00272] A32. A computer program product stored on a non-transitory computer readable media, the computer program product including program code for carrying out the method in any one of solutions A1 to A30.

[00273] In some embodiments, the following technical solutions can be implemented:

[00274] B1. A method of video processing, comprising: performing a conversion between a current block of a video that is coded using a sub-block based temporal motion vector prediction (SbTMVP) tool or a temporal motion vector prediction (TMVP) tool and a bitstream representation of the video, wherein a coordinate of a corresponding position of the current block or a sub-block

of the current block is selectively masked using a mask based on a compression of motion vectors associated with the SbTMVP tool or the TMVP tool, and wherein an application of the mask comprises computing a bitwise AND operation between a value of the coordinate and a value of the mask.

[00275] B2. The method of solution B1, wherein the coordinate is (x_N, y_N) and the mask (MASK) is an integer equal to $\sim(2M-1)$, wherein M is an integer, wherein the application of the mask results in a masked coordinate $(x_{N'}, y_{N'})$, wherein $x_{N'} = x_N \& \text{MASK}$ and $y_{N'} = y_N \& \text{MASK}$, and wherein “ $\sim$ ” is the bitwise NOT operation and “ $\&$ ” is the bitwise AND operation.

[00276] B3. The method of solution B2, wherein $M = 3$ or $M = 4$.

[00277] B4. The method of solution B2 or B3, wherein a plurality of sub-blocks of size $2K \times 2K$ share a same motion information based on the compression of the motion vectors, and wherein K is an integer not equal to M.

[00278] B5. The method of solution B4, wherein $M = K + 1$.

[00279] B6. The method of solution B1, wherein the mask is not applied upon a determination that the motion vectors associated with the SbTMVP tool or the TMVP tool are not compressed.

[00280] B7. The method of any of solutions B1 to B6, wherein the mask for the SbTMVP tool is identical to the mask for the TMVP tool.

[00281] B8. The method of any of solutions B1 to B6, wherein the mask for the ATMVP tool is different from the mask for the TMVP tool.

[00282] B9. The method of solution B1, wherein a type of the compression is no compression, 8×8 compression or 16×16 compression.

[00283] B10. The method of solution B9, wherein the type of the compression is signaled in a video parameter set (VPS), a sequence parameter set (SPS), a picture parameter set (PPS), a slice header or a tile group header.

[00284] B11. The method of solution B9 or B10, wherein the type of the compression is based on a standard profile, level or tier corresponding to the current block.

[00285] B12. A method of video processing, comprising: determining, based on one or more characteristics of a current block of a video segment of a video, a valid corresponding region of the current block for an application of a sub-block based motion vector prediction (SbTMVP) tool on the current block; and performing, based on the determining, a conversion between the current block and a bitstream representation of the video.

[00286] B13. The method of solution B12, wherein the one or more characteristics comprises a height or a width of the current block.

[00287] B14. The method of solution B12, wherein the one or more characteristics comprises a type of a compression of motion vectors associated with the current block.

[00288] B15. The method of solution B14, wherein the valid corresponding region is a first size due to a determination that the type of the compression comprises no compression, and wherein the valid corresponding region is a second size that is larger than the first size due to a determination that the type of compression comprises $K \times K$ compression.

[00289] B16. The method of solution B12, wherein a size of the valid corresponding region is based on a basic region with size $M \times N$ that is smaller than a size of a coding tree unit (CTU) region, and wherein a size of the current block is $W \times H$.

[00290] B17. The method of solution B16, wherein the size of the CTU region is 128×128 , and wherein $M = 64$ and $N = 64$.

[00291] B18. The method of solution B16, wherein the valid corresponding region is a collocated basic region and an extension in a collocated picture due to a determination that $W \leq M$ and $H \leq N$.

[00292] B19. The method of solution B16, wherein the current block is partitioned into several parts upon a determination that $W > M$ and $H > N$, and wherein each of the several parts comprises an individual valid corresponding region for the application of the SbTMVP tool.

