

MOTION VECTOR PREDICTOR LIST GENERATION FOR INTRA BLOCK COPY MODE IN VIDEO CODING

[0001] This application claims priority to U.S. Application No. **16/782,735**, filed February 5, 2020 which claims priority to U.S. Provisional Application No. **62/802,573**, filed February 7, 2019, the entire contents of which are incorporated by reference herein.

TECHNICAL FIELD

[0002] This disclosure relates to video encoding and video decoding.

BACKGROUND

[0003] Digital video capabilities can be incorporated into a wide range of devices, including digital televisions, digital direct broadcast systems, wireless broadcast systems, personal digital assistants (PDAs), laptop or desktop computers, tablet computers, e-book readers, digital cameras, digital recording devices, digital media players, video gaming devices, video game consoles, cellular or satellite radio telephones, so-called “smart phones,” video teleconferencing devices, video streaming devices, and the like. Digital video devices implement video coding techniques, such as those described in the standards defined by MPEG-2, MPEG-4, ITU-T H.263, ITU-T H.264/MPEG-4, Part 10, Advanced Video Coding (AVC), the High Efficiency Video Coding (HEVC) standard, ITU-T H.265/High Efficiency Video Coding (HEVC), and extensions of such standards. The video devices may transmit, receive, encode, decode, and/or store digital video information more efficiently by implementing such video coding techniques.

[0004] Video coding techniques include spatial (intra-picture) prediction and/or temporal (inter-picture) prediction to reduce or remove redundancy inherent in video sequences. For block-based video coding, a video slice (e.g., a video picture or a portion of a video picture) may be partitioned into video blocks, which may also be referred to as coding tree units (CTUs), coding units (CUs) and/or coding nodes. Video blocks in an intra-coded (I) slice of a picture are encoded using spatial prediction with respect to reference samples in neighboring blocks in the same picture. Video blocks in an inter-coded (P or B) slice of a picture may use spatial prediction with respect to reference samples in neighboring blocks in the same picture or temporal prediction with

respect to reference samples in other reference pictures. Pictures may be referred to as frames, and reference pictures may be referred to as reference frames.

SUMMARY

[0005] In one example, a method of coding video data includes responsive to determining to predict a current block of a current picture of video data using intra block copy (IBC), generating a motion vector (MV) candidate list for the current block, wherein generating the MV candidate list comprises: determining one or more history-based motion vector prediction (HMVP) candidates; and including, as a last candidate in the MV candidate list, a HMVP candidate of the one or more HMVP candidates; selecting, from the MV candidate list, a particular MV that identifies a predictor block in the current picture; and reconstructing pixels of the current block based on pixels of the predictor block.

[0006] In another example, a device for coding video data includes a memory configured to store the video data; and one or more processors that are implemented in circuitry and configured to: generate, responsive to determining to predict a current block of a current picture of video data using IBC, a MV candidate list for the current block, wherein, to generate the MV candidate list, the one or more processors are configured to: determine one or more HMVP candidates; and include, as a last candidate in the MV candidate list, a HMVP candidate of the one or more HMVP candidates; select, from the MV candidate list, a particular MV that identifies a predictor block in the current picture; and reconstruct pixels of the current block based on pixels of the predictor block.

[0007] In another example, a computer-readable storage medium stores instructions that, when executed, cause one or more processors of a video coder to generate, responsive to determining to predict a current block of a current picture of video data using IBC, a MV candidate list for the current block, wherein the instructions that cause the one or more processors to generate the MV candidate list comprise instructions that cause the one or more processors to: determine one or more HMVP candidates; and include, as a last candidate in the MV candidate list, a HMVP candidate of the one or more HMVP candidates; select, from the MV candidate list, a particular MV that identifies a predictor block in the current picture; and reconstruct pixels of the current block based on pixels of the predictor block.

[0008] The details of one or more aspects of the disclosure are set forth in the accompanying drawings and the description below. Other features, objects, and advantages of the techniques described in this disclosure will be apparent from the description and drawings, and from the claims.

BRIEF DESCRIPTION OF DRAWINGS

[0009] FIGS. 1A and 1B are conceptual diagrams illustrating spatial neighboring candidates in HEVC.

[0010] FIGS. 2A and 2B are conceptual diagrams illustrating example temporal motion vector predictor (TMVP) candidates and motion vector (MV) scaling.

[0011] FIG. 3 is a block diagram illustrating an example video encoding and decoding system that may perform the techniques of this disclosure.

[0012] FIGS. 4A and 4B are conceptual diagrams illustrating an example quadtree binary tree (QTBT) structure, and a corresponding coding tree unit (CTU).

[0013] FIG. 5 is a block diagram illustrating an example video encoder that may perform the techniques of this disclosure.

[0014] FIG. 6 is a block diagram illustrating an example video decoder that may perform the techniques of this disclosure.

[0015] FIG. 7 is a flowchart illustrating an example method for encoding a current block.

[0016] FIG. 8 is a flowchart illustrating an example method for decoding a current block of video data.

[0017] FIG. 9 is a flowchart illustrating an example method for generating a motion vector predictor candidate list, in accordance with one or more techniques of this disclosure.

DETAILED DESCRIPTION

[0018] In general, this disclosure describes techniques for predicting blocks of video data using intra block copy (IBC). To predict a current block of video data in a current picture of video data using IBC, a video coder may add residual data to a predictor block in the current picture. The video coder may identify the predictor block using a motion vector, which may be referred to as a block vector, that represents a displacement between the current block and the predictor block. To enable a video

decoder to obtain the motion vector, both video encoder and video decoder may construct a motion vector candidate list and the video encoder may encode one or more syntax elements that indicate which candidate from the motion vector list is to be used as a motion vector predictor to construct the motion vector for the current block. In some examples, the video encoder may further encode one or more syntax elements that represent a difference between the motion vector predictor and the motion vector for the current block (e.g., syntax elements that represent a motion vector difference (MVD)).

[0019] The video coder may obtain candidates for the motion vector candidate list from a variety of sources. Some example sources include, but are not limited to, spatial candidates, temporal motion vector predictor (TMVP) candidates, history-based motion vector prediction (HMVP) candidates, and pairwise candidates. To generate the motion vector candidate list, the video coder may evaluate candidates from the sources until a desired quantity of candidates is included in the list. To evaluate a particular candidate for inclusion in the motion vector candidate list, the video coder may determine whether the particular candidate is available and/or whether a value of the particular candidate is the same as a candidate already included in the motion vector candidate list. The video coder may evaluate candidates from the sources in a particular order.

[0020] In some cases, a video coder may utilize the order of sources to generate a motion vector candidate list for a current block regardless of whether the current block is predicted using inter-mode (e.g. using a predictor block located in a different picture than the current block) or using IBC (e.g., using a predictor block located in the same picture as the current block). In other cases, a video coder may utilize different orders of sources for generating motion vector candidate lists for blocks predicting using inter-mode and blocks predicted using IBC. Where the video coder uses different orders, there may still be some overlap in the candidates considered. For instance, the video coder may evaluate HMVP candidates for inclusion in the motion vector candidate list for both blocks coded using inter and IBC with the condition that a last candidate in the motion vector candidate list cannot be a HMVP candidate.

[0021] In accordance with one or more techniques of this disclosure, a video coder may generate a motion vector candidate list for a block of video data predicted using IBC with an HMVP candidate as the last candidate in the motion vector candidate list. For instance, the video coder may include, as a last candidate in a motion vector candidate list for a current block predicted using IBC, an HMVP candidate.

[0022] In some examples, in addition to allowing HMVP candidates to be the last candidate in a motion vector candidate list, the video coder may not utilize some candidate sources for IBC candidate lists that are used for inter candidate lists. For instance, the video coder may not evaluate pairwise candidates when generating candidate lists for blocks coded using IBC and may evaluate pairwise candidates when generating candidate lists for blocks coded using inter.

[0023] The techniques of this disclosure are related to the intra block copy (IBC) mode and shared motion vector predictor list design. The techniques may be applied to any of the existing video codecs, such as HEVC (High Efficiency Video Coding), VVC (Versatile Video Coding) or be an efficient coding tool in any future video coding standards. Below, HEVC and JEM techniques and on-going works in Versatile Video Coding (VVC) related to this disclosure are firstly reviewed.

[0024] Video coding standards include ITU-T H.261, ISO/IEC MPEG-1 Visual, ITU-T H.262 or ISO/IEC MPEG-2 Visual, ITU-T H.263, ISO/IEC MPEG-4 Visual and ITU-T H.264 (also known as ISO/IEC MPEG-4 AVC), including its Scalable Video Coding (SVC) and Multi-view Video Coding (MVC) extensions, High Efficiency Video Coding (HEVC) or ITU-T H.265, including its range extension, multiview extension (MV-HEVC) and scalable extension (SHVC).

[0025] ITU-T VCEG (Q6/16) and ISO/IEC MPEG (JTC 1/SC 29/WG 11) are now studying the potential need for standardization of future video coding technology with a compression capability that significantly exceeds that of the current HEVC standard (including its current extensions and near-term extensions for screen content coding and high-dynamic-range coding). The groups are working together on this exploration activity in a joint collaboration effort known as the Joint Video Exploration Team (JVET) to evaluate compression technology designs proposed by their experts in this area. The JVET first met during 19–21 October 2015. And the latest version of reference software, i.e., Joint Exploration Model 7 (JEM 7) is available at jvet.hhi.fraunhofer.de/svn/svn_HMJEMSoftware/tags/HM-16.6-JEM-7.0/. An Algorithm description of Joint Exploration Test Model 7 (JEM7), hereinafter “JVET-G1001, is available at phenix.it-sudparis.eu/jvet/doc_end_user/current_document.php?id=3286.

[0026] CU structure and motion vector prediction in HEVC. In HEVC, the largest coding unit in a slice is called a coding tree block (CTB) or coding tree unit (CTU). A CTB contains a quad-tree the nodes of which are coding units. The size of a CTB can

be ranges from 16x16 to 64x64 in the HEVC main profile (although 8x8 CTB sizes can be supported). A coding unit (CU) could be the same size of a CTB to as small as 8x8. Each coding unit is coded with one mode, *i.e.* inter or intra. When a CU is inter coded, the CU may be further partitioned into 2 or 4 prediction units (PUs) or become just one PU when further partition doesn't apply. When two PUs are present in one CU, they can be half size rectangles or two rectangle size with $\frac{1}{4}$ or $\frac{3}{4}$ size of the CU. When the CU is inter coded, each PU has one set of motion information, which is derived with a unique inter prediction mode.

[0027] Motion prediction. In the HEVC standard, there are two inter prediction modes, named merge (skip is considered as a special case of merge) and advanced motion vector prediction (AMVP) modes respectively for a prediction unit (PU).

[0028] In either AMVP or merge mode, a motion vector (MV) candidate list is maintained for multiple motion vector predictors. The motion vector(s), as well as reference indices in the merge mode, of the current PU are generated by taking one candidate from the MV candidate list.

[0029] The MV candidate list may contain up to five candidates for the merge mode and only two candidates for the AMVP mode. A merge candidate may contain a set of motion information, e.g., motion vectors corresponding to both reference picture lists (list 0 and list 1) and the reference indices. If a merge candidate is identified by a merge index, the reference pictures are used for the prediction of the current blocks, as well as the associated motion vectors are determined. However, under AMVP mode for each potential prediction direction from either list 0 or list 1, a reference index needs to be explicitly signaled, together with an MVP index to the MV candidate list since the AMVP candidate contains only a motion vector. In AMVP mode, the predicted motion vectors can be further refined.

[0030] As can be seen above, a merge candidate corresponds to a full set of motion information while an AMVP candidate contains just one motion vector for a specific prediction direction and reference index. The candidates for both modes are derived similarly from the same spatial and temporal neighboring blocks.

[0031] FIGS. 1A and 1B are conceptual diagrams illustrating spatial neighboring candidates in HEVC. Spatial MV candidates are derived from the neighboring blocks shown on FIGS. 1A and 1B, for a specific PU (PU0), although the methods of generating the candidates from the blocks differ for merge and AMVP modes.

[0032] In merge mode, up to four spatial MV candidates can be derived with the orders shown in FIG. 1A with numbers, and the order is the following: left (0, A1), above (1, B1), above-right (2, B0), below-left (3, A0), and above left (4, B2), as shown in FIG. 1A. That is, in FIG. 1A, block 1000 includes PU0 1040A and PU1 1040B. When a video coder is to code motion information for PU0 1040A using merge mode, the video coder adds motion information from spatial neighboring blocks 1080A, 1080B, 1080C, 1080D, and 1080E to a candidate list, in that order. Blocks 1080A, 1080B, 1080C, 1080D, and 1080E may also be referred to as, respectively, blocks A1, B1, B0, A0, and B2, as in HEVC.

[0033] In AVMP mode, the neighboring blocks are divided into two groups: a left group including blocks 0 and 1, and an above group including blocks 2, 3, and 4 as shown on FIG. 1B. These blocks are labeled, respectively, as blocks 1100A, 1100B, 1100C, 1100D, and 1100E in FIG. 1B. In particular, in FIG. 1B, block 1020 includes PU0 1060A and PU1 1060B, and blocks 1100A, 1100B, 1100C, 1100D, and 1100E represent spatial neighbors to PU0 1060A. For each group, the potential candidate in a neighboring block referring to the same reference picture as that indicated by the signaled reference index has the highest priority to be chosen to form a final candidate of the group. It is possible that all neighboring blocks do not contain a motion vector pointing to the same reference picture. Therefore, if such a candidate cannot be found, the first available candidate will be scaled to form the final candidate; thus, the temporal distance differences can be compensated.

