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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
15 December 2016 (15.12.2016)

WIPO | PCT

(10) International Publication Number
WO 2016/201094 A1

(51) International Patent Classification:
*H04N 19/52* (2014.01) *H04N 19/61* (2014.01)
*H04N 19/70* (2014.01)

(21) International Application Number:
PCT/US2016/036682

(22) International Filing Date:
9 June 2016 (09.06.2016)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

| | | |
|---|---|---|
| 62/174,393 | 11 June 2015 (11.06.2015) | US |
| 62/295,329 | 15 February 2016 (15.02.2016) | US |
| 15/176,790 | 8 June 2016 (08.06.2016) | US |

(71) Applicant: **QUALCOMM INCORPORATED** [US/US]; International IP Administration, 5775 Morehouse Drive, San Diego, California 92121-1714 (US).

(72) Inventors: **CHIEN, Wei-Jung**; 5775 Morehouse Drive, San Diego, California 92121-1714 (US). **WANG, Xianglin**; 5775 Morehouse Drive, San Diego, California 92121-1714 (US). **ZHANG, Li**; 5775 Morehouse Drive, San Diego, California 92121-1714 (US). **LIU, Hongbin**; Room 9-202, 37th Building, Longhuayuan, 2nd Section, Huilongguan, Changping District, Beijing 102208 (CN). **CHEN, Jianle**; 5775 Morehouse Drive, San Diego, California 92121-1714 (US). **KARCZEWICZ, Marta**; 5775 Morehouse Drive, San Diego, California 92121-1714 (US).

(74) Agent: **DAWLEY, Brian R.**; Shumaker & Sieffert, P.A., 1625 Radio Drive, Suite 300, Woodbury, Minnesota 55125 (US).

*[Continued on next page]*

(54) Title: SUB-PREDICTION UNIT MOTION VECTOR PREDICTION USING SPATIAL AND/OR TEMPORAL MOTION INFORMATION

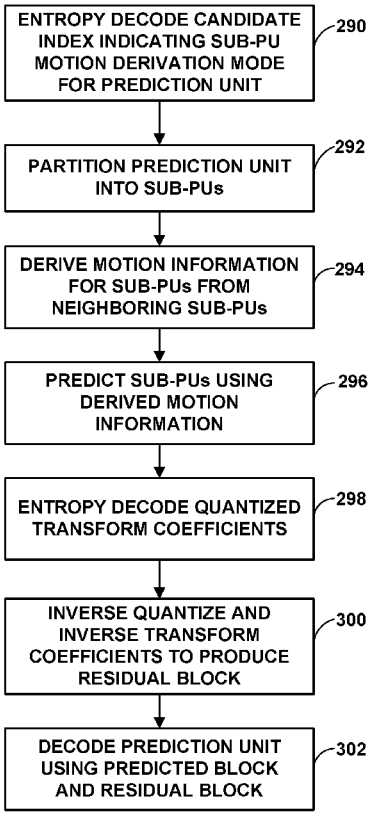


FIG. 13

(57) Abstract: In one example, a device for decoding video data includes a memory configured to store video data and a video decoder configured to determine that a motion prediction candidate for a current block of video data indicates that motion information is to be derived for sub-blocks of the current block, in response to the determination: partition the current block into the sub-blocks, for each of the sub-blocks, derive motion information using motion information for at least two neighboring blocks, and decode the sub-blocks using the respective derived motion information.

WO 2016/201094 A1

(81) **Designated States** *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) **Designated States** *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

**Published**:

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

# SUB-PREDICTION UNIT MOTION VECTOR PREDICTION USING SPATIAL AND/OR TEMPORAL MOTION INFORMATION

**[0001]** This application claims the benefit of U.S. Provisional Application No. **62/174,393**, filed June 11, 2015, and U.S. Provisional Application No. **62/295,329**, filed February 15, 2016, the entire contents of each of which are hereby incorporated by reference.

## TECHNICAL FIELD

**[0002]** This disclosure relates to video coding.

## 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 (also referred to as ITU-T H.265), 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 frame or a portion of a video frame) may be partitioned into video blocks, which for some techniques may also be referred to as treeblocks, 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 a reference frames.

**[0005]** Spatial or temporal prediction results in a predictive block for a block to be coded. Residual data represents pixel differences between the original block to be coded and the predictive block. An inter-coded block is encoded according to a motion vector that points to a block of reference samples forming the predictive block, and the residual data indicating the difference between the coded block and the predictive block. An intra-coded block is encoded according to an intra-coding mode and the residual data. For further compression, the residual data may be transformed from the pixel domain to a transform domain, resulting in residual transform coefficients, which then may be quantized. The quantized transform coefficients, initially arranged in a two-dimensional array, may be scanned in order to produce a one-dimensional vector of transform coefficients, and entropy coding may be applied to achieve even more compression.

## SUMMARY

**[0006]** In general, the techniques of this disclosure relate to derivation of motion information (e.g., motion vectors) for sub-blocks of blocks of video data. For example, the techniques may be used to derive motion information for prediction units (PUs) or sub-prediction units (sub-PUs) of PUs. In general, these techniques include deriving the motion information for each of the sub-blocks from motion information of neighboring sub-blocks. The neighboring sub-blocks may include spatially and/or temporally neighboring sub-blocks. For example, for a given sub-block, a video coder (such as a video encoder or a video decoder) may derive motion information by combining (e.g., averaging) motion information of a left-neighboring sub-block, an above-neighboring sub-block, and/or a temporally neighboring sub-block, such as a bottom-right temporally neighboring sub-block. In addition, derivation of such motion information for sub-blocks may be signaled using a particular candidate of a candidate list for motion information prediction.

**[0007]** In one example, a method of decoding video data includes determining that a motion prediction candidate for a current block of video data indicates that motion information is to be derived for sub-blocks of the current block, and in response to the determination: partitioning the current block into the sub-blocks, for each of the sub-blocks, deriving motion information using motion information for at least two

neighboring blocks, and decoding the sub-blocks using the respective derived motion information.

**[0008]** In another example, a device for decoding video data includes a memory configured to store video data and a video decoder configured to determine that a motion prediction candidate for a current block of video data indicates that motion information is to be derived for sub-blocks of the current block, and in response to the determination: partition the current block into the sub-blocks, for each of the sub-blocks, derive motion information using motion information for at least two neighboring blocks, and decode the sub-blocks using the respective derived motion information.

**[0009]** In another example, a device for decoding video data includes means for determining that a motion prediction candidate for a current block of video data indicates that motion information is to be derived for sub-blocks of the current block, means for partitioning the current block into the sub-blocks in response to the determination, means for deriving, for each of the sub-blocks, motion information using motion information for at least two neighboring blocks in response to the determination, and means for decoding the sub-blocks using the respective derived motion information in response to the determination.

**[0010]** In another example, a computer-readable storage medium has stored thereon instructions that, when executed, cause a processor to determine that a motion prediction candidate for a current block of video data indicates that motion information is to be derived for sub-blocks of the current block, and in response to the determination: partition the current block into the sub-blocks, for each of the sub-blocks, derive motion information using motion information for at least two neighboring blocks, and decode the sub-blocks using the respective derived motion information.

**[0011]** The details of one or more examples are set forth in the accompanying drawings and the description below. Other features, objects, and advantages will be apparent from the description and drawings, and from the claims.

## BRIEF DESCRIPTION OF DRAWINGS

**[0012]** FIG. 1 is a block diagram illustrating an example video encoding and decoding system that may utilize techniques for implementing advanced temporal motion vector prediction (ATMVP).

**[0013]** FIG. 2 is a block diagram illustrating an example of video encoder that may implement techniques for advanced temporal motion vector prediction (ATMVP).

**[0014]** FIG. 3 is a block diagram illustrating an example of video decoder that may implement techniques for advanced temporal motion vector prediction (ATMVP).

**[0015]** FIG. 4 is a conceptual diagram illustrating spatial neighboring candidates in High Efficiency Video Coding (HEVC).

**[0016]** FIG. 5 is a conceptual diagram illustrating temporal motion vector prediction (TMVP) in HEVC.

**[0017]** FIG. 6 illustrates an example prediction structure for 3D-HEVC.

**[0018]** FIG. 7 is a conceptual diagram illustrating sub-PU based inter-view motion prediction in 3D-HEVC.

**[0019]** FIG. 8 is a conceptual diagram illustrating sub-PU motion prediction from a reference picture.

**[0020]** FIG. 9 is a conceptual diagram illustrating relevant pictures in ATMVP (similar to TMVP).

**[0021]** FIG. 10 is a flowchart illustrating an example spatial-temporal motion vector predictor (STMVP) derivation process.

**[0022]** FIGS. 11A and 11B are conceptual diagrams illustrating examples of sub-PUs of a PU, as well as neighboring sub-PUs to the PU.

**[0023]** FIG. 12 is a flowchart illustrating an example method of encoding video data in accordance with the techniques of this disclosure.

**[0024]** FIG. 13 is an example of a method of decoding video data in accordance with the techniques of this disclosure.