[00293] B20. A method of video processing, comprising: determining, for a current block of a video that is coded using a sub-block based temporal motion vector prediction (SbTMVP) tool, a default motion vector; and performing, based on the determining, a conversion between the current block and a bitstream representation of the video, wherein the default motion vector is determined due to a determination that a motion vector is not obtained from a block covering a corresponding position in the collocated picture that is associated with a center position of the current block.

[00294] B21. The method of solution B20, wherein the default motion vector is set to $(0, 0)$.

[00295] B22. The method of solution B20, wherein the default motion vector is derived from a history-based motion vector prediction (HMVP) table.

[00296] B23. The method of solution B22, wherein the default motion vector is set to $(0, 0)$ due to a determination that the HMVP table is empty.

[00297] B24. The method of solution B22, wherein the default motion vector is predefined and signaled in a video parameter set (VPS), a sequence parameter set (SPS), a picture parameter set (PPS), a slice header, a tile group header, a coding tree unit (CTU) or a coding unit (CU).

[00298] B25. The method of solution B22, wherein the default motion vector is set to a first element stored in the HMVP table due to a determination that the HMVP table is non-empty.

[00299] B26. The method of solution B22, wherein the default motion vector is set to a last element stored in the HMVP table due to a determination that the HMVP table is non-empty.

[00300] B27. The method of solution B22, wherein the default motion vector is set to a specific motion vector stored in the HMVP table due to a determination that the HMVP table is non-empty.

[00301] B28. The method of solution B27, wherein the specific motion vector refers to reference list 0.

[00302] B29. The method of solution B27, wherein the specific motion vector refers to reference list 1.

[00303] B30. The method of solution B27, wherein the specific motion vector refers to a specific reference picture in reference list 0.

[00304] B31. The method of solution B27, wherein the specific motion vector refers to a specific reference picture in reference list 1.

[00305] B32. The method of solution B30 or B31, wherein the specific reference picture has index 0.

[00306] B33. The method of solution B27, wherein the specific motion vector refers to a collocated picture.

[00307] B34. The method of solution B22, wherein the default motion vector is set to a predefined default motion vector due to a determination that a search process in the HMVP table cannot find a specific motion vector.

[00308] B35. The method of solution B34, wherein the search process searches only the first element or only the last element of the HMVP table.

[00309] B36. The method of solution B34, wherein the search process searches only a subset of elements of the HMVP table.

[00310] B37. The method of solution B22, wherein the default motion vector does not refer to a current picture of the current block.

[00311] B38. The method of solution B22, wherein the default motion vector is scaled to a collocated picture due to a determination that the default motion vector does not refer to the collocated picture.

[00312] B39. The method of solution B20, wherein the default motion vector is derived from a neighboring block.

[00313] B40. The method of solution B39, wherein an upper right corner of the neighboring block (A0) is directly adjacent to a lower left corner to the current block, or a lower right corner of the neighboring block (A1) is directly adjacent to the lower left corner to the current block, or a lower left corner of the neighboring block (B0) is directly adjacent to an upper right corner of the current block, or a lower right corner of the neighboring block (B1) is directly adjacent to the upper right corner of the current block, or a lower right corner of the neighboring block (B2) is directly adjacent to an upper left corner of the current block.

[00314] B41. The method of solution B40, wherein the default motion vector is derived from only one of neighboring blocks A0, A1, B0, B1 and B2.

[00315] B42. The method of solution B40, wherein the default motion vector is derived from one or more of neighboring blocks A0, A1, B0, B1 and B2.

[00316] B43. The method of solution B40, wherein the default motion vector is set to a predefined default motion vector due to a determination that a valid default motion vector cannot be found in any of the neighboring blocks A0, A1, B0, B1 and B2.