[0034] A temporal motion vector predictor (TMVP) candidate, if enabled and available, is added into the MV candidate list after spatial motion vector candidates. The process of motion vector derivation for TMVP candidate is the same for both merge and AMVP modes; however, the target reference index for the TMVP candidate in the merge mode is always set to 0. The primary block location for TMVP candidate derivation is the bottom right block outside of the collocated PU (shown as block “T” in FIG. 2A) to compensate the bias to the above and left blocks used to generate spatial neighboring candidates. However, if that block is located outside of the current CTB row or motion information is not available, the block is substituted with a center block of the PU.

[0035] As shown in FIG. 2B, motion vector for TMVP candidate is derived from the co-located PU of the co-located picture, indicated in the slice level. Similar to temporal direct mode in AVC, a motion vector of the TMVP candidate may have motion vector scaling performed to compensate the distance differences.

[0036] Other aspects of motion prediction in HEVC will now be described. One such aspect is motion vector scaling. It is assumed that the value of motion vectors is proportional to the distance of pictures in the presentation time. A motion vector associates two pictures, the reference picture, and the picture containing the motion vector (namely the containing picture). When a motion vector is utilized to predict the other motion vector, the distance of the containing picture and the reference picture is calculated based on the Picture Order Count (POC) values.

[0037] For a motion vector to be predicted, both its associated containing picture and reference picture may be different. Therefore a new distance (based on POC) is calculated. And the motion vector is scaled based on these two POC distances. For a spatial neighboring candidate, the containing pictures for the two motion vectors are the same, while the reference pictures are different. In HEVC, motion vector scaling applies to both TMVP and AMVP for spatial and temporal neighboring candidates.

[0038] Another aspect of motion prediction in HEVC is artificial motion vector candidate generation. If a motion vector candidate list is not complete, artificial motion vector candidates are generated and inserted at the end of the list until it will have all candidates. In merge mode, there are two types of artificial MV candidates: combined candidate derived only for B-slices and zero candidates used only for AMVP if the first type does not provide enough artificial candidates. For each pair of candidates that are already in the candidate list and have necessary motion information, bi-directional combined motion vector candidates are derived by a combination of the motion vector of the first candidate referring to a picture in the list 0 and the motion vector of a second candidate referring to a picture in the list 1.

[0039] Another aspect of motion prediction in HEVC is a pruning process for candidate insertion. Candidates from different blocks may happen to be the same, which decreases the efficiency of a merge mode or AMVP mode candidate list. A pruning process may be applied to solve this problem. The pruning process compares one candidate against the others in the current candidate list to avoid inserting identical candidate in certain extent. To reduce the complexity, only limited numbers of pruning process is applied instead of comparing each potential one with all the other existing ones.

[0040] Enhanced motion vector prediction. In development of Versatile Video Coding (VVC), there are several inter coding tools which derive or refine the candidate list of motion vector prediction or merge prediction for the current block. Several of these

approaches are described below. These approaches include history-based motion vector prediction, pairwise average candidates (also referred to as “pairwise candidates”), and merge list in VTM3.0.

[0041] History-based motion vector prediction. History-based motion vector prediction (HMVP) [JVET-K0104, available at phenix.it-sudparis.eu/jvet/doc_end_user/documents/11_Ljubljana/wg11/JVET-K0104-v5.zip] is a history-based method that allows each block to find its MV predictor from a list of MVs decoded from the past in addition to those in immediately adjacent causal neighboring motion fields. A table with multiple HMVP candidates is maintained during the encoding/decoding process. The table is emptied when a new slice is encountered. Whenever there is an inter-coded block, the associated motion information is inserted to the table in a first-in-first-out (FIFO) fashion as a new HMVP candidate. Then, a constraint FIFO rule can be applied. When inserting a HMVP to the table, redundancy check is firstly applied to find whether there is an identical HMVP in the table. If found, that particular HMVP is removed from the table and all the HMVP candidates afterwards are moved.

[0042] HMVP candidates could be used in the merge candidate list construction process. All HMVP candidates from the last entry to the first entry in the table may be inserted after the TMVP candidate. Pruning may be applied on the HMVP candidates. Once the total number of available merge candidates reaches the signaled maximally allowed merge candidates, the merge candidate list construction process is terminated.

[0043] Similarly, HMVP candidates could also be used in the AMVP candidate list construction process. The motion vectors of the last K HMVP candidates in the table are inserted after the TMVP candidate. Only HMVP candidates with the same reference picture as the AMVP target reference picture are used to construct the AMVP candidate list. Pruning is applied on the HMVP candidates.

[0044] Pairwise average candidates. Pairwise average candidates are used in VTM3.0. Pairwise average candidates are generated by averaging predefined pairs of candidates in the current merge candidate list (includes spatial candidates, TMVP, and HMVP), and the predefined pairs are defined as $\{(0, 1), (0, 2), (1, 2), (0, 3), (1, 3), (2, 3)\}$, where the numbers denote the merge indices to the merge candidate list. The averaged motion vectors are calculated separately for each reference list. If both motion vectors are available in one list, these two motion vectors are averaged even when they point to different reference pictures; if only one motion vector is available, use the one directly;

if no motion vector is available, keep this list invalid. The pairwise average candidates replace the combined candidates in HEVC standard.

[0045] Merge list in VTM3.0. In VTM4.0, for normal inter merge mode the size of merge list is 6, the order of merge candidate list may be as follows:

1. Spatial candidates for blocks A1, B1, B0 and A0.
2. If number of candidates less than 4, add B2.
3. TMVP candidate.
4. HMVP candidates (can't be the last candidate in the list).
5. Pairwise candidates.
6. Zero candidates.

[0046] In VTM4.0, for IBC mode, the size of merge list is 6, the order of merge candidate list is may be as follows:

1. Spatial candidates for blocks A1, B1, B0 and A0.
2. If number of candidates less than 4, add B2.
3. HMVP candidates (can't be the last candidate in the list).
4. Pairwise candidates.

[0047] For IBC mode, if the candidates are valid, the video coder may add them into merge/skip list. A candidate may be considered valid where the following conditions are satisfied: B1 is pruned by A1, if B1 is different from A1, add B1 in merge/skip list; B0 is pruned by B1, A0 is pruned by A1; if the number of candidates is less than four, check B2. B2 is pruned by A1 and B1; The first two HMVP candidates are pruned by A1 and B1; pairwise candidates do not need to be pruned.

[0048] Intra block copy (IBC) is sometimes referred to as current picture referencing (CPR), where a motion vector refers to the already reconstructed reference samples in the current picture. IBC was supported in HEVC screen content coding extension (HEVC SCC). An IBC-coded CU is signaled as an inter coded block. The luma motion (or block) vector of an IBC-coded CU must be in integer precision. The chroma motion vector is clipped to integer precision as well. When combined with AMVR, the IBC mode can switch between 1-pel and 4-pel motion vector precisions. The current picture is placed at the end of the reference picture list L0. To reduce memory consumption and decoder complexity, the IBC in VTM-3.0 allows only the reconstructed portion of the current CTU to be used. This restriction allows the IBC mode to be implemented using local on-chip memory for hardware implementations.

[0049] At the encoder side, hash-based motion estimation may be performed for IBC. The encoder performs RD check for blocks with either width or height no larger than 16 luma samples. For non-merge mode, the block vector search is performed using hash-based search first. If hash search does not return valid candidate, block matching based local search will be performed.

[0050] In VTM4.0, IBC mode is signaled with a block level flag and can be signaled as IBC AMVP mode or IBC skip/merge mode. IBC mode is treated as the third prediction mode other than intra or inter prediction modes (e.g., IBC mode is an independent prediction mode that is different than intra and inter). The current picture may no longer be included as one of the reference pictures in the reference picture list 0. The derivation process of motion vectors for IBC mode excludes all neighboring blocks in inter mode and vice versa. Bitstream conformance checks are no longer needed at the encoder and redundant mode signaling is removed.

[0051] The current IBC techniques in JVET may present one or more problems. IBC mode as an independent mode was adopted in the 13th JVET meeting. The current method of merge/skip candidates list generation for IBC merge/skip mode is similar to normal inter merge/skip mode but IBC mode is different from normal inter prediction.

[0052] As one example problem, for inter merge/skip list generation, if the number of candidates reaches to 4, candidate B2 will not be checked. But IBC mode removed TMVP candidate which reduced the number of potential candidates. As such, it may be desirable for the candidate B2 to be checked even the number of candidates reaches to 4. In accordance with one or more techniques of this disclosure, the merge/skip list may be redesigned for IBC mode.

[0053] As another example problem, the pruning operation of inter prediction need to compare motion direction, reference idx/POC, and motion vector (MV) between different candidates. However, for IBC mode, motion direction and reference idx/POC may always be the same. In accordance with one or more techniques of this disclosure, the pruning algorithm may be simplified.

[0054] As another example problem, HMVP candidates can not be the last candidates in the merge/skip list in VTM. In accordance with one or more techniques of this disclosure, this restriction may be removed.

[0055] As another example problem, only one pairwise candidate can be used for IBC merge/skip mode. However, because there is no zero MV as inter merge/skip mode, the list may less than 6.

[0056] As another example problem, the MV rounding algorithm for inter prediction is unified, but the IBC mode (such as IBC AMVP mode, IBC merge/skip) may use a different MV rounding method for chroma MV. This is not only for IBC merge/skip mode, but also other IBC mode. In accordance with one or more techniques of this disclosure, the rounding method may be unified.

[0057] In accordance with one or more techniques of this disclosure, a video coder (e.g., a video encoder, such as video encoder 200 of FIG. 3 and/or FIG. 5, or a video decoder, such as video decoder 300 of FIG. 3 and/or FIG. 6) may be configured to remove the candidate number restriction of checking B2 candidate. As one example, when generating the IBC merge/skip list, the video coder may always check the candidate of B2. If the candidate of B2 is valid, the video coder may add the candidate of B2 to the candidate list. As another example, when generating the IBC merge/skip list, the video coder may remove the restriction of checking B2 only when candidates less than a threshold (e.g., 2, 3, 4, 5, 6) (e.g., in the current VTM). The video coder may always check B2, if B2 is 1) valid, 2) coded in IBC mode, and 3) pruned by A1 and B1 (e.g., if B2 is different from A1 and B1), the video coder may add B2 to the list.

[0058] In accordance with one or more techniques of this disclosure, the video coder may remove the restriction that HMVP candidates can not be the last candidates in the merge/skip list. For instance, in VTM, the video coder may remove the restriction that HMVP candidates can not be the last IBC merge/skip candidates. The video coder may check the available HMVP candidates in the HMVP table until the number of candidates has reached the size of merge/skip list or have checked all of the candidates in the HMVP table.

[0059] In accordance with one or more techniques of this disclosure, the video coder may remove the pairwise candidates for IBC mode. For example, the video coder may not use pairwise candidates in the IBC merge/skip mode. As one example, the video coder may remove the restriction that only one pairwise candidate can be used. As one example, the video coder may use the available candidates in the merge/skip candidate list to generate pairwise candidates and add into the list until the number of candidates reaches the maximum size of the list. As another example, the video coder may use the first N available candidates in the merge/skip candidate list to generate pairwise candidates, and add into the list until the number of candidates reaches the size of the list. For example, $N = 3$, the pairs will be $\{(0, 1), (0, 2), (1, 2), (0, 3), (1, 3), (2, 3)\}$.

For one example, the video coder may not use HMVP candidates to generate pairwise candidates.

[0060] In accordance with one or more techniques of this disclosure, the video coder may remove the pruning operation from adding a new MV candidate into the MV predictor list. For instance, for generating the IBC merge/skip list, and/or AMVP list, when checking a new MV candidate which is derived from a block coded in IBC mode, the video coder does not need to compare the new MV candidate with the other candidates and may add the new candidate into the list directly.

[0061] As another example, the video coder may simplify the pruning operation by only comparing motion vectors. For instance, for generating the IBC merge/skip list, and/or AMVP list, when checking a new MV candidate which is derived from a block coded in IBC mode, the video coder may perform by only comparing the motion vectors. The motion vector predictor list can be merge/skip list, or AMVP list, or other motion vector predictor candidate list.