## DETAILED DESCRIPTION

**[0025]** In general, this disclosure is related to motion vector prediction in video codecs. More specifically, advanced motion vector prediction may be achieved by deriving motion vectors for sub-blocks (e.g., sub-prediction units (PUs)) for a given block (e.g., a prediction unit (PU)) from spatial and temporal neighboring blocks. In one example, a video coder (such as a video encoder or a video decoder) may partition a current block (e.g., a current PU) into sub-blocks (e.g., sub-PUs), and for each sub-PU, derive motion information, including a motion vector, for each of the sub-PUs from neighboring blocks, which may include spatially and/or temporally neighboring blocks. For

example, for each of the sub-blocks, the video coder may derive motion information from a left-neighboring spatial block, an above-neighboring spatial block, and/or a bottom-right neighboring temporal block. The spatially neighboring blocks may be sub-blocks directly adjacent to the sub-block, or that are outside of the current block including the sub-block. Using sub-blocks outside of the current block may allow the motion information for the sub-blocks to be derived in parallel.

**[0026]** 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 Multiview Video Coding (MVC) extensions. The latest joint draft of MVC is described in "Advanced video coding for generic audiovisual services," ITU-T Recommendation H.264, Mar. 2010.

**[0027]** In addition, there is a newly developed video coding standard, namely High Efficiency Video Coding (HEVC), developed by the Joint Collaboration Team on Video Coding (JCT-VC) of ITU-T Video Coding Experts Group (VCEG) and ISO/IEC Motion Picture Experts Group (MPEG). A recent draft of HEVC is available from phenix.int-evry.fr/jct/doc_end_user/documents/12_Geneva/wg11/JCTVC-L1003-v34.zip. The HEVC standard is also presented jointly in Recommendation ITU-T H.265 and International Standard ISO/IEC 23008-2, both entitled "High efficiency video coding," and both published October, 2014.

**[0028]** Motion information: For each block, a set of motion information can be available. A set of motion information contains motion information for forward and backward prediction directions. Here forward and backward prediction directions are two prediction directions corresponding to reference picture list 0 (RefPicList0) and reference picture list 1 (RefPicList1) of a current picture or slice. The terms "forward" and "backward" do not necessarily have a geometry meaning. Instead, they are used to distinguish which reference picture list a motion vector is based on. Forward prediction means the prediction formed based on reference list 0, while backward prediction means the prediction formed based on reference list 1. In case both reference list 0 and reference list 1 are used to form a prediction for a given block, it is called bi-directional prediction.

**[0029]** For a given picture or slice, if only one reference picture list is used, every block inside the picture or slice is forward predicted. If both reference picture lists are used

for a given picture or slice, a block inside the picture or slice may be forward predicted, or backward predicted, or bi-directionally predicted.

**[0030]** For each prediction direction, the motion information contains a reference index and a motion vector. A reference index is used to identify a reference picture in the corresponding reference picture list (e.g., RefPicList0 or RefPicList1). A motion vector has both a horizontal and a vertical component, with each indicating an offset value along horizontal and vertical direction respectively. In some descriptions, for simplicity, the term "motion vector" may be used interchangeably with motion information, to indicate both the motion vector and its associated reference index.

**[0031]** Picture order count (POC) is widely used in video coding standards to identify a display order of a picture. Although there are cases in which two pictures within one coded video sequence may have the same POC value, it typically does not happen within a coded video sequence. When multiple coded video sequences are present in a bitstream, pictures with a same value of POC may be closer to each other in terms of decoding order. POC values of pictures are typically used for reference picture list construction, derivation of reference picture set as in HEVC and motion vector scaling.

**[0032]** Macroblock (MB) structure in Advanced Video Coding (AVC) (H.264): In H.264/AVC, each inter macroblock (MB) may be partitioned into four different ways:

- One 16x16 MB partition
- Two 16x8 MB partitions
- Two 8x16 MB partitions
- Four 8x8 MB partitions

**[0033]** Different MB partitions in one MB may have different reference index values for each direction (RefPicList0 or RefPicList1).

**[0034]** When an MB is not partitioned into four 8x8 MB partitions, it has only one motion vector for each MB partition in each direction.

**[0035]** When an MB is partitioned into four 8x8 MB partitions, each 8x8 MB partition can be further partitioned into sub-blocks, each of which can have a different motion vector in each direction. There are four different ways to get sub-blocks from an 8x8 MB partition:

- One 8x8 sub-block
- Two 8x4 sub-blocks
- Two 4x8 sub-blocks
- Four 4x4 sub-blocks

**[0036]** Each sub-block can have a different motion vector in each direction. Therefore, a motion vector is present in a level equal to or higher than the sub-block.

**[0037]** Temporal direct mode in AVC: In AVC, temporal direct mode could be enabled in either MB or MB partition level for skip or direct mode in B slices. For each MB partition, the motion vectors of the block co-located with the current MB partition in the RefPicList1[ 0 ] of the current block are used to derive the motion vectors. Each motion vector in the co-located block is scaled based on POC distances.

**[0038]** Spatial direct mode in AVC: In AVC, a direct mode can also predict motion information from the spatial neighbors.

**[0039]** Coding Unit (CU) Structure in High Efficiency Video Coding (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.

**[0040]** The size of a CTB can range from 16x16 to 64x64 in the HEVC main profile (although technically 8x8 CTB sizes can be supported). A coding unit (CU) could be the same size of a CTB and as small as 8x8. Each coding unit is coded with one mode. When a CU is inter coded, it 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 ¼ or ¾ size of the CU.

**[0041]** When the CU is inter coded, one set of motion information is present for each PU. In addition, each PU is coded with a unique inter-prediction mode to derive the set of motion information.

**[0042]** Motion prediction in HEVC: In the HEVC standard, there are two motion vector 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).

**[0043]** 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.

**[0044]** The MV candidate list contains up to 5 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.

**[0045]** 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.

**[0046]** The candidates for both modes are derived similarly from the same spatial and temporal neighboring blocks.

**[0047]** FIG. 1 is a block diagram illustrating an example video encoding and decoding system 10 that may utilize techniques for implementing advanced temporal motion vector prediction (ATMVP). As shown in FIG. 1, system 10 includes a source device 12 that provides encoded video data to be decoded at a later time by a destination device 14. In particular, source device 12 provides the video data to destination device 14 via a computer-readable medium 16. Source device 12 and destination device 14 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 so-called "smart" phones, so-called "smart" pads, televisions, cameras, display devices, digital media players, video gaming consoles, video streaming device, or the like. In some cases, source device 12 and destination device 14 may be equipped for wireless communication.

**[0048]** Destination device 14 may receive the encoded video data to be decoded via computer-readable medium 16. Computer-readable medium 16 may comprise any type of medium or device capable of moving the encoded video data from source device 12 to destination device 14. In one example, computer-readable medium 16 may comprise a communication medium to enable source device 12 to transmit encoded video data directly to destination device 14 in real-time. The encoded video data may be modulated according to a communication standard, such as a wireless communication protocol, and transmitted to destination device 14. 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 12 to destination device 14.

**[0049]** In some examples, encoded data may be output from output interface 22 to a storage device. Similarly, encoded data may be accessed from the storage device by input interface. The storage device 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. In a further example, the storage device may correspond to a file server or another intermediate storage device that may store the encoded video generated by source device 12. Destination device 14 may access stored video data from the storage device via streaming or download. The file server may be any type of server capable of storing encoded video data and transmitting that encoded video data to the destination device 14. Example file servers include a web server (e.g., for a website), an FTP server, network attached storage (NAS) devices, or a local disk drive. Destination device 14 may access the encoded video data 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 a file server. The transmission of encoded video data from the storage device may be a streaming transmission, a download transmission, or a combination thereof.

**[0050]** The techniques of this disclosure are not necessarily limited to wireless applications or settings. The techniques 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. In some examples, system 10 may be configured to support one-way or two-way video transmission to support applications such as video streaming, video playback, video broadcasting, and/or video telephony.

**[0051]** In the example of FIG. 1, source device 12 includes video source 18, video encoder 20, and output interface 22. Destination device 14 includes input interface 28, video decoder 30, and display device 32. In accordance with this disclosure, video encoder 20 of source device 12 may be configured to apply the techniques for advanced

temporal motion vector prediction (ATMVP). In other examples, a source device and a destination device may include other components or arrangements. For example, source device 12 may receive video data from an external video source 18, such as an external camera. Likewise, destination device 14 may interface with an external display device, rather than including an integrated display device.

**[0052]** The illustrated system 10 of FIG. 1 is merely one example. Techniques for advanced temporal motion vector prediction (ATMVP) may be performed by any digital video encoding and/or decoding device. Although generally the techniques of this disclosure are performed by a video encoding device, the techniques may also be performed by a video encoder/decoder, typically referred to as a "CODEC." Moreover, the techniques of this disclosure may also be performed by a video preprocessor. Source device 12 and destination device 14 are merely examples of such coding devices in which source device 12 generates coded video data for transmission to destination device 14. In some examples, devices 12, 14 may operate in a substantially symmetrical manner such that each of devices 12, 14 include video encoding and decoding components. Hence, system 10 may support one-way or two-way video transmission between video devices 12, 14, e.g., for video streaming, video playback, video broadcasting, or video telephony.