[00317] B44. The method of solution B43, wherein the predefined default motion vector is signaled in a video parameter set (VPS), a sequence parameter set (SPS), a picture parameter set (PPS), a slice header, a tile group header, a coding tree unit (CTU) or a coding unit (CU).

[00318] B45. The method of solution B43 or B44, wherein the predefined default motion vector is (0, 0).

[00319] B46. The method of solution B39, wherein the default motion vector is set to a specific motion vector from the neighboring block.

[00320] B47. The method of solution B46, wherein the specific motion vector refers to reference list 0.

[00321] B48. The method of solution B46, wherein the specific motion vector refers to reference list 1.

[00322] B49. The method of solution B46, wherein the specific motion vector refers to a specific reference picture in reference list 0.

[00323] B50. The method of solution B46, wherein the specific motion vector refers to a specific reference picture in reference list 1.

[00324] B51. The method of solution B49 or B50, wherein the specific reference picture has index 0.

[00325] B52. The method of solution B46, wherein the specific motion vector refers to a collocated picture.

[00326] B53. The method of solution B20, wherein the default motion vector is used due to a determination that the block covering the corresponding position in the collocated picture is intra-coded.

[00327] B54. The method of solution B20, wherein the derivation method is modified due to a determination that the block covering the corresponding position in the collocated picture is not located.

[00328] B55. The method of solution B20, wherein a default motion vector candidate is always available.

[00329] B56. The method of solution B20, wherein the default motion vector is derived in an alternate manner upon a determination that a default motion vector candidate is set to unavailable.

[00330] B57. The method of solution B20, wherein an availability of the default motion vector is based on syntax information in the bitstream representation associated with the video segment.

[00331] B58. The method of solution B57, wherein the syntax information comprises an indication of enabling the SbTMVP tool, and wherein the video segment is a slice, a tile or a picture.

[00332] B59. The method of solution B58, wherein a current picture of the current block is not an intra random access point (IRAP) picture and the current picture is not inserted to a reference picture list 0 (L0) with a reference index 0.

[00333] B60. The method of solution B20, wherein a fixed index or a fixed group of indices is assigned to candidates associated with the SbTMVP tool due to a determination that the SbTMVP tool is enabled, and wherein the fixed index or the fixed group of indices is assigned to candidates associated with a coding tool other than the SbTMVP tool due to a determination that the SbTMVP tool is disabled.

[00334] B61. A method of video processing, comprising: inferring, for a current block of a video segment of a video, that a sub-block based temporal motion vector prediction (SbTMVP) tool or a temporal motion vector prediction (TMVP) tool is disabled for the video segment due to a determination that a current picture of the current block is a reference picture with an index set to M in a reference picture list X, wherein M and X are integers, and wherein $X = 0$ or $X = 1$; and performing, based on the inferring, a conversion between the current block and a bitstream representation of the video.

[00335] B62. The method of solution B61, wherein M corresponds to a target reference picture index that motion information of a temporal block is scaled to for the reference picture list X for the SbTMVP tool or the TMVP tool.

[00336] B63. The method of solution B61, wherein the current picture is an intra random access point (IRAP) picture.

[00337] B64. A method of video processing, comprising: determining, for a current block of a video, that an application of an sub-block based temporal motion vector prediction (SbTMVP) tool is enabled due to a determination that a current picture of the current block is a reference picture with an index set to M in a reference picture list X, wherein M and X are integers; and performing, based on the determining, a conversion between the current block and a bitstream representation of the video.

[00338] B65. The method of solution B64, wherein motion information corresponding to each sub-block of the current block refers to the current picture.

[00339] B66. The method of solution B64, wherein motion information for a sub-block of the current block is derived from a temporal block, and wherein the temporal block is coded with at least one reference picture that refers to a current picture of the temporal block.

[00340] B67. The method of solution B66, wherein the conversion excludes a scaling operation.