[0062] As another example, the video coder may utilize a unified MV rounding method. For instance, the video coder may utilize a modified chroma MV rounding method of IBC as shown below where: i is equal to x or y (the MV components in the horizontal direction and vertical direction). The $shift_i$ may be associated to the purpose, predefined or depends on MV bit precision, chroma format.

$$offset_i = 1 \ll (shift_i - 1)$$

$$MVChroma_i = \begin{cases} -((-MV_i + offset_i) \gg shift_i), & MV_i < 0 \\ (MV_i + offset_i) \gg shift_i, & MV_i \geq 0 \end{cases}$$

[0063] For example, the input MV_i is luma MV which 1/16 precision with the internal bit precision is 4, the chroma format is 4:2:0. In order to generate integer precision chroma MV, the $shift_i$ is equal to $5=4+1$, the $MVChroma_i$ may be determined as shown below:

$$offset_i = 1 \ll (5 - 1)$$

$$MVChroma_i = \begin{cases} -((-MV_i + offset_i) \gg 5), & MV_i < 0 \\ (MV_i + offset_i) \gg 5, & MV_i \geq 0 \end{cases}$$

[0064] For example, the input MV_i is luma MV which 1/16 precision with the internal bit precision is 4, the chroma format is 4:4:4. In order to generate integer precision chroma MV, the $shift_i$ is equal to 4, the $MVChroma_i$ may be determined as shown below:

$$offset_i = 1 \ll (4 - 1)$$

$$MVChorma_i = \begin{cases} -((-MV_i + offset_i) \gg 4), & MV_i < 0 \\ (MV_i + offset_i) \gg 4, & MV_i \geq 0 \end{cases}$$

[0065] FIG. 3 is a block diagram illustrating an example video encoding and decoding system 100 that may perform the techniques of this disclosure. The techniques of this disclosure are generally directed to coding (encoding and/or decoding) video data. In general, video data includes any data for processing a video. Thus, video data may include raw, uncoded video, encoded video, decoded (e.g., reconstructed) video, and video metadata, such as signaling data.

[0066] As shown in FIG. 3, system 100 includes a source device 102 that provides encoded video data to be decoded and displayed by a destination device 116, in this example. In particular, source device 102 provides the video data to destination device 116 via a computer-readable medium 110. Source device 102 and destination device 116 may comprise any of a wide range of devices, including desktop computers, notebook (i.e., laptop) computers, tablet computers, set-top boxes, telephone handsets such as smartphones, televisions, cameras, display devices, digital media players, video gaming consoles, video streaming device, or the like. In some cases, source device 102 and destination device 116 may be equipped for wireless communication, and thus may be referred to as wireless communication devices.

[0067] In the example of FIG. 3, source device 102 includes video source 104, memory 106, video encoder 200, and output interface 108. Destination device 116 includes input interface 122, video decoder 300, memory 120, and display device 118. In accordance with this disclosure, video encoder 200 of source device 102 and video decoder 300 of destination device 116 may be configured to apply the techniques for local illumination compensation. Thus, source device 102 represents an example of a video encoding device, while destination device 116 represents an example of a video decoding device. In other examples, a source device and a destination device may include other components or arrangements. For example, source device 102 may receive video data from an external video source, such as an external camera. Likewise, destination device 116 may interface with an external display device, rather than including an integrated display device.

[0068] System 100 as shown in FIG. 3 is merely one example. In general, any digital video encoding and/or decoding device may perform techniques for local illumination compensation. Source device 102 and destination device 116 are merely examples of such coding devices in which source device 102 generates coded video data for

transmission to destination device 116. This disclosure refers to a “coding” device as a device that performs coding (encoding and/or decoding) of data. Thus, video encoder 200 and video decoder 300 represent examples of coding devices, in particular, a video encoder and a video decoder, respectively. In some examples, devices 102, 116 may operate in a substantially symmetrical manner such that each of devices 102, 116 include video encoding and decoding components. Hence, system 100 may support one-way or two-way video transmission between video devices 102, 116, e.g., for video streaming, video playback, video broadcasting, or video telephony.

[0069] In general, video source 104 represents a source of video data (i.e., raw, uncoded video data) and provides a sequential series of pictures (also referred to as “frames”) of the video data to video encoder 200, which encodes data for the pictures. Video source 104 of source device 102 may include a video capture device, such as a video camera, a video archive containing previously captured raw video, and/or a video feed interface to receive video from a video content provider. As a further alternative, video source 104 may generate computer graphics-based data as the source video, or a combination of live video, archived video, and computer-generated video. In each case, video encoder 200 encodes the captured, pre-captured, or computer-generated video data. Video encoder 200 may rearrange the pictures from the received order (sometimes referred to as “display order”) into a coding order for coding. Video encoder 200 may generate a bitstream including encoded video data. Source device 102 may then output the encoded video data via output interface 108 onto computer-readable medium 110 for reception and/or retrieval by, e.g., input interface 122 of destination device 116.

[0070] Memory 106 of source device 102 and memory 120 of destination device 116 represent general purpose memories. In some example, memories 106, 120 may store raw video data, e.g., raw video from video source 104 and raw, decoded video data from video decoder 300. Additionally or alternatively, memories 106, 120 may store software instructions executable by, e.g., video encoder 200 and video decoder 300, respectively. Although shown separately from video encoder 200 and video decoder 300 in this example, it should be understood that video encoder 200 and video decoder 300 may also include internal memories for functionally similar or equivalent purposes. Furthermore, memories 106, 120 may store encoded video data, e.g., output from video encoder 200 and input to video decoder 300. In some examples, portions of memories 106, 120 may be allocated as one or more video buffers, e.g., to store raw, decoded, and/or encoded video data.

[0071] Computer-readable medium 110 may represent any type of medium or device capable of transporting the encoded video data from source device 102 to destination device 116. In one example, computer-readable medium 110 represents a communication medium to enable source device 102 to transmit encoded video data directly to destination device 116 in real-time, e.g., via a radio frequency network or computer-based network. Output interface 108 may modulate a transmission signal including the encoded video data, and input interface 122 may modulate the received transmission signal, according to a communication standard, such as a wireless communication protocol. The communication medium may comprise any wireless or wired communication medium, such as a radio frequency (RF) spectrum or one or more physical transmission lines. The communication medium may form part of a packet-based network, such as a local area network, a wide-area network, or a global network such as the Internet. The communication medium may include routers, switches, base stations, or any other equipment that may be useful to facilitate communication from source device 102 to destination device 116.

[0072] In some examples, source device 102 may output encoded data from output interface 108 to storage device 116. Similarly, destination device 116 may access encoded data from storage device 116 via input interface 122. Storage device 116 may include any of a variety of distributed or locally accessed data storage media such as a hard drive, Blu-ray discs, DVDs, CD-ROMs, flash memory, volatile or non-volatile memory, or any other suitable digital storage media for storing encoded video data.

[0073] In some examples, source device 102 may output encoded video data to file server 114 or another intermediate storage device that may store the encoded video generated by source device 102. Destination device 116 may access stored video data from file server 114 via streaming or download. File server 114 may be any type of server device capable of storing encoded video data and transmitting that encoded video data to the destination device 116. File server 114 may represent a web server (e.g., for a website), a File Transfer Protocol (FTP) server, a content delivery network device, or a network attached storage (NAS) device. Destination device 116 may access encoded video data from file server 114 through any standard data connection, including an Internet connection. This may include a wireless channel (e.g., a Wi-Fi connection), a wired connection (e.g., DSL, cable modem, etc.), or a combination of both that is suitable for accessing encoded video data stored on file server 114. File server 114 and

input interface 122 may be configured to operate according to a streaming transmission protocol, a download transmission protocol, or a combination thereof.

[0074] Output interface 108 and input interface 122 may represent wireless transmitters/receiver, modems, wired networking components (e.g., Ethernet cards), wireless communication components that operate according to any of a variety of IEEE 802.11 standards, or other physical components. In examples where output interface 108 and input interface 122 comprise wireless components, output interface 108 and input interface 122 may be configured to transfer data, such as encoded video data, according to a cellular communication standard, such as 4G, 4G-LTE (Long-Term Evolution), LTE Advanced, 5G, or the like. In some examples where output interface 108 comprises a wireless transmitter, output interface 108 and input interface 122 may be configured to transfer data, such as encoded video data, according to other wireless standards, such as an IEEE 802.11 specification, an IEEE 802.15 specification (e.g., ZigBee™), a Bluetooth™ standard, or the like. In some examples, source device 102 and/or destination device 116 may include respective system-on-a-chip (SoC) devices. For example, source device 102 may include an SoC device to perform the functionality attributed to video encoder 200 and/or output interface 108, and destination device 116 may include an SoC device to perform the functionality attributed to video decoder 300 and/or input interface 122.

[0075] The techniques of this disclosure may be applied to video coding in support of any of a variety of multimedia applications, such as over-the-air television broadcasts, cable television transmissions, satellite television transmissions, Internet streaming video transmissions, such as dynamic adaptive streaming over HTTP (DASH), digital video that is encoded onto a data storage medium, decoding of digital video stored on a data storage medium, or other applications.

[0076] Input interface 122 of destination device 116 receives an encoded video bitstream from computer-readable medium 110 (e.g., storage device 112, file server 114, or the like). The encoded video bitstream computer-readable medium 110 may include signaling information defined by video encoder 200, which is also used by video decoder 300, such as syntax elements having values that describe characteristics and/or processing of video blocks or other coded units (e.g., slices, pictures, groups of pictures, sequences, or the like). Display device 118 displays decoded pictures of the decoded video data to a user. Display device 118 may represent any of a variety of display devices such as a cathode ray tube (CRT), a liquid crystal display (LCD), a plasma

display, an organic light emitting diode (OLED) display, or another type of display device.

[0077] Although not shown in FIG. 3, in some examples, video encoder 200 and video decoder 300 may each be integrated with an audio encoder and/or audio decoder, and may include appropriate MUX-DEMUX units, or other hardware and/or software, to handle multiplexed streams including both audio and video in a common data stream. If applicable, MUX-DEMUX units may conform to the ITU H.223 multiplexer protocol, or other protocols such as the user datagram protocol (UDP).

[0078] Video encoder 200 and video decoder 300 each may be implemented as any of a variety of suitable encoder and/or decoder circuitry, such as one or more microprocessors, digital signal processors (DSPs), application specific integrated circuits (ASICs), field programmable gate arrays (FPGAs), discrete logic, software, hardware, firmware or any combinations thereof. When the techniques are implemented partially in software, a device may store instructions for the software in a suitable, non-transitory computer-readable medium and execute the instructions in hardware using one or more processors to perform the techniques of this disclosure. Each of video encoder 200 and video decoder 300 may be included in one or more encoders or decoders, either of which may be integrated as part of a combined encoder/decoder (CODEC) in a respective device. A device including video encoder 200 and/or video decoder 300 may comprise an integrated circuit, a microprocessor, and/or a wireless communication device, such as a cellular telephone.

[0079] Video encoder 200 and video decoder 300 may operate according to a video coding standard, such as ITU-T H.265, also referred to as High Efficiency Video Coding (HEVC) or extensions thereto, such as the multi-view and/or scalable video coding extensions. Alternatively, video encoder 200 and video decoder 300 may operate according to other proprietary or industry standards, such as the Joint Exploration Test Model (JEM). The techniques of this disclosure, however, are not limited to any particular coding standard.

[0080] In general, video encoder 200 and video decoder 300 may perform block-based coding of pictures. The term “block” generally refers to a structure including data to be processed (e.g., encoded, decoded, or otherwise used in the encoding and/or decoding process). For example, a block may include a two-dimensional matrix of samples of luminance and/or chrominance data. In general, video encoder 200 and video decoder 300 may code video data represented in a YUV (e.g., Y, Cb, Cr) format. That is, rather

than coding red, green, and blue (RGB) data for samples of a picture, video encoder 200 and video decoder 300 may code luminance and chrominance components, where the chrominance components may include both red hue and blue hue chrominance components. In some examples, video encoder 200 converts received RGB formatted data to a YUV representation prior to encoding, and video decoder 300 converts the YUV representation to the RGB format. Alternatively, pre- and post-processing units (not shown) may perform these conversions.

[0081] This disclosure may generally refer to coding (e.g., encoding and decoding) of pictures to include the process of encoding or decoding data of the picture. Similarly, this disclosure may refer to coding of blocks of a picture to include the process of encoding or decoding data for the blocks, e.g., prediction and/or residual coding. An encoded video bitstream generally includes a series of values for syntax elements representative of coding decisions (e.g., coding modes) and partitioning of pictures into blocks. Thus, references to coding a picture or a block should generally be understood as coding values for syntax elements forming the picture or block.

[0082] HEVC defines various blocks, including coding units (CUs), prediction units (PUs), and transform units (TUs). According to HEVC, a video coder (such as video encoder 200) partitions a coding tree unit (CTU) into CUs according to a quadtree structure. That is, the video coder partitions CTUs and CUs into four equal, non-overlapping squares, and each node of the quadtree has either zero or four child nodes. Nodes without child nodes may be referred to as “leaf nodes,” and CUs of such leaf nodes may include one or more PUs and/or one or more TUs. The video coder may further partition PUs and TUs. For example, in HEVC, a residual quadtree (RQT) represents partitioning of TUs. In HEVC, PUs represent inter-prediction data, while TUs represent residual data. CUs that are intra-predicted include intra-prediction information, such as an intra-mode indication.

[0083] As another example, video encoder 200 and video decoder 300 may be configured to operate according to JEM. According to JEM, a video coder (such as video encoder 200) partitions a picture into a plurality of coding tree units (CTUs). Video encoder 200 may partition a CTU according to a tree structure, such as a quadtree-binary tree (QTBT) structure. The QTBT structure of JEM removes the concepts of multiple partition types, such as the separation between CUs, PUs, and TUs of HEVC. A QTBT structure of JEM includes two levels: a first level partitioned according to quadtree partitioning, and a second level partitioned according to binary

tree partitioning. A root node of the QTBT structure corresponds to a CTU. Leaf nodes of the binary trees correspond to coding units (CUs).