**[0053]** Video source 18 of source device 12 may include a video capture device, such as a video camera, a video archive containing previously captured video, and/or a video feed interface to receive video from a video content provider. As a further alternative, video source 18 may generate computer graphics-based data as the source video, or a combination of live video, archived video, and computer-generated video. In some cases, if video source 18 is a video camera, source device 12 and destination device 14 may form so-called camera phones or video phones. As mentioned above, however, the techniques described in this disclosure may be applicable to video coding in general, and may be applied to wireless and/or wired applications. In each case, the captured, pre-captured, or computer-generated video may be encoded by video encoder 20. The encoded video information may then be output by output interface 22 onto a computer-readable medium 16.

**[0054]** Computer-readable medium 16 may include transient media, such as a wireless broadcast or wired network transmission, or storage media (that is, non-transitory storage media), such as a hard disk, flash drive, compact disc, digital video disc, Blu-ray disc, or other computer-readable media. In some examples, a network server (not

shown) may receive encoded video data from source device 12 and provide the encoded video data to destination device 14, e.g., via network transmission. Similarly, a computing device of a medium production facility, such as a disc stamping facility, may receive encoded video data from source device 12 and produce a disc containing the encoded video data. Therefore, computer-readable medium 16 may be understood to include one or more computer-readable media of various forms, in various examples.

**[0055]** Input interface 28 of destination device 14 receives information from computer-readable medium 16. The information of computer-readable medium 16 may include syntax information defined by video encoder 20, which is also used by video decoder 30, that includes syntax elements that describe characteristics and/or processing of blocks and other coded units, e.g., GOPs. Display device 32 displays the decoded video data to a user, and may comprise 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.

**[0056]** Video encoder 20 and video decoder 30 may operate according to a video coding standard, such as the High Efficiency Video Coding (HEVC) standard, extensions to the HEVC standard, or subsequent standards, such as ITU-T H.266. Alternatively, video encoder 20 and video decoder 30 may operate according to other proprietary or industry standards, such as the ITU-T H.264 standard, alternatively referred to as MPEG-4, Part 10, Advanced Video Coding (AVC), or extensions of such standards. The techniques of this disclosure, however, are not limited to any particular coding standard. Other examples of video coding standards include MPEG-2 and ITU-T H.263. Although not shown in FIG. 1, in some aspects, video encoder 20 and video decoder 30 may each be integrated with an audio encoder and decoder, and may include appropriate MUX-DEMUX units, or other hardware and software, to handle encoding of both audio and video in a common data stream or separate data streams. If applicable, MUX-DEMUX units may conform to the ITU H.223 multiplexer protocol, or other protocols such as the user datagram protocol (UDP).

**[0057]** Video encoder 20 and video decoder 30 each may be implemented as any of a variety of suitable encoder 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 20 and video decoder 30 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.

**[0058]** In general, the HEVC standard describes that a video frame or picture may be divided into a sequence of treeblocks or largest coding units (LCU) that include both luma and chroma samples. Syntax data within a bitstream may define a size for the LCU, which is a largest coding unit in terms of the number of pixels. A slice includes a number of consecutive treeblocks in coding order. A video frame or picture may be partitioned into one or more slices. Each treeblock may be split into coding units (CUs) according to a quadtree. In general, a quadtree data structure includes one node per CU, with a root node corresponding to the treeblock. If a CU is split into four sub-CUs, the node corresponding to the CU includes four leaf nodes, each of which corresponds to one of the sub-CUs.

**[0059]** Each node of the quadtree data structure may provide syntax data for the corresponding CU. For example, a node in the quadtree may include a split flag, indicating whether the CU corresponding to the node is split into sub-CUs. Syntax elements for a CU may be defined recursively, and may depend on whether the CU is split into sub-CUs. If a CU is not split further, it is referred as a leaf-CU. In this disclosure, four sub-CUs of a leaf-CU will also be referred to as leaf-CUs even if there is no explicit splitting of the original leaf-CU. For example, if a CU at 16x16 size is not split further, the four 8x8 sub-CUs will also be referred to as leaf-CUs although the 16x16 CU was never split.

**[0060]** A CU in H.265 has a similar purpose as a macroblock of the H.264 standard, except that a CU does not have a size distinction. For example, a treeblock may be split into four child nodes (also referred to as sub-CUs), and each child node may in turn be a parent node and be split into another four child nodes. A final, unsplit child node, referred to as a leaf node of the quadtree, comprises a coding node, also referred to as a leaf-CU. Syntax data associated with a coded bitstream may define a maximum number of times a treeblock may be split, referred to as a maximum CU depth, and may also define a minimum size of the coding nodes. Accordingly, a bitstream may also define a smallest coding unit (SCU). This disclosure uses the term "block" to refer to any of a CU, PU, or TU, in the context of HEVC, or similar data structures in the context of other standards (e.g., macroblocks and sub-blocks thereof in H.264/AVC).

**[0061]** A CU includes a coding node and prediction units (PUs) and transform units (TUs) associated with the coding node. A size of the CU corresponds to a size of the coding node and must be square in shape. The size of the CU may range from 8x8 pixels up to the size of the treeblock with a maximum of 64x64 pixels or greater. Each CU may contain one or more PUs and one or more TUs. Syntax data associated with a CU may describe, for example, partitioning of the CU into one or more PUs. Partitioning modes may differ between whether the CU is skip or direct mode encoded, intra-prediction mode encoded, or inter-prediction mode encoded. PUs may be partitioned to be non-square in shape. Syntax data associated with a CU may also describe, for example, partitioning of the CU into one or more TUs according to a quadtree. A TU can be square or non-square (e.g., rectangular) in shape.

**[0062]** The HEVC standard allows for transformations according to TUs, which may be different for different CUs. The TUs are typically sized based on the size of PUs within a given CU defined for a partitioned LCU, although this may not always be the case. The TUs are typically the same size or smaller than the PUs. In some examples, residual samples corresponding to a CU may be subdivided into smaller units using a quadtree structure known as "residual quad tree" (RQT). The leaf nodes of the RQT may be referred to as transform units (TUs). Pixel difference values associated with the TUs may be transformed to produce transform coefficients, which may be quantized.

**[0063]** A leaf-CU may include one or more prediction units (PUs). In general, a PU represents a spatial area corresponding to all or a portion of the corresponding CU, and may include data for retrieving a reference sample for the PU. Moreover, a PU includes data related to prediction. For example, when the PU is intra-mode encoded, data for the PU may be included in a residual quadtree (RQT), which may include data describing an intra-prediction mode for a TU corresponding to the PU. As another example, when the PU is inter-mode encoded, the PU may include data defining one or more motion vectors for the PU. The data defining the motion vector for a PU may describe, for example, a horizontal component of the motion vector, a vertical component of the motion vector, a resolution for the motion vector (e.g., one-quarter pixel precision or one-eighth pixel precision), a reference picture to which the motion vector points, and/or a reference picture list (e.g., List 0, List 1, or List C) for the motion vector.

**[0064]** A leaf-CU having one or more PUs may also include one or more transform units (TUs). The transform units may be specified using an RQT (also referred to as a

TU quadtree structure), as discussed above. For example, a split flag may indicate whether a leaf-CU is split into four transform units. Then, each transform unit may be split further into further sub-TUs. When a TU is not split further, it may be referred to as a leaf-TU. Generally, for intra coding, all the leaf-TUs belonging to a leaf-CU share the same intra prediction mode. That is, the same intra-prediction mode is generally applied to calculate predicted values for all TUs of a leaf-CU. For intra coding, a video encoder may calculate a residual value for each leaf-TU using the intra prediction mode, as a difference between the portion of the CU corresponding to the TU and the original block. A TU is not necessarily limited to the size of a PU. Thus, TUs may be larger or smaller than a PU. For intra coding, a PU may be collocated with a corresponding leaf-TU for the same CU. In some examples, the maximum size of a leaf-TU may correspond to the size of the corresponding leaf-CU.

**[0065]** Moreover, TUs of leaf-CUs may also be associated with respective quadtree data structures, referred to as residual quadtrees (RQTs). That is, a leaf-CU may include a quadtree indicating how the leaf-CU is partitioned into TUs. The root node of a TU quadtree generally corresponds to a leaf-CU, while the root node of a CU quadtree generally corresponds to a treeblock (or LCU). TUs of the RQT that are not split are referred to as leaf-TUs. In general, this disclosure uses the terms CU and TU to refer to leaf-CU and leaf-TU, respectively, unless noted otherwise.

**[0066]** A video sequence typically includes a series of video frames or pictures. A group of pictures (GOP) generally comprises a series of one or more of the video pictures. A GOP may include syntax data in a header of the GOP, a header of one or more of the pictures, or elsewhere, that describes a number of pictures included in the GOP. Each slice of a picture may include slice syntax data that describes an encoding mode for the respective slice. Video encoder 20 typically operates on video blocks within individual video slices in order to encode the video data. A video block may correspond to a coding node within a CU. The video blocks may have fixed or varying sizes, and may differ in size according to a specified coding standard.