[00341] B68. A method of video processing, comprising: performing a conversion between a current block of a video and a bitstream representation of the video, wherein the current block is coded using a sub-block based coding tool, and wherein performing the conversion comprises using a plurality of bins (N) to code a sub-block merge index with a unified method due to a determination that a sub-block based temporal motion vector prediction (SbTMVP) tool is enabled or disabled.

[00342] B69. The method of solution B68, wherein a first number of bins (L) of the plurality of bins are context coded, and wherein a second number of bins ($N-L$) are bypass coded.

[00343] B70. The method of solution B69, wherein $L = 1$.

[00344] B71. The method of solution B68, wherein each of the plurality of bins is context coded.

[00345] B72. The method of any of solutions B1 to B71, the conversion generates the current block from the bitstream representation.

[00346] B73. The method of any of solutions B1 to B71, wherein the conversion generates the bitstream representation from the current block.

[00347] B74. The method of any of solutions B1 to B71, wherein performing the conversion comprises parsing the bitstream representation based on one or more decoding rules.

[00348] B75. An apparatus in a video system comprising a processor and a non-transitory memory with instructions thereon, wherein the instructions upon execution by the processor, cause the processor to implement the method in any one of solutions B1 to B74.

[00349] B76. A computer program product stored on a non-transitory computer readable media, the computer program product including program code for carrying out the method in any one of solutions B1 to B74.

[00350] In some embodiments, the following technical solutions can be implemented:

[00351] C1. A method of video processing, comprising: determining, for a current block of a video coded using a subblock-based temporal motion vector prediction (SbTMVP) tool, a motion vector used by the SbTMVP tool to locate a corresponding block in a picture different from a current picture comprising the current block; and performing, based on the determining, a conversion between the current block and a bitstream representation of the video.

[00352] C2. The method of solution C1, wherein the motion vector is set to a default motion vector.

[00353] C3. The method of solution C2, wherein the default motion vector is (0, 0).

[00354] C4. The method of solution C2, wherein the default motion vector is signaled in a video parameter set (VPS), a sequence parameter set (SPS), a picture parameter set (PPS), a slice header, a tile group header, a coding tree unit (CTU) or a coding unit (CU).

[00355] C5. The method of solution C1, wherein motion vector is set to a motion vector stored in a history-based motion vector prediction (HMVP) table.

[00356] C6. The method of solution C5, wherein the motion vector is set to a default motion vector due to a determination that the HMVP table is empty.

- [00357] C7. The method of solution C6, wherein the default motion vector is (0, 0).
- [00358] C8. The method of solution C5, wherein the motion vector is set to a first motion vector stored in the HMVP table due to a determination that the HMVP table is non-empty.
- [00359] C9. The method of solution C5, wherein the motion vector is set to a last motion vector stored in the HMVP table due to a determination that the HMVP table is non-empty.
- [00360] C10. The method of solution C5, wherein the motion vector is set to a specific motion vector stored in the HMVP table due to a determination that the HMVP table is non-empty.
- [00361] C11. The method of solution C10, wherein the specific motion vector refers to reference list 0.
- [00362] C12. The method of solution C10, wherein the specific motion vector refers to reference list 1.
- [00363] C13. The method of solution C10, wherein the specific motion vector refers to a specific reference picture in reference list 0.
- [00364] C14. The method of solution C10, wherein the specific motion vector refers to a specific reference picture in reference list 1.
- [00365] C15. The method of solution C13 or 14, wherein the specific reference picture has index 0.
- [00366] C16. The method of solution C10, wherein the specific motion vector refers to a collocated picture.
- [00367] C17. The method of solution C5, wherein the motion vector is set to a default motion vector due to a determination that a search process in the HMVP table cannot find a specific motion vector.
- [00368] C18. The method of solution C17, wherein the search process searches only the first element or only the last element of the HMVP table.
- [00369] C19. The method of solution C17, wherein the search process searches only a subset of elements of the HMVP table.
- [00370] C20. The method of solution C5, wherein the motion vector stored in the HMVP table does not refer to the current picture.
- [00371] C21. The method of solution C5, wherein the motion vector stored in the HMVP table is scaled to a collocated picture due to a determination that the motion vector stored in the HMVP table does not refer to the collocated picture.