[0084] In some examples, video encoder 200 and video decoder 300 may use a single QTBT structure to represent each of the luminance and chrominance components, while in other examples, video encoder 200 and video decoder 300 may use two or more QTBT structures, such as one QTBT structure for the luminance component and another QTBT structure for both chrominance components (or two QTBT structures for respective chrominance components).

[0085] Video encoder 200 and video decoder 300 may be configured to use quadtree partitioning per HEVC, QTBT partitioning according to JEM, or other partitioning structures. For purposes of explanation, the description of the techniques of this disclosure is presented with respect to QTBT partitioning. However, it should be understood that the techniques of this disclosure may also be applied to video coders configured to use quadtree partitioning, or other types of partitioning as well.

[0086] This disclosure may use “NxN” and “N by N” interchangeably to refer to the sample dimensions of a block (such as a CU or other video block) in terms of vertical and horizontal dimensions, e.g., 16x16 samples or 16 by 16 samples. In general, a 16x16 CU will have 16 samples in a vertical direction ($y = 16$) and 16 samples in a horizontal direction ($x = 16$). Likewise, an NxN CU generally has N samples in a vertical direction and N samples in a horizontal direction, where N represents a nonnegative integer value. The samples in a CU may be arranged in rows and columns. Moreover, CUs need not necessarily have the same number of samples in the horizontal direction as in the vertical direction. For example, CUs may comprise NxM samples, where M is not necessarily equal to N.

[0087] Video encoder 200 encodes video data for CUs representing prediction and/or residual information, and other information. The prediction information indicates how the CU is to be predicted in order to form a prediction block for the CU. The residual information generally represents sample-by-sample differences between samples of the CU prior to encoding and the prediction block.

[0088] To predict a CU, video encoder 200 may generally form a prediction block for the CU through inter-prediction or intra-prediction. Inter-prediction generally refers to predicting the CU from data of a previously coded picture, whereas intra-prediction generally refers to predicting the CU from previously coded data of the same picture. To perform inter-prediction, video encoder 200 may generate the prediction block using

one or more motion vectors. Video encoder 200 may generally perform a motion search to identify a reference block that closely matches the CU, e.g., in terms of differences between the CU and the reference block. Video encoder 200 may calculate a difference metric using a sum of absolute difference (SAD), sum of squared differences (SSD), mean absolute difference (MAD), mean squared differences (MSD), or other such difference calculations to determine whether a reference block closely matches the current CU. In some examples, video encoder 200 may predict the current CU using uni-directional prediction or bi-directional prediction.

[0089] JEM also provides an affine motion compensation mode, which may be considered an inter-prediction mode. In affine motion compensation mode, video encoder 200 may determine two or more motion vectors that represent non-translational motion, such as zoom in or out, rotation, perspective motion, or other irregular motion types.

[0090] To perform intra-prediction, video encoder 200 may select an intra-prediction mode to generate the prediction block. JEM provides sixty-seven intra-prediction modes, including various directional modes, as well as planar mode and DC mode. In general, video encoder 200 selects an intra-prediction mode that describes neighboring samples to a current block (e.g., a block of a CU) from which to predict samples of the current block. Such samples may generally be above, above and to the left, or to the left of the current block in the same picture as the current block, assuming video encoder 200 codes CTUs and CUs in raster scan order (left to right, top to bottom).

[0091] Video encoder 200 encodes data representing the prediction mode for a current block. For example, for inter-prediction modes, video encoder 200 may encode data representing which of the various available inter-prediction modes is used, as well as motion information for the corresponding mode. For uni-directional or bi-directional inter-prediction, for example, video encoder 200 may encode motion vectors using advanced motion vector prediction (AMVP) or merge mode. Video encoder 200 may use similar modes to encode motion vectors for affine motion compensation mode.

[0092] Following prediction, such as intra-prediction or inter-prediction of a block, video encoder 200 may calculate residual data for the block. The residual data, such as a residual block, represents sample by sample differences between the block and a prediction block for the block, formed using the corresponding prediction mode. Video encoder 200 may apply one or more transforms to the residual block, to produce transformed data in a transform domain instead of the sample domain. For example,

video encoder 200 may apply a discrete cosine transform (DCT), an integer transform, a wavelet transform, or a conceptually similar transform to residual video data.

Additionally, video encoder 200 may apply a secondary transform following the first transform, such as a mode-dependent non-separable secondary transform (MDNSST), a signal dependent transform, a Karhunen-Loeve transform (KLT), or the like. Video encoder 200 produces transform coefficients following application of the one or more transforms.

[0093] As noted above, following any transforms to produce transform coefficients, video encoder 200 may perform quantization of the transform coefficients.

Quantization generally refers to a process in which transform coefficients are quantized to possibly reduce the amount of data used to represent the coefficients, providing further compression. By performing the quantization process, video encoder 200 may reduce the bit depth associated with some or all of the coefficients. For example, video encoder 200 may round an n -bit value down to an m -bit value during quantization, where n is greater than m . In some examples, to perform quantization, video encoder 200 may perform a bitwise right-shift of the value to be quantized.

[0094] Following quantization, video encoder 200 may scan the transform coefficients, producing a one-dimensional vector from the two-dimensional matrix including the quantized transform coefficients. The scan may be designed to place higher energy (and therefore lower frequency) coefficients at the front of the vector and to place lower energy (and therefore higher frequency) transform coefficients at the back of the vector. In some examples, video encoder 200 may utilize a predefined scan order to scan the quantized transform coefficients to produce a serialized vector, and then entropy encode the quantized transform coefficients of the vector. In other examples, video encoder 200 may perform an adaptive scan. After scanning the quantized transform coefficients to form the one-dimensional vector, video encoder 200 may entropy encode the one-dimensional vector, e.g., according to context-adaptive binary arithmetic coding (CABAC). Video encoder 200 may also entropy encode values for syntax elements describing metadata associated with the encoded video data for use by video decoder 300 in decoding the video data.

[0095] To perform CABAC, video encoder 200 may assign a context within a context model to a symbol to be transmitted. The context may relate to, for example, whether neighboring values of the symbol are zero-valued or not. The probability determination may be based on a context assigned to the symbol.

[0096] Video encoder 200 may further generate syntax data, such as block-based syntax data, picture-based syntax data, and sequence-based syntax data, to video decoder 300, e.g., in a picture header, a block header, a slice header, or other syntax data, such as a sequence parameter set (SPS), picture parameter set (PPS), or video parameter set (VPS). Video decoder 300 may likewise decode such syntax data to determine how to decode corresponding video data.

[0097] In this manner, video encoder 200 may generate a bitstream including encoded video data, e.g., syntax elements describing partitioning of a picture into blocks (e.g., CUs) and prediction and/or residual information for the blocks. Ultimately, video decoder 300 may receive the bitstream and decode the encoded video data.

[0098] In general, video decoder 300 performs a reciprocal process to that performed by video encoder 200 to decode the encoded video data of the bitstream. For example, video decoder 300 may decode values for syntax elements of the bitstream using CABAC in a manner substantially similar to, albeit reciprocal to, the CABAC encoding process of video encoder 200. The syntax elements may define partitioning information of a picture into CTUs, and partitioning of each CTU according to a corresponding partition structure, such as a QTBT structure, to define CUs of the CTU. The syntax elements may further define prediction and residual information for blocks (e.g., CUs) of video data.

[0099] The residual information may be represented by, for example, quantized transform coefficients. Video decoder 300 may inverse quantize and inverse transform the quantized transform coefficients of a block to reproduce a residual block for the block. Video decoder 300 uses a signaled prediction mode (intra- or inter-prediction) and related prediction information (e.g., motion information for inter-prediction) to form a prediction block for the block. Video decoder 300 may then combine the prediction block and the residual block (on a sample-by-sample basis) to reproduce the original block. Video decoder 300 may perform additional processing, such as performing a deblocking process to reduce visual artifacts along boundaries of the block.

[0100] This disclosure may generally refer to “signaling” certain information, such as syntax elements. The term “signaling” may generally refer to the communication of values syntax elements and/or other data used to decode encoded video data. That is, video encoder 200 may signal values for syntax elements in the bitstream. In general, signaling refers to generating a value in the bitstream. As noted above, source device 102 may transport the bitstream to destination device 116 substantially in real time, or

not in real time, such as might occur when storing syntax elements to storage device 112 for later retrieval by destination device 116.

[0101] FIGS. 4A and 4B are conceptual diagram illustrating an example quadtree binary tree (QTBT) structure 130, and a corresponding coding tree unit (CTU) 132. The solid lines represent quadtree splitting, and dotted lines indicate binary tree splitting. In each split (i.e., non-leaf) node of the binary tree, one flag is signaled to indicate which splitting type (i.e., horizontal or vertical) is used, where 0 indicates horizontal splitting and 1 indicates vertical splitting in this example. For the quadtree splitting, there is no need to indicate the splitting type, since quadtree nodes split a block horizontally and vertically into 4 sub-blocks with equal size. Accordingly, video encoder 200 may encode, and video decoder 300 may decode, syntax elements (such as splitting information) for a region tree level of QTBT structure 130 (i.e., the solid lines) and syntax elements (such as splitting information) for a prediction tree level of QTBT structure 130 (i.e., the dashed lines). Video encoder 200 may encode, and video decoder 300 may decode, video data, such as prediction and transform data, for CUs represented by terminal leaf nodes of QTBT structure 130.

[0102] In general, CTU 132 of FIG. 17B may be associated with parameters defining sizes of blocks corresponding to nodes of QTBT structure 130 at the first and second levels. These parameters may include a CTU size (representing a size of CTU 132 in samples), a minimum quadtree size (MinQTSIZE, representing a minimum allowed quadtree leaf node size), a maximum binary tree size (MaxBTSIZE, representing a maximum allowed binary tree root node size), a maximum binary tree depth (MaxBTDepth, representing a maximum allowed binary tree depth), and a minimum binary tree size (MinBTSIZE, representing the minimum allowed binary tree leaf node size).

[0103] The root node of a QTBT structure corresponding to a CTU may have four child nodes at the first level of the QTBT structure, each of which may be partitioned according to quadtree partitioning. That is, nodes of the first level are either leaf nodes (having no child nodes) or have four child nodes. The example of QTBT structure 130 represents such nodes as including the parent node and child nodes having solid lines for branches. If nodes of the first level are not larger than the maximum allowed binary tree root node size (MaxBTSIZE), they can be further partitioned by respective binary trees. The binary tree splitting of one node can be iterated until the nodes resulting from the split reach the minimum allowed binary tree leaf node size (MinBTSIZE) or the

maximum allowed binary tree depth (MaxBTDepth). The example of QTBT structure 130 represents such nodes as having dashed lines for branches. The binary tree leaf node is referred to as a coding unit (CU), which is used for prediction (e.g., intra-picture or inter-picture prediction) and transform, without any further partitioning. As discussed above, CUs may also be referred to as “video blocks” or “blocks.”

[0104] In one example of the QTBT partitioning structure, the CTU size is set as 128x128 (luma samples and two corresponding 64x64 chroma samples), the MinQTSIZE is set as 16x16, the MaxBTSIZE is set as 64x64, the MinBTSIZE (for both width and height) is set as 4, and the MaxBTDepth is set as 4. The quadtree partitioning is applied to the CTU first to generate quad-tree leaf nodes. The quadtree leaf nodes may have a size from 16x16 (i.e., the MinQTSIZE) to 128x128 (i.e., the CTU size). If the leaf quadtree node is 128x128, it will not be further split by the binary tree, since the size exceeds the MaxBTSIZE (i.e., 64x64, in this example). Otherwise, the leaf quadtree node will be further partitioned by the binary tree. Therefore, the quadtree leaf node is also the root node for the binary tree and has the binary tree depth as 0. When the binary tree depth reaches MaxBTDepth (4, in this example), no further splitting is permitted. When the binary tree node has width equal to MinBTSIZE (4, in this example), it implies no further horizontal splitting is permitted. Similarly, a binary tree node having a height equal to MinBTSIZE implies no further vertical splitting is permitted for that binary tree node. As noted above, leaf nodes of the binary tree are referred to as CUs, and are further processed according to prediction and transform without further partitioning.

[0105] FIG. 5 is a block diagram illustrating an example video encoder 200 that may perform the techniques of this disclosure. FIG. 5 is provided for purposes of explanation and should not be considered limiting of the techniques as broadly exemplified and described in this disclosure. For purposes of explanation, this disclosure describes video encoder 200 in the context of video coding standards such as the HEVC video coding standard and the H.266 video coding standard in development. However, the techniques of this disclosure are not limited to these video coding standards, and are applicable generally to video encoding and decoding.

[0106] In the example of FIG. 5, video encoder 200 includes video data memory 230, mode selection unit 202, residual generation unit 204, transform processing unit 206, quantization unit 208, inverse quantization unit 210, inverse transform processing unit 212, reconstruction unit 214, filter unit 216, decoded picture buffer (DPB) 218, and

entropy encoding unit 220. Any or all of video data memory 230, mode selection unit 202, residual generation unit 204, transform processing unit 206, quantization unit 208, inverse quantization unit 210, inverse transform processing unit 212, reconstruction unit 214, filter unit 216, DPB 218, and entropy encoding unit 220 may be implemented in one or more processors or in processing circuitry. Moreover, video encoder 200 may include additional or alternative processors or processing circuitry to perform these and other functions.