**[0067]** As an example, the HM supports prediction in various PU sizes. Assuming that the size of a particular CU is 2Nx2N, the HM supports intra-prediction in PU sizes of 2Nx2N or NxN, and inter-prediction in symmetric PU sizes of 2Nx2N, 2NxN, Nx2N, or NxN. The HM also supports asymmetric partitioning for inter-prediction in PU sizes of 2NxnU, 2NxnD, nLx2N, and nRx2N. In asymmetric partitioning, one direction of a CU is not partitioned, while the other direction is partitioned into 25% and 75%. The

portion of the CU corresponding to the 25% partition is indicated by an "n" followed by an indication of "Up", "Down," "Left," or "Right." Thus, for example, "2NxnU" refers to a 2Nx2N CU that is partitioned horizontally with a 2Nx0.5N PU on top and a 2Nx1.5N PU on bottom.

**[0068]** In this disclosure, "NxN" and "N by N" may be used interchangeably to refer to the pixel dimensions of a video block in terms of vertical and horizontal dimensions, e.g., 16x16 pixels or 16 by 16 pixels. In general, a 16x16 block will have 16 pixels in a vertical direction (y = 16) and 16 pixels in a horizontal direction (x = 16). Likewise, an NxN block generally has N pixels in a vertical direction and N pixels in a horizontal direction, where N represents a nonnegative integer value. The pixels in a block may be arranged in rows and columns. Moreover, blocks need not necessarily have the same number of pixels in the horizontal direction as in the vertical direction. For example, blocks may comprise NxM pixels, where M is not necessarily equal to N.

**[0069]** Following intra-predictive or inter-predictive coding using the PUs of a CU, video encoder 20 may calculate residual data for the TUs of the CU. The PUs may comprise syntax data describing a method or mode of generating predictive pixel data in the spatial domain (also referred to as the pixel domain) and the TUs may comprise coefficients in the transform domain following application of a transform, e.g., a discrete cosine transform (DCT), an integer transform, a wavelet transform, or a conceptually similar transform to residual video data. The residual data may correspond to pixel differences between pixels of the unencoded picture and prediction values corresponding to the PUs. Video encoder 20 may form the TUs including the residual data for the CU, and then transform the TUs to produce transform coefficients for the CU.

**[0070]** Following any transforms to produce transform coefficients, video encoder 20 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. The quantization process may reduce the bit depth associated with some or all of the coefficients. For example, an *n*-bit value may be rounded down to an *m*-bit value during quantization, where *n* is greater than *m*.

**[0071]** Following quantization, the video encoder 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 array and to place lower energy (and therefore higher frequency) coefficients at the back of the array. In some examples, video encoder 20 may utilize a predefined scan order to scan the quantized transform coefficients to produce a serialized vector that can be entropy encoded. In other examples, video encoder 20 may perform an adaptive scan. After scanning the quantized transform coefficients to form a one-dimensional vector, video encoder 20 may entropy encode the one-dimensional vector, e.g., according to context-adaptive variable length coding (CAVLC), context-adaptive binary arithmetic coding (CABAC), syntax-based context-adaptive binary arithmetic coding (SBAC), Probability Interval Partitioning Entropy (PIPE) coding or another entropy encoding methodology. Video encoder 20 may also entropy encode syntax elements associated with the encoded video data for use by video decoder 30 in decoding the video data.

**[0072]** To perform CABAC, video encoder 20 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 non-zero or not. To perform CAVLC, video encoder 20 may select a variable length code for a symbol to be transmitted. Codewords in VLC may be constructed such that relatively shorter codes correspond to more probable symbols, while longer codes correspond to less probable symbols. In this way, the use of VLC may achieve a bit savings over, for example, using equal-length codewords for each symbol to be transmitted. The probability determination may be based on a context assigned to the symbol.

**[0073]** U.S. Application No. 15/005,564 (hereinafter, "the '564 Application"), filed January 25, 2016, described the following techniques, which may be performed by video encoder 20 and/or video decoder 30, alone or in any combination, in addition to the techniques of this disclosure. In particular, the '564 Application describes techniques related to the position of the ATMVP candidate, if inserted, e.g., as a merge candidate list. Assume the spatial candidates and TMVP candidate are inserted into a merge candidate list in a certain order. The ATMVP candidate may be inserted in any relatively fixed position of those candidates. In one alternative, for example, the ATMVP candidate can be inserted in the merge candidate list after the first two spatial candidates e.g., A1 and B1. In one alternative, for example, the ATMVP candidate can be inserted after the first three spatial candidates e.g., A1 and B1 and B0. In one alternative, for example, the ATMVP candidate can be inserted after the first four candidates e.g., A1, B1, B0, and A0. In one alternative, for example, the ATMVP

candidate can be inserted right before the TMVP candidate. In one alternatively, for example, the ATMVP candidate can be inserted right after the TMVP candidate. Alternatively, the position of ATMVP candidate in the candidate list can be signaled in the bitstream. The positions of other candidates, including the TMVP candidate can be additionally signaled.

**[0074]** The '564 Application also describes techniques related to an availability check of the ATMVP candidate can apply by accessing just one set of motion information, which video encoder 20 and/or video decoder 30 may be configured to perform. When such a set of information is unavailable, e.g., one block being intra-coded, the whole ATMVP candidate is considered as unavailable. In that case, the ATMVP will not be inserted into the merge list. A center position, or a center sub-PU is used purely to check the availability of the ATMVP candidate. When a center sub-PU is used, the center sub-PU is chosen to be the one that covers the center position (e.g., the center 3 position, with a relative coordinate of (W/2, H/2) to the top-left sample of the PU, wherein WxH is the size of the PU). Such a position or center sub-PU may be used together with the temporal vector to identify a corresponding block in the motion source picture. A set of motion information from the block that covers the center position of a corresponding block is identified.

**[0075]** The '564 Application also describes techniques Representative set of motion information for the ATMVP coded PU from a sub-PU, which video encoder 20 and/or video decoder 30 may be configured to perform. To form the ATMVP candidate the representative set of motion information is first formed. Such a representative set of motion information may be derived from a fixed position or fixed sub-PU. It can be chosen in the same way as that of the set of motion information used to determine the availability of the ATMVP candidate, as described above. When a sub-PU has identified its own set of motion information and it is being unavailable, it is set to be equal to the representative set of motion information. If the representative set of motion information is set to be that of a sub-PU, no additional motion storage is needed at the decoder side for the current CTU or slice in the worst case scenario. Such a representative set of motion information is used in all scenarios when the decoding processes requires the whole PU to be represented by one set of motion information, including pruning, such that the process is used to generate combined bi-predictive merging candidates.

**[0076]** The '564 Application also describes techniques related to how the ATMVP candidate may be pruned with TMVP candidate and interactions between TMVP and ATMVP can be considered, which may be performed by video encoder 20 and/or video decoder 30. The pruning of a sub-PU based candidate, e.g., ATMVP candidate with a normal candidate, may be conducted by using the representative set of motion information (as in bullet #3) for such a sub-PU based candidate. If such set of motion information is the same as a normal merge candidate, the two candidates are considered as the same. Alternatively, or in addition, a check is performed to determine whether the ATMVP contains multiple different sets of motion information for multiple sub-Pus; if at least two different sets are identified, the sub-PU based candidate is not used for pruning, i.e., is considered to be different to any other candidate; Otherwise, it may be used for pruning (e.g., may be pruned during the pruning process). Alternatively, or in addition, the ATMVP candidate may be pruned with the spatial candidates, e.g., the left and top ones only, with positions denoted as A1 and B1. Alternatively, only one candidate is formed from temporal reference, being either ATMVP candidate or TMVP candidate. When ATMVP is available, the candidate is ATMVP; otherwise, the candidate is TMVP. Such a candidate is inserted into the merge candidate list in a position similar to the position of TMVP. In this case, the maximum number of candidates may be kept as unchanged. Alternatively, TMVP is always disabled even when ATMVP is unavailable. Alternatively, TMVP is used only when ATMVP is unavailable. Alternatively, when ATMVP is available and TMVP is unavailable, one set of motion information of one sub-PU is used as the TMVP candidate. In this case, furthermore, the pruning process between ATMVP and TMVP is not applied. Alternatively, or additionally, the temporal vector used for ATMVP may be also used for TMVP, such that the bottom-right position or center 3 position as used for current TMVP in HEVC do not need to be used. Alternatively, the position identified by the temporal vector and the bottom-right and center 3 positions are jointly considered to provide an available TMVP candidate.