[00372] C22. The method of solution C1, wherein the motion vector is set to a specific motion vector of a specific neighboring block.

[00373] C23. The method of solution C22, wherein an upper right corner of the specific neighboring block (A0) is directly adjacent to a lower left corner to the current block, or a lower right corner of the specific neighboring block (A1) is directly adjacent to the lower left corner to the current block, or a lower left corner of the specific neighboring block (B0) is directly adjacent to an upper right corner of the current block, or a lower right corner of the specific neighboring block (B1) is directly adjacent to the upper right corner of the current block, or a lower right corner of the specific neighboring block (B2) is directly adjacent to an upper left corner of the current block.

[00374] C24. The method of solution C1, wherein the motion vector is set to a default motion vector due to a determination that a specific neighboring block does not exist.

[00375] C25. The method of solution C1, wherein the motion vector is set to a default motion vector due to a determination that a specific neighboring block is not inter-coded.

[00376] C26. The method of solution C22, wherein the specific motion vector refers to reference list 0.

[00377] C27. The method of solution C22, wherein the specific motion vector refers to reference list 1.

[00378] C28. The method of solution C22, wherein the specific motion vector refers to a specific reference picture in reference list 0.

[00379] C29. The method of solution C22, wherein the specific motion vector refers to a specific reference picture in reference list 1.

[00380] C30. The method of solution C28 or C29, wherein the specific reference picture has index 0.

[00381] C31. The method of solution C22 or C23, wherein the specific motion vector refers to a collocated picture.

[00382] C32. The method of solution C22 or C23, wherein the motion vector is set to a default motion vector due to a determination that the specific neighboring block does not refer to the collocated picture.

[00383] C33. The method of any of solutions C24 to C32, wherein the default motion vector is (0, 0).

[00384] C34. The method of solution C1, wherein the motion vector is set to a default motion vector due to a determination that a specific motion vector stored in a specific neighboring block cannot be found.

[00385] C35. The method of solution C22, wherein the specific motion vector is scaled to a collocated picture due to a determination that the specific motion vector not refer to the collocated picture.

[00386] C36. The method of solution C22, wherein the specific motion vector does not refer to the current picture.

[00387] C37. The method of any of solutions C1 to C36, the conversion generates the current block from the bitstream representation.

[00388] C38. The method of any of solutions C1 to C36, wherein the conversion generates the bitstream representation from the current block.

[00389] C39. The method of any of solutions C1 to C36, wherein performing the conversion comprises parsing the bitstream representation based on one or more decoding rules.

[00390] C40. An apparatus in a video system comprising a processor and a non-transitory memory with instructions thereon, wherein the instructions upon execution by the processor, cause the processor to implement the method in any one of solutions C1 to C39.

[00391] C41. A computer program product stored on a non-transitory computer readable media, the computer program product including program code for carrying out the method in any one of solutions C1 to C39.

[00392] In some embodiments, the following technical solutions can be implemented:

[00393] D1. A method of video processing, comprising: determining, for a conversion between a current block of a video and a bitstream representation of the video, whether a zero motion affine merge candidate is inserted into a sub-block merge candidate list based on whether affine prediction is enabled for the conversion of the current block; and performing, based on the determining, the conversion.

[00394] D2. The method of solution D1, wherein the zero motion affine merge candidate is not inserted into the sub-block merge candidate list due to a determination that an affine usage flag in the bitstream representation is off.

[00395] D3. The method of solution D2, further comprising: inserting default motion vector candidates that are non-affine candidates into the sub-block merge candidate list due to a determination that the affine usage flag is off.