[0107] Video data memory 230 may store video data to be encoded by the components of video encoder 200. Video encoder 200 may receive the video data stored in video data memory 230 from, for example, video source 104 (FIG. 3). DPB 218 may act as a reference picture memory that stores reference video data for use in prediction of subsequent video data by video encoder 200. Video data memory 230 and DPB 218 may be formed by any of a variety of memory devices, such as dynamic random access memory (DRAM), including synchronous DRAM (SDRAM), magnetoresistive RAM (MRAM), resistive RAM (RRAM), or other types of memory devices. Video data memory 230 and DPB 218 may be provided by the same memory device or separate memory devices. In various examples, video data memory 230 may be on-chip with other components of video encoder 200, as illustrated, or off-chip relative to those components.

[0108] In this disclosure, reference to video data memory 230 should not be interpreted as being limited to memory internal to video encoder 200, unless specifically described as such, or memory external to video encoder 200, unless specifically described as such. Rather, reference to video data memory 230 should be understood as reference memory that stores video data that video encoder 200 receives for encoding (e.g., video data for a current block that is to be encoded). Memory 106 of FIG. 3 may also provide temporary storage of outputs from the various units of video encoder 200.

[0109] The various units of FIG. 5 are illustrated to assist with understanding the operations performed by video encoder 200. The units may be implemented as fixed-function circuits, programmable circuits, or a combination thereof. Fixed-function circuits refer to circuits that provide particular functionality, and are preset on the operations that can be performed. Programmable circuits refer to circuits that can be programmed to perform various tasks, and provide flexible functionality in the operations that can be performed. For instance, programmable circuits may execute software or firmware that cause the programmable circuits to operate in the manner

defined by instructions of the software or firmware. Fixed-function circuits may execute software instructions (e.g., to receive parameters or output parameters), but the types of operations that the fixed-function circuits perform are generally immutable. In some examples, the one or more of the units may be distinct circuit blocks (fixed-function or programmable), and in some examples, the one or more units may be integrated circuits.

[0110] Video encoder 200 may include arithmetic logic units (ALUs), elementary function units (EFUs), digital circuits, analog circuits, and/or programmable cores, formed from programmable circuits. In examples where the operations of video encoder 200 are performed using software executed by the programmable circuits, memory 106 (FIG. 3) may store the object code of the software that video encoder 200 receives and executes, or another memory within video encoder 200 (not shown) may store such instructions.

[0111] Video data memory 230 is configured to store received video data. Video encoder 200 may retrieve a picture of the video data from video data memory 230 and provide the video data to residual generation unit 204 and mode selection unit 202. Video data in video data memory 230 may be raw video data that is to be encoded.

[0112] Mode selection unit 202 includes a motion estimation unit 222, motion compensation unit 224, and an intra-prediction unit 226. Mode selection unit 202 may include additional functional units to perform video prediction in accordance with other prediction modes. As examples, mode selection unit 202 may include a palette unit, an intra-block copy unit (which may be part of motion estimation unit 222 and/or motion compensation unit 224), an affine unit, a linear model (LM) unit, or the like.

[0113] Mode selection unit 202 generally coordinates multiple encoding passes to test combinations of encoding parameters and resulting rate-distortion values for such combinations. The encoding parameters may include partitioning of CTUs into CUs, prediction modes for the CUs, transform types for residual data of the CUs, quantization parameters for residual data of the CUs, and so on. Mode selection unit 202 may ultimately select the combination of encoding parameters having rate-distortion values that are better than the other tested combinations.

[0114] Video encoder 200 may partition a picture retrieved from video data memory 230 into a series of CTUs, and encapsulate one or more CTUs within a slice. Mode selection unit 202 may partition a CTU of the picture in accordance with a tree structure, such as the QTBT structure or the quad-tree structure of HEVC described

above. As described above, video encoder 200 may form one or more CUs from partitioning a CTU according to the tree structure. Such a CU may also be referred to generally as a “video block” or “block.”

[0115] In general, mode selection unit 202 also controls the components thereof (e.g., motion estimation unit 222, motion compensation unit 224, and intra-prediction unit 226) to generate a prediction block for a current block (e.g., a current CU, or in HEVC, the overlapping portion of a PU and a TU). For inter-prediction of a current block, motion estimation unit 222 may perform a motion search to identify one or more closely matching reference blocks in one or more reference pictures (e.g., one or more previously coded pictures stored in DPB 218). In particular, motion estimation unit 222 may calculate a value representative of how similar a potential reference block is to the current block, e.g., according to sum of absolute difference (SAD), sum of squared differences (SSD), mean absolute difference (MAD), mean squared differences (MSD), or the like. Motion estimation unit 222 may generally perform these calculations using sample-by-sample differences between the current block and the reference block being considered. Motion estimation unit 222 may identify a reference block having a lowest value resulting from these calculations, indicating a reference block that most closely matches the current block.

[0116] Motion estimation unit 222 may form one or more motion vectors (MVs) that defines the positions of the reference blocks in the reference pictures relative to the position of the current block in a current picture. Motion estimation unit 222 may then provide the motion vectors to motion compensation unit 224. For example, for uni-directional inter-prediction, motion estimation unit 222 may provide a single motion vector, whereas for bi-directional inter-prediction, motion estimation unit 222 may provide two motion vectors. Motion compensation unit 224 may then generate a prediction block using the motion vectors. For example, motion compensation unit 224 may retrieve data of the reference block using the motion vector. As another example, if the motion vector has fractional sample precision, motion compensation unit 224 may interpolate values for the prediction block according to one or more interpolation filters. Moreover, for bi-directional inter-prediction, motion compensation unit 224 may retrieve data for two reference blocks identified by respective motion vectors and combine the retrieved data, e.g., through sample-by-sample averaging or weighted averaging.

[0117] As another example, for intra-prediction, or intra-prediction coding, intra-prediction unit 226 may generate the prediction block from samples neighboring the current block. For example, for directional modes, intra-prediction unit 226 may generally mathematically combine values of neighboring samples and populate these calculated values in the defined direction across the current block to produce the prediction block. As another example, for DC mode, intra-prediction unit 226 may calculate an average of the neighboring samples to the current block and generate the prediction block to include this resulting average for each sample of the prediction block.

[0118] Mode selection unit 202 provides the prediction block to residual generation unit 204. Residual generation unit 204 receives a raw, uncoded version of the current block from video data memory 230 and the prediction block from mode selection unit 202. Residual generation unit 204 calculates sample-by-sample differences between the current block and the prediction block. The resulting sample-by-sample differences define a residual block for the current block. In some examples, residual generation unit 204 may also determine differences between sample values in the residual block to generate a residual block using residual differential pulse code modulation (RDPCM). In some examples, residual generation unit 204 may be formed using one or more subtractor circuits that perform binary subtraction.

[0119] In examples where mode selection unit 202 partitions CUs into PUs, each PU may be associated with a luma prediction unit and corresponding chroma prediction units. Video encoder 200 and video decoder 300 may support PUs having various sizes. As indicated above, the size of a CU may refer to the size of the luma coding block of the CU and the size of a PU may refer to the size of a luma prediction unit of the PU. Assuming that the size of a particular CU is $2N \times 2N$, video encoder 200 may support PU sizes of $2N \times 2N$ or $N \times N$ for intra prediction, and symmetric PU sizes of $2N \times 2N$, $2N \times N$, $N \times 2N$, $N \times N$, or similar for inter prediction. Video encoder 20 and video decoder 30 may also support asymmetric partitioning for PU sizes of $2N \times nU$, $2N \times nD$, $nL \times 2N$, and $nR \times 2N$ for inter prediction.

[0120] In examples where mode selection unit does not further partition a CU into PUs, each CU may be associated with a luma coding block and corresponding chroma coding blocks. As above, the size of a CU may refer to the size of the luma coding block of the CU. The video encoder 200 and video decoder 120 may support CU sizes of $2N \times 2N$, $2N \times N$, or $N \times 2N$.

[0121] For other video coding techniques such as an intra-block copy mode coding, an affine-mode coding, and linear model (LM) mode coding, as few examples, mode selection unit 202, via respective units associated with the coding techniques, generates a prediction block for the current block being encoded. In some examples, such as palette mode coding, mode selection unit 202 may not generate a prediction block, and instead generate syntax elements that indicate the manner in which to reconstruct the block based on a selected palette. In such modes, mode selection unit 202 may provide these syntax elements to entropy encoding unit 220 to be encoded.

[0122] As described above, residual generation unit 204 receives the video data for the current block and the corresponding prediction block. Residual generation unit 204 then generates a residual block for the current block. To generate the residual block, residual generation unit 204 calculates sample-by-sample differences between the prediction block and the current block.

[0123] Transform processing unit 206 applies one or more transforms to the residual block to generate a block of transform coefficients (referred to herein as a “transform coefficient block”). Transform processing unit 206 may apply various transforms to a residual block to form the transform coefficient block. For example, transform processing unit 206 may apply a discrete cosine transform (DCT), a directional transform, a Karhunen-Loeve transform (KLT), or a conceptually similar transform to a residual block. In some examples, transform processing unit 206 may perform multiple transforms to a residual block, e.g., a primary transform and a secondary transform, such as a rotational transform. In some examples, transform processing unit 206 does not apply transforms to a residual block.

[0124] Quantization unit 208 may quantize the transform coefficients in a transform coefficient block, to produce a quantized transform coefficient block. Quantization unit 208 may quantize transform coefficients of a transform coefficient block according to a quantization parameter (QP) value associated with the current block. Video encoder 200 (e.g., via mode selection unit 202) may adjust the degree of quantization applied to the coefficient blocks associated with the current block by adjusting the QP value associated with the CU. Quantization may introduce loss of information, and thus, quantized transform coefficients may have lower precision than the original transform coefficients produced by transform processing unit 206.

[0125] Inverse quantization unit 210 and inverse transform processing unit 212 may apply inverse quantization and inverse transforms to a quantized transform coefficient

block, respectively, to reconstruct a residual block from the transform coefficient block. Reconstruction unit 214 may produce a reconstructed block corresponding to the current block (albeit potentially with some degree of distortion) based on the reconstructed residual block and a prediction block generated by mode selection unit 202. For example, reconstruction unit 214 may add samples of the reconstructed residual block to corresponding samples from the prediction block generated by mode selection unit 202 to produce the reconstructed block.

[0126] Filter unit 216 may perform one or more filter operations on reconstructed blocks. For example, filter unit 216 may perform deblocking operations to reduce blockiness artifacts along edges of CUs. Operations of filter unit 216 may be skipped, in some examples.

[0127] Video encoder 200 stores reconstructed blocks in DPB 218. For instance, in examples where operations of filter unit 216 are not needed, reconstruction unit 214 may store reconstructed blocks to DPB 218. In examples where operations of filter unit 216 are needed, filter unit 216 may store the filtered reconstructed blocks to DPB 218. Motion estimation unit 222 and motion compensation unit 224 may retrieve a reference picture from DPB 218, formed from the reconstructed (and potentially filtered) blocks, to inter-predict blocks of subsequently encoded pictures. In addition, intra-prediction unit 226 may use reconstructed blocks in DPB 218 of a current picture to intra-predict other blocks in the current picture.

[0128] In general, entropy encoding unit 220 may entropy encode syntax elements received from other functional components of video encoder 200. For example, entropy encoding unit 220 may entropy encode quantized transform coefficient blocks from quantization unit 208. As another example, entropy encoding unit 220 may entropy encode prediction syntax elements (e.g., motion information for inter-prediction or intra-mode information for intra-prediction) from mode selection unit 202. Entropy encoding unit 220 may perform one or more entropy encoding operations on the syntax elements, which are another example of video data, to generate entropy-encoded data. For example, entropy encoding unit 220 may perform a context-adaptive variable length coding (CAVLC) operation, a CABAC operation, a variable-to-variable (V2V) length coding operation, a syntax-based context-adaptive binary arithmetic coding (SBAC) operation, a Probability Interval Partitioning Entropy (PIPE) coding operation, an Exponential-Golomb encoding operation, or another type of entropy encoding operation

on the data. In some examples, entropy encoding unit 220 may operate in bypass mode where syntax elements are not entropy encoded.

[0129] Video encoder 200 may output a bitstream that includes the entropy encoded syntax elements needed to reconstruct blocks of a slice or picture. In particular, entropy encoding unit 220 may output the bitstream.

[0130] The operations described above are described with respect to a block. Such description should be understood as being operations for a luma coding block and/or chroma coding blocks. As described above, in some examples, the luma coding block and chroma coding blocks are luma and chroma components of a CU. In some examples, the luma coding block and the chroma coding blocks are luma and chroma components of a PU.

[0131] In some examples, operations performed with respect to a luma coding block need not be repeated for the chroma coding blocks. As one example, operations to identify a motion vector (MV) and reference picture for a luma coding block need not be repeated for identifying a MV and reference picture for the chroma blocks. Rather, the MV for the luma coding block may be scaled to determine the MV for the chroma blocks, and the reference picture may be the same. As another example, the intra-prediction process may be the same for the luma coding blocks and the chroma coding blocks.