**[0077]** The '564 Application also describes how multiple availability checks for ATMVP can be supported to give higher chances for the ATMVP candidate to be more accurate and efficient, which may be performed by video encoder 20 and/or video decoder 30. When the current ATMVP candidate from the motion source picture as identified by the first temporal vector (e.g., as shown in FIG. 9) is unavailable, other pictures can be considered as motion source picture. When another picture is

considered, it may be associated with a different second temporal vector, or may be associated simply with a second temporal vector scaled from the first temporal vector that points to the unavailable ATMVP candidate. A second temporal vector can identify an ATMVP candidate in a second motion source picture and the same availability check can apply. If the ATMVP candidate as derived from the second motion source picture is available, the ATMVP candidate is derived and no other pictures need to be checked; otherwise, other pictures as motion source pictures need to be checked. Pictures to be checked may be those in the reference picture lists of the current picture, with a given order. For each list, the pictures are checked in the ascending order of the reference index. List X is first checked and pictures in list Y (being 1-X) follows. List X is chosen so that list X is the list that contains the collocated picture used for TMVP. Alternatively, X is simply set to be 1 or 0. Pictures to be checked may include those identified by motion vectors of the spatial neighbors, with a given order. A partition of the PU that the current ATMVP apply to may be 2Nx2N, NxN, 2NxN, Nx2N or other AMP partitions, such as 2NxN/2. Alternatively, or in addition, if other partition sizes can be allowed, ATMVP can be supported too, and such a size may include e.g., 64x8. Alternatively, the mode may be only applied to certain partitions, e.g., 2Nx2N.

**[0078]** The '564 Application also describes how the ATMVP candidate may be marked using a different type of merge mode, which video encoder 20 and/or video decoder 30 may be configured to perform.

**[0079]** When identifying a vector (temporal vector as in the first stage) from neighbors, multiple neighboring positions, e.g., those used in merge candidate list construction, can be checked in order. For each of the neighbors, the motion vectors corresponding to reference picture list 0 (list 0) or reference picture list 1 (list 1) can be checked in order. When two motion vectors are available, the motion vectors in list X can be checked first and followed by list Y (with Y being equal to 1-X), so that list X is the list that contains the collocated picture used for TMVP. In ATMVP, a temporal vector is used be added as a shift of any center position of a sub-PU, wherein the components of temporal vector may need to be shifted to integer numbers. Such a shifted center position is used to identify a smallest unit that motion vectors can be allocated to, e.g., with a size of 4x4 that covers the current center position. Alternatively, motion vectors corresponding to list 0 may be checked before those corresponding to list 1. Alternatively, motion vectors corresponding to list 1 may be checked before those corresponding to list 0. Alternatively, all motion vectors corresponding to list X in all spatial neighbors are

checked in order, followed by the motion vectors corresponding to list Y (with Y being equal to 1-X). Here X can be the one that indicates where collocated picture belongs to or just simply set to be 0 or 1. The order of the spatial neighbors can be the same as that used in HEVC merge mode.

**[0080]** The '564 Application also describes techniques relating to, when in the first stage of identifying a temporal vector does not include identifying a reference picture, the motion source picture as shown in FIG. 9, may be simply set to be a fixed picture, e.g., the collocated picture used for TMVP, which may be performed by video encoder 20 and/or video decoder 30. In such a case the vector may only be identified from the motion vectors that point to such a fixed picture. In such a case the vector may only be identified from the motion vectors that point to any picture but further scaled towards the fixed picture. When in the first stage of identifying a vector consists identifying a reference picture, the motion source picture as shown in FIG. 9, one or more of the following additional checks may apply for a candidate motion vector. If the motion vector is associated with a picture or slice that is Intra coded, such a motion vector is considered as unavailable and cannot be used to be converted to the vector. If the motion vector identifies an Intra block (by e.g., adding the current center coordinate with the motion vector) in the associated a picture, such a motion vector is considered as unavailable and cannot be used to be converted to the vector.

**[0081]** The '564 Application also describes techniques relating to, when in the first stage of identifying a vector, the components of the vector may be set to be (half width of the current PU, half height of the current PU), so that it identifies a bottom-right pixel position in the motion source picture, which may be performed by video encoder 20 and/or video decoder 30. Here (x, y) indicates horizontal and vertical components of one motion vector. Alternatively, the components of the vector may be set to be (sum(half width of the current PU, M), sum(half height of the current PU, N)) where the function sum(a, b) returns the sum of a and b. In one example, when the motion information is stored in 4x4 unit, M and N are both set to be equal to 2. In another example, when the motion information is stored in 8x8 unit, M and N are both set to be equal to 4.

**[0082]** The '564 Application also describes techniques related to the sub-block/sub-PU size when ATMVP applies being signaled in a parameter set, e.g., sequence parameter set of picture parameter set, which may be performed by video encoder 20 and/or video decoder 30. The size ranges from the least PU size to the CTU size. The size can be

also pre-defined or signaled. The size can be e.g., as small as 4x4. Alternatively, the sub-block/sub-PU size can be derived based on the size of PU or CU. For example, the sub-block/sub-PU can be set equal to max (4x4, (width of CU)>> M). The value of M can be pre-defined or signaled in the bitstream.

**[0083]** The '564 Application also describes techniques relating to the maximum number of merge candidates being increased by 1 due to the fact that ATMVP can be considered as a new merge candidate, which may be performed by video encoder 20 and/or video decoder 30. For example, compared to HEVC which takes up to 5 candidates in a merge candidate list after pruning, the maximum number of merge candidates can be increased to 6. Alternatively, pruning with conventional TMVP candidate or unification with the conventional TMVP candidate can be performed for ATMVP such that the maximum number of merge candidates can be kept as unchanged. Alternatively, when ATMVP is identified to be available, a spatial neighboring candidate is excluded from the merge candidate list, e.g. the last spatial neighboring candidate in fetching order is excluded.

**[0084]** The '564 Application also describes techniques relating to, when multiple spatial neighboring motion vectors are considered to derive the temporal vector, a motion vector similarity may be calculated based on the neighboring motion vectors of the current PU as well as the neighboring motion vectors identified by a specific temporal vector being set equal to a motion vector, which may be performed by video encoder 20 and/or video decoder 30. The one that leads to the highest motion similarity may be chosen as the final temporal vector. In one alternative, for each motion vector from a neighboring position N, it identifies a block (same size as the current PU) in the motion source picture, wherein its neighboring position N contains a set of the motion information. This set of motion vector is compared with the set of motion information as in the neighboring position N of the current block. In another alternative, for each motion vector from a neighboring position N, it identifies a block in the motion source picture, wherein its neighboring positions contain multiple sets of motion information. These multiple sets of motion vector are compared with the multiple sets of motion information from the neighboring positions of the current PU in the same relative positions.

**[0085]** A motion information similarity may be calculated according to the techniques above. For example, the current PU has the following sets of motion information from A1, B1, A0 and B0, denoted as MIA1, MIB1, MIA0 and MIB0. For a temporal vector

TV, it identifies a block corresponding to the PU in the motion source picture. Such a block has motion information from the same relative A1, B1, A0 and B0 positions, and denoted as TMIA1, TMIB1, TMIA0 and TMIB0. The motion similarity as determined by TV is calculated as MStv= $\sum_{N\in\{A1,B1,A0,B0\}} MVSim(MI_N, TMI_N)$, wherein MVSim() defines the similarity between two sets (MIN, TMIN) of motion information. In both of the above cases, the motion similarity MVSim can be used, wherein the two input parameters are the two motion information, each containing up to two motion vectors and two reference indices. Since each pair of the motion vectors in list X are actually associated with reference pictures in different list X of different pictures, the current picture and the motion source picture.

**[0086]** For each of the two motion vectors MVXN and TMVXN (with X being equal to 0 or 1), the motion vector difference MVDXN can be calculated as MVXN – TMVXN, according to the techniques above. Afterwards, the difference MVSimX is calculated as e.g., abs($MVDX_N[0]$)+ abs($MVDX_N[1]$), or ($MVDX_N[0]$*$MVDX_N[0]$+ $MVDX_N[1]$*$MVDX_N[1]$). If both sets of motion information contain available motion vectors, MVSim is set equal to MVSim0 + MVSim1. In order to have a unified calculation of the motion difference, both of the motion vectors need to be scaled towards the same fixed picture, which can be, e.g., the first reference picture RefPicListX[0] of the list X of the current picture. If the availability of the motion vector in list X from the first set and the availability of the motion vector in list X from the second set are different, i.e., one reference index is -1 while the other is not, such two sets of motion information are considered as not similar in direction X.

**[0087]** If the two sets are not similar in both sets, the final MVSim function may return a big value T, which may be, e.g., considered as infinite, according to the techniques above. Alternatively, for a pair of sets of motion information, if one is predicted from list X (X being equal to 0 or 1) but not list Y (Y being equal to 1-X) and the other has the same status, a weighting between 1 and 2 (e.g., MVSim is equal to MVSimX *1.5) may be used. When one set is only predicted from list X and the other is only predicted from list Y, MVSim is set to the big value T. Alternatively, for any set of motion information, as long as one motion vector is available, both motion vectors will be produced. In the case that only one motion vector is available (corresponding to list X), it is scaled to form the motion vector corresponding to the other list Y. Alternatively, the motion vector may be measured based on differences between the neighboring pixels of the current PU and the neighboring pixels of the block (same size as the

current PU) identified by the motion vector. The motion vector that leads to the smallest difference may be chosen as the final temporal vector.