[00396] D4. A method of video processing, comprising: inserting, for a conversion between a current block of a video and a bitstream representation of the video that uses a sub-block merge candidate list, zero motion non-affine padding candidates into the sub-block merge candidate list due to a determination that the sub-block merge candidate list is not fulfilled; and performing, subsequent to the inserting, the conversion.

[00397] D5. The method of solution D4, further comprising: setting an affine usage flag of the current block to zero.

[00398] D6. The method of solution D4, wherein the inserting is further based on whether an affine usage flag in the bitstream representation is off.

[00399] D7. A method of video processing, comprising: determining, for a conversion between a current block of a video and a bitstream representation of the video, motion vectors using a rule that determines that the motion vectors are derived from one or more motion vectors of a block covering a corresponding position in a collocated picture; and performing, based on the motion vectors, the conversion.

[00400] D8. The method of solution D7, wherein the one or more motion vectors comprise MV0 and MV1 that represent motion vectors in reference list 0 and reference list 1, respectively, and wherein the motion vectors to be derived comprise MV0' and MV1' that represent motion vectors in the reference list 0 and the reference list 1.

[00401] D9. The method of solution D8, wherein MV0' and MV1' are derived based on MV0 due to a determination that a collocated picture is in the reference list 0.

[00402] D10. The method of solution D8, wherein MV0' and MV1' are derived based on MV1 due to a determination that a collocated picture is in the reference list 1.

[00403] D11. The method of any of solutions D1 to D10, the conversion generates the current block from the bitstream representation.

[00404] D12. The method of any of solutions D1 to D10, wherein the conversion generates the bitstream representation from the current block.

[00405] D13. The method of any of solutions D1 to D10, wherein performing the conversion comprises parsing the bitstream representation based on one or more decoding rules.

[00406] D14. An apparatus in a video system comprising a processor and a non-transitory memory with instructions thereon, wherein the instructions upon execution by the processor, cause the processor to implement the method in any one of solutions D1 to D13.

[00407] D15. A computer program product stored on a non-transitory computer readable media, the computer program product including program code for carrying out the method in any one of solutions D1 to D13.

[00408] In some embodiments, the following technical solutions can be implemented:

[00409] E1. A method of video processing, comprising: determining, for a conversion between a current block of video and a bitstream representation of the video, to use an sub-block based temporal motion vector prediction (sbTMVP) tool with a default motion candidate for the conversion in case that a temporal block associated with the current block or a sub-block of the current block in a collocated picture is coded using a coding mode in which a video unit is reconstructed based on previously coded samples in a same picture; and performing, based on the default motion candidate, the conversion.

[00410] E2. The method of solution E1, wherein the default motion candidate is used instead of a sub-block or non-sub-block sbTMVP candidate.

[00411] E3. The method of solution E1 or E2, wherein the default motion candidate comprises a motion candidate associated with a center position of the current block.

[00412] E4. The method of solution E1 or E2, wherein the default motion candidate is (0, 0), and wherein the reference picture index is zero for reference picture list 0 or reference picture list 1.

[00413] E5. The method of any of solutions E1 to E4, wherein the coding mode is an intra block copy (IBC) mode.

[00414] E6. A method of video processing, comprising: deriving, for an sub-block based temporal motion vector prediction (sbTMVP) process that is part of a conversion between a current block of video and a bitstream representation of the video, default motion information for the sbTMVP process based on a location of a position that is used in a sub-block motion information derivation process for a sub-block of the current block; and performing, based on the default motion information, the conversion.

[00415] E7. The method of solution E6, wherein the position is a center position of a specific sub-block of the current block instead of a center position of the current block.

[00416] E8. The method of solution E7, wherein the specific sub-block is a center sub-block.

[00417] E9. The method of any of solutions E1 to E8, the conversion generates the current block from the bitstream representation.

[00418] E10. The method of any of solutions E1 to E8, wherein the conversion generates the bitstream representation from the current block.