[0132] FIG. 6 is a block diagram illustrating an example video decoder 300 that may perform the techniques of this disclosure. FIG. 6 is provided for purposes of explanation and is not limiting on the techniques as broadly exemplified and described in this disclosure. For purposes of explanation, this disclosure describes video decoder 300 is described according to the techniques of JEM and HEVC. However, the techniques of this disclosure may be performed by video coding devices that are configured to other video coding standards.

[0133] In the example of FIG. 6, video decoder 300 includes coded picture buffer (CPB) memory 320, entropy decoding unit 302, prediction processing unit 304, inverse quantization unit 306, inverse transform processing unit 308, reconstruction unit 310, filter unit 312, and decoded picture buffer (DPB) 314. Any or all of CPB memory 320, entropy decoding unit 302, prediction processing unit 304, inverse quantization unit 306, inverse transform processing unit 308, reconstruction unit 310, filter unit 312, and DPB 314 may be implemented in one or more processors or in processing circuitry.

Moreover, video decoder 300 may include additional or alternative processors or processing circuitry to perform these and other functions.

[0134] Prediction processing unit 304 includes motion compensation unit 316 and intra-prediction unit 318. Prediction processing unit 304 may include addition units to perform prediction in accordance with other prediction modes. As examples, prediction processing unit 304 may include a palette unit, an intra-block copy unit (which may form part of motion compensation unit 316), an affine unit, a linear model (LM) unit, or the like. In other examples, video decoder 300 may include more, fewer, or different functional components.

[0135] CPB memory 320 may store video data, such as an encoded video bitstream, to be decoded by the components of video decoder 300. The video data stored in CPB memory 320 may be obtained, for example, from computer-readable medium 110 (FIG. 3). CPB memory 320 may include a CPB that stores encoded video data (e.g., syntax elements) from an encoded video bitstream. Also, CPB memory 320 may store video data other than syntax elements of a coded picture, such as temporary data representing outputs from the various units of video decoder 300. DPB 314 generally stores decoded pictures, which video decoder 300 may output and/or use as reference video data when decoding subsequent data or pictures of the encoded video bitstream. CPB memory 320 and DPB 314 may be formed by any of a variety of memory devices, such as dynamic random access memory (DRAM), including synchronous DRAM (SDRAM), magnetoresistive RAM (MRAM), resistive RAM (RRAM), or other types of memory devices. CPB memory 320 and DPB 314 may be provided by the same memory device or separate memory devices. In various examples, CPB memory 320 may be on-chip with other components of video decoder 300, or off-chip relative to those components.

[0136] Additionally or alternatively, in some examples, video decoder 300 may retrieve coded video data from memory 120 (FIG. 3). That is, memory 120 may store data as discussed above with CPB memory 320. Likewise, memory 120 may store instructions to be executed by video decoder 300, when some or all of the functionality of video decoder 300 is implemented in software to be executed by processing circuitry of video decoder 300.

[0137] The various units shown in FIG. 6 are illustrated to assist with understanding the operations performed by video decoder 300. The units may be implemented as fixed-function circuits, programmable circuits, or a combination thereof. Similar to FIG. 5,

fixed-function circuits refer to circuits that provide particular functionality, and are preset on the operations that can be performed. Programmable circuits refer to circuits that can be programmed to perform various tasks, and provide flexible functionality in the operations that can be performed. For instance, programmable circuits may execute software or firmware that cause the programmable circuits to operate in the manner defined by instructions of the software or firmware. Fixed-function circuits may execute software instructions (e.g., to receive parameters or output parameters), but the types of operations that the fixed-function circuits perform are generally immutable. In some examples, the one or more of the units may be distinct circuit blocks (fixed-function or programmable), and in some examples, the one or more units may be integrated circuits.

[0138] Video decoder 300 may include ALUs, EFUs, digital circuits, analog circuits, and/or programmable cores formed from programmable circuits. In examples where the operations of video decoder 300 are performed by software executing on the programmable circuits, on-chip or off-chip memory may store instructions (e.g., object code) of the software that video decoder 300 receives and executes.

[0139] Entropy decoding unit 302 may receive encoded video data from the CPB and entropy decode the video data to reproduce syntax elements. Prediction processing unit 304, inverse quantization unit 306, inverse transform processing unit 308, reconstruction unit 310, and filter unit 312 may generate decoded video data based on the syntax elements extracted from the bitstream.

[0140] In general, video decoder 300 reconstructs a picture on a block-by-block basis. Video decoder 300 may perform a reconstruction operation on each block individually (where the block currently being reconstructed, i.e., decoded, may be referred to as a “current block”).

[0141] Entropy decoding unit 302 may entropy decode syntax elements defining quantized transform coefficients of a quantized transform coefficient block, as well as transform information, such as a quantization parameter (QP) and/or transform mode indication(s). Inverse quantization unit 306 may use the QP associated with the quantized transform coefficient block to determine a degree of quantization and, likewise, a degree of inverse quantization for inverse quantization unit 306 to apply. Inverse quantization unit 306 may, for example, perform a bitwise left-shift operation to inverse quantize the quantized transform coefficients. Inverse quantization unit 306 may thereby form a transform coefficient block including transform coefficients.

[0142] After inverse quantization unit 306 forms the transform coefficient block, inverse transform processing unit 308 may apply one or more inverse transforms to the transform coefficient block to generate a residual block associated with the current block. For example, inverse transform processing unit 308 may apply an inverse DCT, an inverse integer transform, an inverse Karhunen-Loeve transform (KLT), an inverse rotational transform, an inverse directional transform, or another inverse transform to the coefficient block.

[0143] Furthermore, prediction processing unit 304 generates a prediction block according to prediction information syntax elements that were entropy decoded by entropy decoding unit 302. For example, if the prediction information syntax elements indicate that the current block is inter-predicted, motion compensation unit 316 may generate the prediction block. In this case, the prediction information syntax elements may indicate a reference picture in DPB 314 from which to retrieve a reference block, as well as a motion vector identifying a location of the reference block in the reference picture relative to the location of the current block in the current picture. Motion compensation unit 316 may generally perform the inter-prediction process in a manner that is substantially similar to that described with respect to motion compensation unit 224 (FIG. 5).

[0144] As another example, if the prediction information syntax elements indicate that the current block is intra-predicted, intra-prediction unit 318 may generate the prediction block according to an intra-prediction mode indicated by the prediction information syntax elements. Again, intra-prediction unit 318 may generally perform the intra-prediction process in a manner that is substantially similar to that described with respect to intra-prediction unit 226 (FIG. 5). Intra-prediction unit 318 may retrieve data of neighboring samples to the current block from DPB 314.

[0145] Reconstruction unit 310 may reconstruct the current block using the prediction block and the residual block. For example, reconstruction unit 310 may add samples of the residual block to corresponding samples of the prediction block to reconstruct the current block.

[0146] Filter unit 312 may perform one or more filter operations on reconstructed blocks. For example, filter unit 312 may perform deblocking operations to reduce blockiness artifacts along edges of the reconstructed blocks. Operations of filter unit 312 are not necessarily performed in all examples.

[0147] Video decoder 300 may store the reconstructed blocks in DPB 314. As discussed above, DPB 314 may provide reference information, such as samples of a current picture for intra-prediction and previously decoded pictures for subsequent motion compensation, to prediction processing unit 304. Moreover, video decoder 300 may output decoded pictures from DPB for subsequent presentation on a display device, such as display device 118 of FIG. 3.

[0148] FIG. 7 is a flowchart illustrating an example method for encoding a current block. The current block may comprise a current CU. Although described with respect to video encoder 200 (FIGS. 3 and 5), it should be understood that other devices may be configured to perform a method similar to that of FIG. 7.

[0149] In this example, video encoder 200 initially predicts the current block (350). For example, video encoder 200 may form a prediction block for the current block. Video encoder 200 may then calculate a residual block for the current block (352). To calculate the residual block, video encoder 200 may calculate a difference between the original, uncoded block and the prediction block for the current block. Video encoder 200 may then transform and quantize coefficients of the residual block (354). Next, video encoder 200 may scan the quantized transform coefficients of the residual block (356). During the scan, or following the scan, video encoder 200 may entropy encode the coefficients (358). For example, video encoder 200 may encode the coefficients using CAVLC or CABAC. Video encoder 200 may then output the entropy coded data of the block (360).

[0150] FIG. 8 is a flowchart illustrating an example method for decoding a current block of video data. The current block may comprise a current CU. Although described with respect to video decoder 300 (FIGS. 16 and 19), it should be understood that other devices may be configured to perform a method similar to that of FIG. 8.

[0151] Video decoder 300 may receive entropy coded data for the current block, such as entropy coded prediction information and entropy coded data for coefficients of a residual block corresponding to the current block (370). Video decoder 300 may entropy decode the entropy coded data to determine prediction information for the current block and to reproduce coefficients of the residual block (372). Video decoder 300 may predict the current block (374), e.g., using an intra- or inter-prediction mode as indicated by the prediction information for the current block, to calculate a prediction block for the current block. Video decoder 300 may then inverse scan the reproduced coefficients (376), to create a block of quantized transform coefficients. Video decoder

300 may then inverse quantize and inverse transform the coefficients to produce a residual block (378). Video decoder 300 may ultimately decode the current block by combining the prediction block and the residual block (380).

[0152] FIG. 9 is a flowchart illustrating an example method for generating a motion vector predictor candidate list, in accordance with one or more techniques of this disclosure. The current block may comprise a current CU. Although described with respect to video decoder 300 (FIGS. 16 and 19), it should be understood that other devices may be configured to perform a method similar to that of FIG. 9, such as video encoder 200 (FIGS. 3 and 5).

[0153] As discussed above, a video coder, such as video decoder 300, may construct a motion vector candidate list for a current block of video data. The video coder may obtain candidates for the motion vector candidate list from a variety of sources. Some example sources include, but are not limited to, spatial candidates, temporal motion vector predictor (TMVP) candidates, history-based motion vector prediction (HMVP) candidates, and pairwise candidates. To generate the motion vector candidate list, the video coder may evaluate candidates from the sources until a desired quantity of candidates is included in the list. As shown in FIG. 9, video decoder 300 may determine a maximum number of candidates (*MAX*) to include in a motion vector (MV) list for a current block (902). The value of *MAX* may be fixed, or may be variable based on one or more parameters.

[0154] To evaluate a particular candidate for inclusion in the motion vector candidate list, the video coder may determine whether the particular candidate is available and/or whether a value of the particular candidate is the same as a candidate already included in the motion vector candidate list. The video coder may evaluate candidates from the sources in a particular order.

[0155] As discussed above and in accordance with one or more techniques of this disclosure, a video decoder, such as video decoder 300, may generate motion vector candidate lists for blocks using different sources and/or different restrictions based on whether a block is coded using intra block copy (IBC) or inter coding. Video decoder 300 may determine whether the current block is coded using IBC (902). Responsive to determining that the current block is coded using IBC (“Yes” branch of 902), video decoder 300 may evaluate one or more spatial candidates for inclusion in the MV candidates list (906). For instance, video decoder 300 may evaluate spatial candidates A1, B1, B0, and A0 as discussed above with reference to FIGS. 1A and 1B.

[0156] As discussed above and in accordance with one or more techniques of this disclosure, video decoder 300 may allow a history-based motion vector predictor (HMVP) to be the last candidate included in the MV candidate list (e.g., the final candidate added to the MV candidate list, which may have the highest index value). For instance, as shown in FIG. 9, regardless of whether the number of candidates included in the MV list is less than or equal to the maximum number of candidates to be included in the MV candidate list minus one (e.g., $MAX-1$) (both branches of 908), video decoder 300 may evaluate an HMVP candidate for inclusion in the MV candidate list (910). As discussed above, video decoder 300 may obtain the HMVP candidate from a list of MVs decoded in the past (e.g., video decoder 300 may maintain a table or list of MVs during the decoding process).

[0157] Responsive to determining that the current block is not coded using IBC (e.g., is coded using inter prediction) (“No” branch of 902), video decoder 300 may evaluate one or more spatial candidates for inclusion in the MV candidates list (912). For instance, video decoder 300 may evaluate spatial candidates A1, B1, B0, and A0 as discussed above with reference to FIGS. 1A and 1B.

[0158] In contrast to blocks coded using IBC, video decoder 300 may not allow an HMVP to be the last candidate included in the MV candidate list for blocks coded using inter coding. For instance, as shown in FIG. 9, video decoder 300 may evaluate an HMVP candidate for inclusion in the MV candidate list (916) responsive to determining that the number of candidates included in the MV list is not greater than or equal to the maximum number of candidates to be included in the MV candidate list minus one (e.g., $MAX-1$) (“No” branch of 914). However, responsive to determining that the number of candidates included in the MV list is greater than or equal to the maximum number of candidates to be included in the MV candidate list minus one (e.g., $MAX-1$) (“Yes” branch of 914), video decoder 300 may instead evaluate a pairwise candidate for inclusion in the MV candidate list (918).