**[0088]** The '564 Application also describes techniques relating to, when deriving the temporal vector of the current block, motion vectors and/or temporal vectors from neighboring blocks that are coded with ATMVP may have a higher priority than motion vectors from other neighboring blocks, which may be performed by video encoder 20 and/or video decoder 30. In one example, only temporal vectors of neighboring blocks are checked first, and the first available one can be set to the temporal vector of the current block. Only when such temporal vectors are not present, normal motion vectors are further checked. In this case, temporal vectors for ATMVP coded blocks need to be stored. In another example, only motion vectors from ATMVP coded neighboring blocks are checked first, and the first available one can be set to the temporal vector of the current block. Only when such temporal vectors are not present, normal motion vectors are further checked. In another example, only motion vectors from ATMVP coded neighboring blocks are checked first, and the first available one can be set to the temporal vector of the current block. If such motion vectors are not available, the checking of temporal vector continues similar to the manner discussed above. In another example, temporal vectors from neighboring blocks are checked first, the first available one can be set to the temporal vector of the current block. If such motion vectors are not available, the checking of temporal vector continues similar to the manner discussed above. In another example, temporal vectors and motion vectors of ATMVP coded neighboring blocks are checked first, the first available one can be set to the temporal vector of the current block. Only when such temporal vectors and motion vectors are not present, normal motion vectors are further checked.

**[0089]** The '564 Application also describes techniques relating to, when multiple spatial neighboring motion vectors are considered to derive the temporal vector, a motion vector may be chosen so that it minimizes the distortion that are calculated from pixel domain, e.g., template matching may be used to derive the temporal vector such that the one leads to minimal matching cost is selected as the final temporal vector. These techniques may also be performed by video encoder 20 and/or video decoder 30.

**[0090]** The '564 Application also describes techniques for derivation of a set of motion information from a corresponding block (in the motion source picture) being performed in a way that when a motion vector is available in the corresponding block for any list X (denote the motion vector to be MVX), for the current sub-PU of the ATMVP

candidate, the motion vector is considered as available for list X (by scaling the MVX), which may be performed by video encoder 20 and/or video decoder 30. If the motion vector is unavailable in the corresponding block for any list X, the motion vector is considered as unavailable for list X. Alternatively, when motion vector in the corresponding block is unavailable for list X but available for list 1 – X (denoted 1 – X by Y and denote the motion vector to be MVY), the motion vector is still considered as available for list X (by scaling the MVY towards the target reference picture in list X). Alternatively, or in addition, when both motion vectors in the corresponding block for list X and list Y (equal to 1-X) are available, the motion vectors from list X and list Y are not necessary to be used to be directly scaled to generate the two motion vectors of a current sub-PU by scaling. In one example, when formulating the ATMVP candidate, the low-delay check as done in TMVP applies to each sub-PU. If for every picture (denoted by refPic) in every reference picture list of the current slice, picture order count (POC) value of refPic is smaller than POC of current slice, current slice is considered with low-delay mode. In this low-delay mode, motion vectors from list X and list Y are scaled to generate the motion vectors of a current sub-PU for list X and list Y respectively. When not in the low-delay mode, only one motion vector MVZ from MVX or MVY is chosen and scaled to generate the two motion vectors for a current sub-PU. Similar to TMVP, in such a case Z is set equal to collocated_from_l0_flag, meaning that it depends on whether the collocated picture as in TMVP is in the list X or list Y of the current picture. Alternatively, Z is set as follows: if the motion source picture is identified from list X, Z is set to X. Alternatively, in addition, when the motion source pictures belong to both reference picture lists, and RefPicList0[idx0] is the motion source picture that is first present in list 0 and RefPicList(1)[idx1] is the motion source picture that is first present in list 1, Z is set to be 0 if idx0 is smaller than or equal to idx1, and set to be 1 otherwise.

**[0091]** The ‘564 Application also describes techniques for signaling the motion source picture, which may be performed by video encoder 20 and/or video decoder 30. In detail, a flag indicating whether the motion source picture is from list 0 or list 1 is signaled for a B slice. Alternatively, in addition, a reference index to a list 0 or list 1 of the current picture may be signaled to identify the motion source picture.

**[0092]** The ‘564 Application also describes techniques related to, when identifying a temporal vector, a vector being considered as unavailable (thus other ones can be

considered) if it points to an Intra coded block in the associated motion source picture, which may be performed by video encoder 20 and/or video decoder 30.

**[0093]** In accordance with the techniques of this disclosure, video encoder 20 and/or video decoder 30 may be configured to derive motion vectors for a sub-block (e.g., a sub-PU) of a block (e.g., a PU) from spatial and temporal neighboring blocks. As discussed below, a video coder (such as video encoder 20 or video decoder 30) may derive a motion vector for each sub-PU of a PU from information of neighboring blocks in a three-dimensional domain. This means the neighboring blocks could be the spatial neighbors in the current picture or the temporal neighbors in previous coded pictures. FIG. 10, discussed in greater detail below, is a flowchart illustrating an example spatial-temporal motion vector predictor (STMVP) derivation process. In addition, the methods described above with respect to bullets #1, #2, #3, #4, #6, #7, #12, and #13 could be directly extended to STMVP.

**[0094]** In the following description, the term "block" is used to refer the block-unit for storage of prediction related info, e.g. inter or intra prediction, intra prediction mode, motion information, etc. Such prediction info is saved and may be used for coding future blocks, e.g. predicting the prediction mode information for future blocks. In AVC and HEVC, the size of such a block is 4x4.

**[0095]** It is noted that in the following description, 'PU' indicates the inter-coded block unit and sub-PU to indicate the unit that derives the motion information from neighboring blocks.

**[0096]** Video encoder 20 and/or video decoder 30 may be configured to apply any of the following methods, alone or in any combination.

**[0097]** Sizes of sub-PUs and neighboring blocks: Considering a PU with multiple sub-PUs, the size of a sub-PU is usually equal to or bigger than that neighboring block size. In one example, as shown in FIG. 11A, shaded squares represent neighboring blocks (represented using lower-case letters, a, b, … i) that are outside of the current PU and the remaining squares (represented using upper-case letters, A, B, …, P) represent the sub-PUs in the current PU. The sizes of a sub-PU and its neighboring blocks are the same. For example, the size is equal to 4x4. FIG. 11B shows another example where sub-PUs are larger than the neighboring blocks. In this manner, sizes of neighboring blocks used for motion information derivation may be equal to or smaller than sizes of the sub-blocks for which motion information is derived. Alternatively, sub-PUs may take non-squared shapes, such as rectangle or triangle shapes. Furthermore, the size of

the sub-PU may be signaled in the slice header. In some examples, the process discussed above regarding signaling sub-block or sub-PU sizes, e.g., in a parameter set, may be extended to these techniques. For example, the sub-PU size may be signaled in a parameter set, such as a sequence parameter set (SPS) or a picture parameter set (PPS).

**[0098]** With respect to the example of FIG. 11A, assume that the video coder applies a raster scan order (A, B, C, D, E, etc.) to the sub-PUs to derive motion prediction for the sub-blocks. However, other scan orders may be applied also and it should be noted that these techniques are not limited to raster scan order only.

**[0099]** Neighboring blocks may be classified into two different types: spatial and temporal. A spatial neighboring block is an already coded block or an already scanned sub-PU that is in the current picture or slice and neighboring to the current sub-PU. A temporal neighboring block is a block in the previous coded picture and neighboring to the co-located block of the current sub-PU. In one example, the video coder uses all the reference pictures associated with a current PU to obtain the temporal neighboring block. In another example, the video coder uses a sub-set of reference pictures for STMVP derivation, e.g., only the first entry of each reference picture list.

**[0100]** Following these definitions, for sub-PU (A), with further reference to FIG. 11A, all white blocks (a, b, …, i) and their collocated blocks in previously coded pictures are spatial and temporal neighboring blocks that are treated as available. According to raster scan order, blocks B, C, D, E…P are not spatially available for sub-PU (A). Though, all sub-PUs (from A to P) are temporally available neighboring blocks for sub-PU (A), because their motion information can be found in their collocated blocks in previous coded pictures. Take sub-PU (G) as another example: its spatial neighboring blocks that are available include those from a, b… to i, and also from A to F. Furthermore, in some examples, certain restriction may be applied to the spatial neighbouring blocks, e.g., the spatial neighbouring blocks (i.e., from a, b … to i) may be restricted to be in the same LCU/slice/tile.

**[0101]** In accordance with the techniques of this disclosure, the video coder (video encoder 20 or video decoder 30) may select a subset of all available neighboring blocks to derive motion information or a motion field for each sub-PU. The subset used for derivation of each PU may be pre-defined; alternatively, video encoder 20 may signal (and video decoder 30 may receive signaled data indicating) the subset as high level syntax in a slice header, PPS, SPS, or the like. To optimize the coding performance, the

subset may be different for each sub-PU. In practice, a fixed pattern of location for the subset is preferred for simplicity. For example, each sub-PU may use its immediate above spatial neighbor, its immediate left spatial neighbor and its immediate bottom-right temporal neighbor as the subset. With respect to the example of FIG. 11A, when considering sub-PU (J) (horizontally hashed), the block above (F) and the block left (I) (diagonally down-left hashed) are spatially available neighboring blocks, and the bottom-right block (O) (diagonally hashed in both directions) is a temporally available neighboring block. With such a subset, sub-PUs in the current PU are to be processed sequentially (in the defined order, such as raster scan order) due to processing dependency.