[00419] E11. The method of any of solutions E1 to E8, wherein performing the conversion comprises parsing the bitstream representation based on one or more decoding rules.

[00420] E12. An apparatus in a video system comprising a processor and a non-transitory memory with instructions thereon, wherein the instructions upon execution by the processor, cause the processor to implement the method in any one of solutions E1 to E11.

[00421] E13. A computer program product stored on a non-transitory computer readable media, the computer program product including program code for carrying out the method in any one of solutions E1 to E11.

[00422] In some embodiments, the following technical solutions can be implemented:

[00423] F1. A method of video processing, comprising: determining, for a conversion between a current block of a video coded using an sub-block based temporal motion vector prediction (sbTMVP) tool and a bitstream representation of the video, a modified motion vector used to locate a corresponding block in a different picture from the current picture of the current block, wherein the modified motion vector is generated by right-shifting a motion vector, used for prediction in the sbTMVP tool, to an integer precision; and performing, based on the determining, the conversion.

[00424] F2. The method of solution F1, wherein the right-shifting uses a rounding operation that is identical to that used in a motion vector scaling process.

[00425] F3. The method of solution F1, wherein the right-shifting uses a rounding operation that is identical to that used in a motion vector averaging process.

[00426] F4. The method of solution F1, wherein the right-shifting uses a rounding operation that is identical to that used in an adaptive motion vector resolution (AMVR) process.

[00427] F5. The method of solution F1, wherein the right-shifting uses a rounding toward zero operation, wherein the motion vector is denoted $MV = (MV_x, MV_y)$, and wherein a right-shifted motion vector is denoted $MV' = (MV'_x, MV'_y)$.

[00428] F6. The method of solution F5, wherein the right-shifted motion vector is computed as: $MV'_x = (MV_x + ((1 \ll N) \gg 1) - (MV_x \geq 0 ? 1 : 0)) \gg N$, and $MV'_y = (MV_y + ((1 \ll$

$N) \gg 1) - (MV_y \geq 0 ? 1 : 0) \gg N$, wherein N is an integer representing a motion vector resolution.

[00429] F7. The method of solution F6, wherein $N = 4$.

[00430] F8. The method of solution F5, wherein the right-shifted motion vector is computed as: $MV_x' = (MV_x + (MV_x \geq 0 ? 7 : 8)) \gg 4$, and $MV_y' = (MV_y + (MV_y \geq 0 ? 7 : 8)) \gg 4$.

[00431] F9. The method of any of solutions F1 to F8, wherein the modified motion vector is further used to locate another corresponding block to derive default motion information used in the sbTMVP tool.

[00432] F10. The method of solution F9, wherein deriving the default motion information is further based on a center position of the current block and the modified motion vector.

[00433] F11. The method of solution F9, wherein deriving the default motion information for a sub-block of the current block is further based on a center position of the sub-block and the modified motion vector.

[00434] F12. The method of any of solutions F1 to F11, wherein the conversion uses another coding tool, different from the sbTMVP tool, that uses the modified motion vector to locate a reference block in the different picture or the current picture.

[00435] F13. The method of any of solutions F1 to F12, the conversion generates the current block from the bitstream representation.

[00436] F14. The method of any of solutions F1 to F12, wherein the conversion generates the bitstream representation from the current block.

[00437] F15. The method of any of solutions F1 to F12, wherein performing the conversion comprises parsing the bitstream representation based on one or more decoding rules.

[00438] F16. An apparatus in a video system comprising a processor and a non-transitory memory with instructions thereon, wherein the instructions upon execution by the processor, cause the processor to implement the method in any one of solutions F1 to F15.

[00439] F17. A computer program product stored on a non-transitory computer readable media, the computer program product including program code for carrying out the method in any one of solutions F1 to F15.

[00440] 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.

[00441] 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.

[00442] 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).

[00443] 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.

[00444] 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.

[00445] 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.

[00446] 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.