[0159] Video decoder 300 may utilize the MV candidate list to reconstruct samples of the current block. For instance, video decoder 300 may select, from the MV candidate list, a candidate MV. In some examples, video decoder 300 may select the candidate MV based on one or more syntax elements that indicate an index value in the MV candidate list. Video decoder 300 may generate a MV for the current block based on the selected candidate MV predictor. For instance, video decoder 300 may decode one or more syntax elements that represent a difference between a value of the selected

candidate MV predictor and the MV for the current block (e.g., and add the difference to the value of the selected candidate MV predictor to obtain the MV for the current block). The MV for the current block may identify a predictor block for the current block. Where the current block is coded using IBC, the predictor block may be in the same picture as the current block. Conversely, where the current block is coded using inter prediction, the predictor block may be in a different picture than the current block. Video decoder 300 may obtain residual data and add the residual data to the predictor block to reconstruct samples of the current block (e.g., 372–380 of FIG. 8). It is noted that video encoder 200 may perform similar operations (e.g., as part of a reconstruction loop).

[0160] The following numbered examples may illustrate one or more aspects of this disclosure:

[0161] Example 1. A method of coding video data, the method comprising: responsive to determining to predict a current block of a current picture of video data using intra block copy (IBC), generating a motion vector (MV) candidate list for the current block, wherein generating the MV candidate list comprises: determining one or more history-based motion vector prediction (HMVP) candidates; and including, as a last candidate in the MV candidate list, an HMVP candidate of the one or more HMVP candidates; selecting, from the MV candidate list, a particular MV candidate that identifies a predictor block in the current picture; and reconstructing pixels of the current block based on pixels of the predictor block.

[0162] Example 2. The method of example 1, wherein the current block is a first block of the current picture, the method further comprising: responsive to determining to predict a second block of the current picture of video data using inter prediction, generating a MV candidate list for the second block, wherein generating the MV candidate list for the second block comprises: determining a maximum number of candidates to be included in the MV candidate list for the second block; and responsive to determining that a quantity of candidates in the MV candidate list for the second block is equal to the maximum number of candidates to be included in the MV candidate list for the second block minus one, refraining from including additional HMVP candidates for the second block in the MV candidate list for the second block such that the last candidate in the MV candidate list for the second block is not an HMVP candidate; selecting, from the MV candidate list for the second block, a particular MV candidate that identifies a predictor block for the second block in a

picture of the video data that is not the same as the current picture; and reconstructing pixels of the second block based on pixels of the predictor block for the second block.

[0163] Example 3. The method of example 2, wherein generating the MV candidate list for the first block comprises not evaluating any pairwise candidates for inclusion in the MV candidate list for the first block.

[0164] Example 4. The method of example 3, wherein generating the MV candidate list for the second block comprises evaluating one or more pairwise candidates for inclusion in the MV candidate list for the second block.

[0165] Example 5. The method of example 4, wherein generating the MV candidate list for the first block further comprises: evaluating an above-left neighboring B2 candidate for inclusion in the MV candidate list for the first block where a current number of candidates included in the MV candidate list for the first block is greater than or equal to four.

[0166] Example 6. The method of example 4 or 5, wherein the MV candidate list for the first block comprises an IBC merge/skip list.

[0167] Example 7. A device for coding video data, the device comprising: a memory configured to store the video data; and one or more processors that are implemented in circuitry and configured to: generate, responsive to determining to predict a current block of a current picture of video data using intra block copy (IBC), a motion vector (MV) candidate list for the current block, wherein, to generate the MV candidate list, the one or more processors are configured to: determine one or more history-based motion vector prediction (HMVP) candidates; and include, as a last candidate in the MV candidate list, an HMVP candidate of the one or more HMVP candidates; select, from the MV candidate list, a particular MV candidate that identifies a predictor block in the current picture; and reconstruct pixels of the current block based on pixels of the predictor block.

[0168] Example 8. The device of example 7, wherein the current block is a first block of the current picture, and wherein the one or more processors are further configured to: generate, responsive to determining to predict a second block of the current picture of video data using inter prediction, a MV candidate list for the second block, wherein, to generate the MV candidate list for the second block, the one or more processors are configured to: determine a maximum number of candidates to be included in the MV candidate list for the second block; and refrain, responsive to determining that a quantity of candidates in the MV candidate list for the second block

is equal to the maximum number of candidates to be included in the MV candidate list for the second block minus one, from including additional HMVP candidates for the second block in the MV candidate list for the second block such that the last candidate in the MV candidate list for the second block is not an HMVP candidate; select, from the MV candidate list for the second block, a particular MV candidate that identifies a predictor block for the second block in a picture of the video data that is not the same as the current picture; and reconstruct pixels of the second block based on pixels of the predictor block for the second block.

[0169] Example 9. The device of example 8, wherein, to generate the MV candidate list for the first block, the one or more processors are configured to not evaluate any pairwise candidates for inclusion in the MV candidate list for the first block.

[0170] Example 10. The device of example 9, wherein, to generate the MV candidate list for the second block, the one or more processors are configured to evaluate one or more pairwise candidates for inclusion in the MV candidate list for the second block.

[0171] Example 11. The device of example 10, wherein, to generate the MV candidate list for the first block, the one or more processors are configured to: evaluate a B2 candidate for inclusion in the MV candidate list for the first block where a current number of candidates included in the MV candidate list for the first block is greater than or equal to four.

[0172] Example 12. The device of example 10, wherein the MV candidate list for the first block comprises an IBC merge/skip list.

[0173] Example 13. The device of any of examples 7–12, wherein the video coder comprises a video decoder.

[0174] Example 14. The device of example 13, further comprising a display configured to display decoded video data.

[0175] Example 15. The device of example 13, wherein the device comprises one or more of a camera, a computer, a mobile device, a broadcast receiver device, or a set-top box.

[0176] Example 16. The device of any of examples 7–15, wherein the device comprises a video encoder.

[0177] Example 17. A computer-readable storage medium storing instructions that, when executed, cause one or more processors of a video coder to: generate, responsive to determining to predict a current block of a current picture of video data using intra block copy (IBC), a motion vector (MV) candidate list for the current block, wherein

the instructions that cause the one or more processors to generate the MV candidate list comprise instructions that cause the one or more processors to: determine one or more history-based motion vector prediction (HMVP) candidates; and include, as a last candidate in the MV candidate list, an HMVP candidate of the one or more HMVP candidates; select, from the MV candidate list, a particular MV candidate that identifies a predictor block in the current picture; and reconstruct pixels of the current block based on pixels of the predictor block.

[0178] Example 18. The computer-readable storage medium of example 17, wherein the current block is a first block of the current picture, and further comprising instructions that cause the one or more processors to: generate, responsive to determining to predict a second block of the current picture of video data using inter prediction, a MV candidate list for the second block, wherein the instructions that cause the one or more processors to generate the MV candidate list for the second block comprise instructions that cause the one or more processors to: determine a maximum number of candidates to be included in the MV candidate list for the second block; and refrain, responsive to determining that a quantity of candidates in the MV candidate list for the second block is equal to the maximum number of candidates to be included in the MV candidate list for the second block minus one, from including additional HMVP candidates for the second block in the MV candidate list for the second block such that the last candidate in the MV candidate list for the second block is not an HMVP candidate; select, from the MV candidate list for the second block, a particular MV candidate that identifies a predictor block for the second block in a picture of the video data that is not the same as the current picture; and reconstruct pixels of the second block based on pixels of the predictor block for the second block.

[0179] Example 19. The computer-readable storage medium of example 18, wherein the instructions that cause the one or more processors to generate the MV candidate list for the first block comprise instructions that cause the one or more processors to not evaluate any pairwise candidates for inclusion in the MV candidate list for the first block.

[0180] Example 20. The computer-readable storage medium of example 19, wherein the instructions that cause the one or more processors to generate the MV candidate list for the second block comprise instructions that cause the one or more processors to evaluate one or more pairwise candidates for inclusion in the MV candidate list for the second block.

[0181] Example 21. A method of coding video data, the method comprising any combination of the techniques described in this disclosure.

[0182] Example 22. A method of coding video data, the method comprising: generating a motion vector (MV) candidate list for a current block of a current picture of video data, wherein generating the MV candidate list comprises: determining a maximum size of the MV candidate list; and responsive to determining that a current number of candidates included in the MV candidate list is less than the maximum size of the MV candidate list, evaluating a B2 candidate for inclusion in the MV candidate list regardless of the current number of candidates included in the MV candidate list; selecting, from the MV candidate list, a particular MV that identifies a predictor block in the current picture; and reconstructing pixels of the current block based on pixels of the predictor block.

[0183] Example 23. The method of example 22, wherein the B2 candidate is an above-left candidate.

[0184] Example 24. The method of any of examples 22–23, wherein the MV candidate list comprises an intra block copy (IBC) merge/skip list.

[0185] Example 25. The method of any of examples 23–24, wherein evaluating the B2 candidate comprises determining whether the B2 candidate is valid, the method further comprising: responsive to determining that the B2 candidate is valid, adding the B2 candidate to the MV candidate list.

[0186] Example 26. The method of any of examples 23–25, wherein evaluating the B2 candidate comprises determining whether the B2 candidate is different than other candidates already included in the MV candidate list, the method further comprising: responsive to determining that the B2 candidate is different than the other candidates already included in the MV candidate list, adding the B2 candidate to the MV candidate list.

[0187] Example 27. A method of coding video data, the method comprising: generating a motion vector (MV) candidate list for a current block of a current picture of video data, wherein generating the MV candidate list comprises: determining one or more history-based motion vector prediction (HMVP) candidates; and including, as a last candidate in the MV candidate list, a HMVP candidate of the one or more HMVP candidates; selecting, from the MV candidate list, a particular MV that identifies a predictor block in the current picture; and reconstructing pixels of the current block based on pixels of the predictor block.

[0188] Example 28. The method of any of examples 22–27, wherein the MV candidate list comprises an intra block copy (IBC) merge/skip list.

[0189] Example 29. A method of coding video data, the method comprising: generating a motion vector (MV) candidate list for a current block of a current picture of video data, wherein generating the MV candidate list comprises evaluating, for inclusion in the MV candidate list, one or more MV candidates, wherein the one or more MV candidates do not include any pairwise candidates; selecting, from the MV candidate list, a particular MV that identifies a predictor block in the current picture; and reconstructing pixels of the current block based on pixels of the predictor block.

[0190] Example 30. The method of any of examples 22–29, wherein the MV candidate list comprises an intra block copy (IBC) merge/skip list.

[0191] Example 31. A method of coding video data, the method comprising: generating a motion vector (MV) candidate list for a current block of a current picture of video data, wherein generating the MV candidate list comprises: determining a plurality pairwise candidates; and including, in the MV candidate list, at least two of the plurality of pairwise candidates; selecting, from the MV candidate list, a particular MV that identifies a predictor block in the current picture; and reconstructing pixels of the current block based on pixels of the predictor block.

[0192] Example 32. The method of example 31, wherein: determining the plurality of pairwise candidates comprises generating, based on candidates included in the MV candidate list, pairwise candidates of the plurality of pairwise candidates; and including the at least two pairwise candidates comprises including, in the MV candidate list, pairwise candidates of the plurality of pairwise candidates until a number of candidates included in the MV candidate list reaches a maximum size.

[0193] Example 33. The method of any of examples 31–32, wherein determining the plurality of pairwise candidates comprises generating, based on candidates included in the MV candidate list other than history-based motion vector prediction (HMVP) candidates, pairwise candidates of the plurality of pairwise candidates.

[0194] Example 34. A method of coding video data, the method comprising: generating a motion vector (MV) candidate list for a current block of a current picture of video data, wherein generating the MV candidate list comprises: determining a plurality of MV candidates; and including, in the MV candidate list, at least one MV candidate of the plurality of MV candidates without evaluating whether the at least one MV candidate is different than MV candidates already included in the MV candidate list;

selecting, from the MV candidate list, a particular MV that identifies a predictor block in the current picture; and reconstructing pixels of the current block based on pixels of the predictor block.

[0195] Example 35. The method of example 34, further comprising: responsive to determining that the at least one MV candidate is derived from a block coded using intra block copy (IBC), determining to include the at least one MV candidate of the plurality of MV candidates in the MV candidate without evaluating whether the at least one MV candidate is different than MV candidates already included in the MV candidate list.

[0196] Example 36. A method of coding video data, the method comprising: generating a motion vector (MV) candidate list for a current block of a current picture of video data, wherein generating the MV candidate list comprises evaluating a respective MV candidate of a plurality of MV candidates for inclusion in the MV candidate list based on motion vectors of the respective MV candidate; selecting, from the MV candidate list, a particular MV that identifies a predictor block in the current picture; and reconstructing pixels of the current block based on pixels of the predictor block.

[0197] Example 37. The method of example 36, wherein evaluating the respective MV candidates based on the motion vectors of the respective MV candidate comprises pruning the respective MV candidate based on a comparison between the motion vectors of the respective MV candidate and motion vectors of candidates included in the MV candidate list.

[0198] Example 38. The method of any of examples 36–37, wherein pruning the respective MV candidate based on a comparison between the motion vectors of the respective MV candidate and motion vectors of candidates included in the MV candidate list comprises: determining whether the motion vectors of the respective MV candidate are different than motion vectors of candidates included in the MV candidate list; and responsive to determining that the motion vectors of the respective MV candidate are different than motion vectors of candidates included in the MV candidate list, including the respective MV candidate in the MV candidate list.