**[0102]** Additionally or alternatively, when considering sub-PU (J), video encoder 20 and video decoder 30 may treat the block above (F) and the block left (I) as spatially available neighboring blocks, and the bottom block (N) and the right block (K) as temporally available neighboring blocks. With such a subset, video encoder 20 and video decoder 30 may process sub-PUs in the current PU sequentially due to processing dependency.

**[0103]** To allow paralleling processing of each sub-PU in the current PU, video encoder 20 and video decoder 30 may use a different subset of neighboring blocks for some sub-PUs for motion prediction derivation. In one example, a subset may be defined that only contains spatial neighbor blocks that do not belong to the current PU, e.g. blocks a,b,...i. In this case, parallel processing would be possible.

**[0104]** In another example, for a given sub-PU, if the sub-PUs spatial neighboring block is within the current PU, the collocated block of that spatial neighboring block may be put in the subset and used to derive the motion information of the current sub-PU. For example, when considering sub-PU (J), the temporal collocated blocks of the above block (F) and the left block (I) and bottom-right block (O) are selected as the subset to derive the motion of the sub-PU (J). In this case, the subset for sub-PU (J) contains three temporal neighboring blocks. In another example, a partially-paralleling process may be enabled wherein one PU is split into several regions and each region (covering several sub-PUs) could be processed independently.

**[0105]** Sometimes the neighboring blocks are intra-coded, wherein it is desirable to have a rule to determine replacement motion information for those blocks for better motion prediction and coding efficiency. For example, considering sub-PU (A), there might be cases where blocks b, c, and/or f are intra-coded, and a, d, e, g, h, and i are

inter-coded. For spatial neighbors, video encoder 20 and video decoder 30 may use a pre-defined order to populate the motion information of intra-coded blocks with that of the first found inter coded block. For example, the searching order of the above neighbors can be set as starting from the immediate above neighbor rightward until the rightmost neighbor, meaning the order of b, c, d, and e. The search order of the left neighbors can be set as starting from the immediate left neighbor downward until the bottommost neighbor, meaning the order of f, g, h, and i. If no inter-coded block is found through the search process, then above or left spatial neighbor is considered unavailable. For temporal neighbors, the same rule as specified in the TMVP derivation can be used. However, it should be noted that other rules can also be used, e.g. rules based on motion direction, temporal distance (search in different reference pictures) and spatial locations, etc.

**[0106]** Video encoder 20 and video decoder 30 may use the following method for deriving motion information for a given sub-PU in accordance with the techniques of this disclosure. Video encoder 20 and video decoder 30 may first determine a target reference picture, and perform motion vector scaling. For each neighboring block, motion vector scaling may be applied to its motion vector based on each reference picture list in order to map all the neighboring blocks' motion vectors to the same reference picture in each list. There are two steps: first, determine a source motion vector to be used for scaling. Second, determine a target reference picture where the source motion vector is projected to. For the first step, several methods can be used:

a) for each reference list, motion vector scaling is independent from a motion vector in another reference list; for a given block's motion information, if there is no motion vector in a reference list (e.g. uni-prediction mode instead of bi-prediction mode), no motion vector scaling is performed for that list.

b) motion vector scaling is not independent from motion vector in another reference list; for a given block's motion information, if no motion vector is unavailable in a reference list, it can be scaled from the one in another reference list.

c) both motion vectors are scaled from one pre-defined reference list (as in TMVP)

**[0107]** In one example, in accordance with the techniques of this disclosure, video encoder 20 and video decoder 30 use method a) above for scaling motion vectors of spatial neighboring blocks, and method c) above for scaling motion vectors of temporal neighboring blocks. However, other combinations may be used in other examples.

**[0108]** As for the second step, the target reference picture can be selected according to a certain rule based on the motion information (e.g. reference pictures) of available spatial neighboring blocks. One example of such a rule is the majority rule, i.e., selecting the reference picture shared by majority of the blocks. In this case, there is no signaling needed for the target reference picture from the encoder to decoder because the same information can also be inferred at decoder side using the same rule. Alternatively, such reference picture may also be specified explicitly in slice header, or signalled in some other methods to decoder. In one example, the target reference picture are determined as the first reference picture (refidx =0) of each reference list.

**[0109]** After determining the target reference picture and scaling the motion vectors as necessary, video encoder 20 and video decoder 30 derive motion information for a given sub-PU. Assume there are N available neighboring blocks with motion information for a given sub-PU. First, video encoder 20 and video decoder 30 determine the prediction direction (*InterDir*). One simple method for determining the prediction direction is as follows:

a. *InterDir* is initialized as zero, then looping through the motion information of N available neighboring blocks;
b. *InterDir* = ( *InterDir* bitwiseOR 1 ), if there is at least one motion vector in List 0;
c. *InterDir* = ( *InterDir* bitwiseOR 2 ), if there is at least one motion vector in List 1.

**[0110]** Here "bitwiseOR" represent the bitwise OR operation. The value of *InterDir* is defined as: 0 (no inter prediction), 1 (inter prediction based on List 0), 2 (inter prediction based on List 1), and 3 (inter prediction based on both List 0 and List 1), in this example.

**[0111]** Alternatively, similar to the determination on target reference picture for motion vector scaling described above, the majority rule may be used to determine the value of *InterDir* for the given sub-PU based on all available neighboring blocks' motion information.

**[0112]** After InterDir is determined, motion vectors can be derived. For each reference list based on the derived InterDir, there may be M motion vectors (M <= N) available through motion vector scaling to a target reference picture as discussed above. The motion vector for the reference list can be derived as:

$$(MV_x, MV_y) = ((\sum_{i=0}^{M} w_i * MV_{xi} + O_i)/\sum_{i=0}^{M} w_i, (\sum_{j=0}^{M} w_j * MV_{yj} + O_j)/\sum_{j=0}^{M} w_j) \quad (1)$$

where $w_i$ and $w_j$ are the weighting factors for the horizontal and the vertical motion component respectively, and $O_i$ and $O_j$ are the offset values that are dependent on the weighting factors.

**[0113]** The weighting factors may be determined based on various factors. In one example, the same rule may be applied to all sub-PUs within one PU. The rule may be defined as follows:

- For example, the weighting factor can be determined based on the location distance of the current sub-PU and a corresponding neighboring block.
- In another example, the weighting factor can also be determined based on the POC distance between the target reference picture and the reference picture associated with a corresponding neighboring block's motion vector before scaling.
- In yet another example, the weighting factor may be determined based on motion vector difference or consistency.
- For simplicity, all the weighting factors may also be set to 1.

**[0114]** Alternatively, different rules may be applied to sub-PUs within one PU. For example, the above rule may be applied, in addition, for sub-PUs located at the first row/first column, the weighting factors for motion vectors derived from temporal neighboring blocks are set to 0 while for the remaining blocks, the weighting factors for motion vectors derived from spatial neighboring blocks are set to 0.

**[0115]** It should be noted that in practice, the equations above may be implemented as is, or simplified for easy implementation. For example, to avoid division or floating point operation, fixed point operation may be used to approximate the equation above. One instance is that to avoid dividing by 3, one may instead choose to multiply with 43/128 to replace division operation with multiplication and bit-shift. Those variations in implementation should be considered covered under the same spirit of the techniques of this disclosure.

**[0116]** Additionally or alternatively, when the process invokes two motion vectors, equation (1) may be substituted with equation (2) below:

$$(MV_x, MV_y) = ((\sum_{i=0}^{1} MV_{xi})/2, (\sum_{i=0}^{1} MV_{yi})/2) \quad (2)$$

**[0117]** Additionally or alternatively, when the process invokes three motion vectors, equation (1) may be substituted with equation (3) below:

$$(MV_x, MV_y) = ((\sum_{i=0}^{2} MV_{xi} + sign(\sum_{i=0}^{2} MV_{xi}) * 1) * 43/128, (\sum_{i=0}^{2} MV_{yi} + sign(-\sum_{i=0}^{2} MV_{yi}) * 1) * 43/128)) \quad (3)$$

**[0118]** Additionally or alternatively, when the process invokes four motion vectors, equation (1) may be substituted with equation (4) below:

$$(\mathrm{MV_x}, \mathrm{MV_y}) = ((\sum_{\mathrm{i=0}}^{3} \mathrm{MV_{xi}} + \mathrm{sign}\left(\sum_{\mathrm{i=0}}^{3} \mathrm{MV_{xi}}\right) * 2)/4, (\sum_{\mathrm{i=0}}^{3} \mathrm{MV_{yi}} + \mathrm{sign}(\sum_{\mathrm{i=0}}^{3} \mathrm{MV_{yi}}) * 2)/4) \quad (4)$$

where sign(*t*) is 1 if t is a positive value and -1 if t is a negative value.