[0199] Example 39. The method of any of examples 36–38, wherein pruning the respective MV candidate based on a comparison between the motion vectors of the respective MV candidate and motion vectors of candidates included in the MV candidate list comprises pruning the respective MV candidate based only on a comparison between the motion vectors of the respective MV candidate and motion vectors of candidates included in the MV candidate list.

[0200] Example 40. The method of any of examples 36–39, wherein pruning the respective MV candidate based on a comparison between the motion vectors of the respective MV candidate and motion vectors of candidates included in the MV candidate list comprises: not pruning the respective MV candidate based on a comparison between reference indices of the respective MV candidate and reference indices of candidates included in the MV candidate list.

[0201] Example 41. The method of any of examples 32–40, wherein the MV candidate list comprises one of: an intra block copy (IBC) merge/skip list; or an advanced motion vector prediction (AMVP) list.

[0202] Example 42. A method of coding video data, the method comprising: obtaining a motion vector for a current block of a current picture of video data, wherein the motion vector identifies a predictor block in the current picture; rounding the motion vector to obtain a chroma motion vector; and reconstructing pixels of the current block based on pixels of the predictor block.

[0203] Example 43. The method of example 42, wherein the motion vector to obtain a chroma motion vector comprises rounding the motion vector in accordance with the following equations:

$$\begin{aligned} offset_i &= 1 \ll (shift_i - 1) \\ MVChorma_i &= \begin{cases} -((-MV_i + offset_i) \gg shift_i), & MV_i < 0 \\ (MV_i + offset_i) \gg shift_i, & MV_i \geq 0 \end{cases} \end{aligned}$$

wherein MV_i represents the motion vector, $MVChorma_i$ represents the chroma motion vector, i is equal to x or y (the MV components in the horizontal direction and vertical direction), and $shift_i$ represents a shift value.

[0204] Example 44. The method of example 43, wherein the shift value is determined based on one or more of MV bit precision, and chroma format.

[0205] Example 45. The method of any of examples 43–44, wherein the shift value is predetermined.

[0206] Example 46. The method of any combination of examples 21–45, wherein coding comprises decoding.

[0207] Example 47. The method of any combination of examples 21–45, wherein coding comprises encoding.

[0208] Example 48. A device for coding video data, the device comprising one or more means for performing the method of any combination of examples 21–45.

[0209] Example 49. The device of example 48, wherein the one or more means comprise one or more processors implemented in circuitry

[0210] Example 50. The device of any combination of examples 48 and 49, further comprising a memory to store the video data.

[0211] Example 51. The device of any combination of examples 48–50, further comprising a display configured to display decoded video data.

[0212] Example 52. The device of any combination of examples 48–51, wherein the device comprises one or more of a camera, a computer, a mobile device, a broadcast receiver device, or a set-top box.

[0213] Example 53. The device of any combination of examples 48–52, wherein the device comprises a video decoder.

[0214] Example 54. The device of any combination of examples 48–53, wherein the device comprises a video encoder.

[0215] Example 55. A computer-readable storage medium having stored thereon instructions that, when executed, cause one or more processors to perform the method of any combination of examples 1–55.

[0216] Example 56. Any combination of examples 1–55.

[0217] It is to be recognized that depending on the example, certain acts or events of any of the techniques described herein can be performed in a different sequence, may be added, merged, or left out altogether (e.g., not all described acts or events are necessary for the practice of the techniques). Moreover, in certain examples, acts or events may be performed concurrently, e.g., through multi-threaded processing, interrupt processing, or multiple processors, rather than sequentially.

[0218] In one or more examples, the functions described may be implemented in hardware, software, firmware, or any combination thereof. If implemented in software, the functions may be stored on or transmitted over as one or more instructions or code on a computer-readable medium and executed by a hardware-based processing unit. Computer-readable media may include computer-readable storage media, which corresponds to a tangible medium such as data storage media, or communication media including any medium that facilitates transfer of a computer program from one place to another, e.g., according to a communication protocol. In this manner, computer-readable media generally may correspond to (1) tangible computer-readable storage media which is non-transitory or (2) a communication medium such as a signal or carrier wave. Data storage media may be any available media that can be accessed by

one or more computers or one or more processors to retrieve instructions, code and/or data structures for implementation of the techniques described in this disclosure. A computer program product may include a computer-readable medium.

[0219] By way of example, and not limitation, such computer-readable storage media can comprise RAM, ROM, EEPROM, CD-ROM or other optical disk storage, magnetic disk storage, or other magnetic storage devices, flash memory, or any other medium that can be used to store desired program code in the form of instructions or data structures and that can be accessed by a computer. Also, any connection is properly termed a computer-readable medium. For example, if instructions are transmitted from a website, server, or other remote source using a coaxial cable, fiber optic cable, twisted pair, digital subscriber line (DSL), or wireless technologies such as infrared, radio, and microwave, then the coaxial cable, fiber optic cable, twisted pair, DSL, or wireless technologies such as infrared, radio, and microwave are included in the definition of medium. It should be understood, however, that computer-readable storage media and data storage media do not include connections, carrier waves, signals, or other transitory media, but are instead directed to non-transitory, tangible storage media. Disk and disc, as used herein, includes compact disc (CD), laser disc, optical disc, digital versatile disc (DVD), floppy disk and Blu-ray disc, where disks usually reproduce data magnetically, while discs reproduce data optically with lasers. Combinations of the above should also be included within the scope of computer-readable media.

[0220] Instructions may be executed by one or more processors, such as one or more digital signal processors (DSPs), general purpose microprocessors, application specific integrated circuits (ASICs), field programmable gate arrays (FPGAs), or other equivalent integrated or discrete logic circuitry. Accordingly, the terms “processor” and “processing circuitry,” as used herein may refer to any of the foregoing structures or any other structure suitable for implementation of the techniques described herein. In addition, in some aspects, the functionality described herein may be provided within dedicated hardware and/or software modules configured for encoding and decoding, or incorporated in a combined codec. Also, the techniques could be fully implemented in one or more circuits or logic elements.

[0221] The techniques of this disclosure may be implemented in a wide variety of devices or apparatuses, including a wireless handset, an integrated circuit (IC) or a set of ICs (e.g., a chip set). Various components, modules, or units are described in this disclosure to emphasize functional aspects of devices configured to perform the

disclosed techniques, but do not necessarily require realization by different hardware units. Rather, as described above, various units may be combined in a codec hardware unit or provided by a collection of interoperative hardware units, including one or more processors as described above, in conjunction with suitable software and/or firmware.

[0222] Various examples have been described. These and other examples are within the scope of the following claims.

CLAIMS

1. A method of coding video data, the method comprising:
 - responsive to determining to predict a current block of a current picture of video data using intra block copy (IBC), generating a motion vector (MV) candidate list for the current block,
 - wherein the current block is a first block of the current picture; and
 - wherein generating the MV candidate list comprises:
 - determining one or more history-based motion vector prediction (HMVP) candidates; and
 - including, as a last candidate in the MV candidate list, an HMVP candidate of the one or more HMVP candidates;
 - selecting, from the MV candidate list, a particular MV candidate that identifies a predictor block in the current picture;
 - reconstructing pixels of the current block based on pixels of the predictor block;
 - responsive to determining to predict a second block of the current picture of video data using inter prediction, generating a MV candidate list for the second block, wherein generating the MV candidate list for the second block comprises:
 - determining a maximum number of candidates to be included in the MV candidate list for the second block; and
 - responsive to determining that a quantity of candidates in the MV candidate list for the second block is equal to the maximum number of candidates to be included in the MV candidate list for the second block minus one, refraining from including additional HMVP candidates for the second block in the MV candidate list for the second block such that the last candidate in the MV candidate list for the second block is not an HMVP candidate;
 - selecting, from the MV candidate list for the second block, a particular MV candidate that identifies a predictor block for the second block in a picture of the video data that is not the same as the current picture; and
 - reconstructing pixels of the second block based on pixels of the predictor block for the second block.

2. The method of claim 1, wherein generating the MV candidate list for the first block comprises not evaluating any pairwise candidates for inclusion in the MV candidate list for the first block.
3. The method of claim 2, wherein generating the MV candidate list for the second block comprises evaluating one or more pairwise candidates for inclusion in the MV candidate list for the second block.
4. The method of claim 3, wherein generating the MV candidate list for the first block further comprises:
 - evaluating an above-left neighboring B2 candidate for inclusion in the MV candidate list for the first block where a current number of candidates included in the MV candidate list for the first block is greater than or equal to four.
5. The method of claim 3, wherein the MV candidate list for the first block comprises an IBC merge/skip list.
6. A device for coding video data, the device comprising:
 - a memory configured to store the video data; and
 - one or more processors that are implemented in circuitry and configured to:
 - generate, responsive to determining to predict a current block of a current picture of video data using intra block copy (IBC), a motion vector (MV) candidate list for the current block,
 - wherein the current block is a first block of the current picture;
 - and
 - wherein, to generate the MV candidate list, the one or more processors are configured to:
 - determine one or more history-based motion vector prediction (HMVP) candidates; and
 - include, as a last candidate in the MV candidate list, an HMVP candidate of the one or more HMVP candidates;
 - select, from the MV candidate list, a particular MV candidate that identifies a predictor block in the current picture;

reconstruct pixels of the current block based on pixels of the predictor block;

generate, responsive to determining to predict a second block of the current picture of video data using inter prediction, a MV candidate list for the second block, wherein, to generate the MV candidate list for the second block, the one or more processors are configured to:

determine a maximum number of candidates to be included in the MV candidate list for the second block; and

refrain, responsive to determining that a quantity of candidates in the MV candidate list for the second block is equal to the maximum number of candidates to be included in the MV candidate list for the second block minus one, from including additional HMVP candidates for the second block in the MV candidate list for the second block such that the last candidate in the MV candidate list for the second block is not an HMVP candidate;

select, from the MV candidate list for the second block, a particular MV candidate that identifies a predictor block for the second block in a picture of the video data that is not the same as the current picture; and

reconstruct pixels of the second block based on pixels of the predictor block for the second block.

7. The device of claim 6, wherein, to generate the MV candidate list for the first block, the one or more processors are configured to not evaluate any pairwise candidates for inclusion in the MV candidate list for the first block.

8. The device of claim 7, wherein, to generate the MV candidate list for the second block, the one or more processors are configured to evaluate one or more pairwise candidates for inclusion in the MV candidate list for the second block.

9. The device of claim 8, wherein, to generate the MV candidate list for the first block, the one or more processors are configured to:

evaluate a B2 candidate for inclusion in the MV candidate list for the first block where a current number of candidates included in the MV candidate list for the first block is greater than or equal to four.

10. The device of claim 8, wherein the MV candidate list for the first block comprises an IBC merge/skip list.
11. The device of claim 6, wherein the video coder comprises a video decoder.
12. The device of claim 11, further comprising a display configured to display decoded video data.
13. The device of claim 11, wherein the device comprises one or more of a camera, a computer, a mobile device, a broadcast receiver device, or a set-top box.
14. The device of claim 6, wherein the device comprises a video encoder.
15. A computer-readable storage medium storing instructions that, when executed, cause one or more processors of a video coder to:
 - generate, responsive to determining to predict a current block of a current picture of video data using intra block copy (IBC), a motion vector (MV) candidate list for the current block,
 - wherein the current block is a first block of the current picture; and
 - wherein the instructions that cause the one or more processors to generate the MV candidate list comprise instructions that cause the one or more processors to:
 - determine one or more history-based motion vector prediction (HMVP) candidates; and
 - include, as a last candidate in the MV candidate list, an HMVP candidate of the one or more HMVP candidates;
 - select, from the MV candidate list, a particular MV candidate that identifies a predictor block in the current picture;
 - reconstruct pixels of the current block based on pixels of the predictor block;
 - generate, responsive to determining to predict a second block of the current picture of video data using inter prediction, a MV candidate list for the second block, wherein the instructions that cause the one or more processors to generate the MV candidate list for the second block comprise instructions that cause the one or more processors to:

determine a maximum number of candidates to be included in the MV candidate list for the second block; and

refrain, responsive to determining that a quantity of candidates in the MV candidate list for the second block is equal to the maximum number of candidates to be included in the MV candidate list for the second block minus one, from including additional HMVP candidates for the second block in the MV candidate list for the second block such that the last candidate in the MV candidate list for the second block is not an HMVP candidate;

select, from the MV candidate list for the second block, a particular MV candidate that identifies a predictor block for the second block in a picture of the video data that is not the same as the current picture; and

reconstruct pixels of the second block based on pixels of the predictor block for the second block.

16. The computer-readable storage medium of claim 15, wherein the instructions that cause the one or more processors to generate the MV candidate list for the first block comprise instructions that cause the one or more processors to not evaluate any pairwise candidates for inclusion in the MV candidate list for the first block.

17. The computer-readable storage medium of claim 16, wherein the instructions that cause the one or more processors to generate the MV candidate list for the second block comprise instructions that cause the one or more processors to evaluate one or more pairwise candidates for inclusion in the MV candidate list for the second block.