**[0119]** Additionally or alternatively, a non-linear operation may be also applied to derive the motion vectors, such as median filter.

**[0120]** Video encoder 20 and video decoder 30 may further determine motion vector availability for these techniques. Even when the motion vector predictors of each sub-PU are available, the STMVP mode may be reset to be unavailable for one PU. For example, once a motion vector predictor of each sub-PU is derived for a given PU, some availability checks are performed to determine if STMVP mode should be made available for the given PU. Such an operation is used to eliminate the cases where it is very unlikely for STMVP mode to be finally chosen for a given PU. When STMVP mode is not available, mode signaling does not include STMVP. In case that STMVP mode is implemented by inserting SMTVP in merge list, the merge list doesn't include this STMVP candidate when STMVP mode is determined to be not available. As a result, signaling overhead may be reduced.

**[0121]** Consider one PU partitioned into M sub-PUs. In one example, if N1 (N1<=M) sub-PUs among the M sub-PUs have the same motion vector predictor (i.e., same motion vectors and same reference picture indices), STMVP is only made available when N1 is smaller than a threshold or the predictor is different from other motion vector predictors (with smaller merge index) in the merge list. In another example, if N2 (N2<=M) sub-PUs under STMVP mode share the same motion vector predictors as corresponding sub-PUs under ATMVP, STMVP is only made available when N2 is smaller than another threshold. In one example, both thresholds for N1 and N2 are set equal to M.

**[0122]** Video encoder 20 and video decoder 30 may then insert the derived motion predictors into a candidate list, e.g., a merge list. If the STMVP candidate is available, video encoder 20 and video decoder 30 may insert the STMVP candidate into the

candidate list (e.g., merge list). The process in bullet #1 above can be extended and the STMVP candidate can be inserted either before or after the ATMVP candidate. In one example, video encoder 20 and video decoder 30 insert the STMVP immediately after the ATMVP candidate in the merge list.

**[0123]** Video encoder 20 may further send syntax data, such as block-based syntax data, frame-based syntax data, and GOP-based syntax data, to video decoder 30, e.g., in a frame header, a block header, a slice header, or a GOP header. The GOP syntax data may describe a number of frames in the respective GOP, and the frame syntax data may indicate an encoding/prediction mode used to encode the corresponding frame.

**[0124]** Video encoder 20 and video decoder 30 each may be implemented as any of a variety of suitable encoder or decoder circuitry, as applicable, such as one or more microprocessors, digital signal processors (DSPs), application specific integrated circuits (ASICs), field programmable gate arrays (FPGAs), discrete logic circuitry, software, hardware, firmware or any combinations thereof. Each of video encoder 20 and video decoder 30 may be included in one or more encoders or decoders, either of which may be integrated as part of a combined video encoder/decoder (CODEC). A device including video encoder 20 and/or video decoder 30 may comprise an integrated circuit, a microprocessor, and/or a wireless communication device, such as a cellular telephone.

**[0125]** FIG. 2 is a block diagram illustrating an example of video encoder 20 that may implement techniques for advanced temporal motion vector prediction (ATMVP). Video encoder 20 may perform intra- and inter-coding of video blocks within video slices. Intra-coding relies on spatial prediction to reduce or remove spatial redundancy in video within a given video frame or picture. Inter-coding relies on temporal prediction to reduce or remove temporal redundancy in video within adjacent frames or pictures of a video sequence. Intra-mode (I mode) may refer to any of several spatial based coding modes. Inter-modes, such as uni-directional prediction (P mode) or bi-prediction (B mode), may refer to any of several temporal-based coding modes.

**[0126]** As shown in FIG. 2, video encoder 20 receives a current video block within a video frame to be encoded. In the example of FIG. 2, video encoder 20 includes mode select unit 40, reference picture memory 64, summer 50, transform processing unit 52, quantization unit 54, and entropy encoding unit 56. Mode select unit 40, in turn, includes motion compensation unit 44, motion estimation unit 42, intra prediction unit 46, and partition unit 48. For video block reconstruction, video encoder 20 also

includes inverse quantization unit 58, inverse transform unit 60, and summer 62. A deblocking filter (not shown in FIG. 2) may also be included to filter block boundaries to remove blockiness artifacts from reconstructed video. If desired, the deblocking filter would typically filter the output of summer 62. Additional filters (in loop or post loop) may also be used in addition to the deblocking filter. Such filters are not shown for brevity, but if desired, may filter the output of summer 50 (as an in-loop filter).

**[0127]** During the encoding process, video encoder 20 receives a video frame or slice to be coded. The frame or slice may be divided into multiple video blocks. Motion estimation unit 42 and motion compensation unit 44 perform inter-predictive coding of the received video block relative to one or more blocks in one or more reference frames to provide temporal prediction. Intra-prediction unit 46 may alternatively perform intra-predictive coding of the received video block relative to one or more neighboring blocks in the same frame or slice as the block to be coded to provide spatial prediction. Video encoder 20 may perform multiple coding passes, e.g., to select an appropriate coding mode for each block of video data.

**[0128]** Moreover, partition unit 48 may partition blocks of video data into sub-blocks, based on evaluation of previous partitioning schemes in previous coding passes. For example, partition unit 48 may initially partition a frame or slice into LCUs, and partition each of the LCUs into sub-CUs based on rate-distortion analysis (e.g., rate-distortion optimization). Mode select unit 40 may further produce a quadtree data structure indicative of partitioning of an LCU into sub-CUs. Leaf-node CUs of the quadtree may include one or more PUs and one or more TUs.

**[0129]** Mode select unit 40 may select one of the coding modes, intra or inter, e.g., based on error results, and provides the resulting intra- or inter-coded block to summer 50 to generate residual block data and to summer 62 to reconstruct the encoded block for use as a reference frame. Mode select unit 40 also provides syntax elements, such as motion vectors, intra-mode indicators, partition information, and other such syntax information, to entropy encoding unit 56.

**[0130]** Motion estimation unit 42 and motion compensation unit 44 may be highly integrated, but are illustrated separately for conceptual purposes. Motion estimation, performed by motion estimation unit 42, is the process of generating motion vectors, which estimate motion for video blocks. A motion vector, for example, may indicate the displacement of a PU of a video block within a current video frame or picture relative to a predictive block within a reference frame (or other coded unit) relative to

the current block being coded within the current frame (or other coded unit). A predictive block is a block that is found to closely match the block to be coded, in terms of pixel difference, which may be determined by sum of absolute difference (SAD), sum of square difference (SSD), or other difference metrics. In some examples, video encoder 20 may calculate values for sub-integer pixel positions of reference pictures stored in reference picture memory 64. For example, video encoder 20 may interpolate values of one-quarter pixel positions, one-eighth pixel positions, or other fractional pixel positions of the reference picture. Therefore, motion estimation unit 42 may perform a motion search relative to the full pixel positions and fractional pixel positions and output a motion vector with fractional pixel precision.

**[0131]** Motion estimation unit 42 calculates a motion vector for a PU of a video block in an inter-coded slice by comparing the position of the PU to the position of a predictive block of a reference picture. The reference picture may be selected from a first reference picture list (List 0) or a second reference picture list (List 1), each of which identify one or more reference pictures stored in reference picture memory 64. Motion estimation unit 42 sends the calculated motion vector to entropy encoding unit 56 and motion compensation unit 44.

**[0132]** Motion compensation, performed by motion compensation unit 44, may involve fetching or generating the predictive block based on the motion vector determined by motion estimation unit 42. Again, motion estimation unit 42 and motion compensation unit 44 may be functionally integrated, in some examples. Upon receiving the motion vector for the PU of the current video block, motion compensation unit 44 may locate the predictive block to which the motion vector points in one of the reference picture lists. Summer 50 forms a residual video block by subtracting pixel values of the predictive block from the pixel values of the current video block being coded, forming pixel difference values, as discussed below. In general, motion estimation unit 42 performs motion estimation relative to luma components, and motion compensation unit 44 uses motion vectors calculated based on the luma components for both chroma components and luma components. Mode select unit 40 may also generate syntax elements associated with the video blocks and the video slice for use by video decoder 30 in decoding the video blocks of the video slice.

**[0133]** Mode select unit 40 may also select a sub-block (e.g., sub-PU) motion derivation mode for a block (e.g., a PU). That is, mode select unit 40 may compare a variety of encoding factors, including prediction modes, among a range of encoding passes to

determine which encoding pass (and thus, which set of factors, including which prediction mode) yields desirable rate-distortion optimization (RDO) characteristics. When mode select unit 40 selects the sub-block motion information derivation mode for a block of video data (e.g., a PU), motion compensation unit 44 may use the techniques of this disclosure to predict the block.

**[0134]** In particular, using the sub-block motion information derivation mode, motion compensation unit 44 may derive motion information for sub-blocks of the block. For example, motion compensation unit 44 may, for each sub-block, determine motion information for two or more neighboring sub-blocks and derive motion information for the sub-block from the motion information for the neighboring sub-blocks. The neighboring sub-blocks may include, for example, spatial and/or temporal neighboring sub-blocks. In one example, motion compensation unit 44 derives motion information for each sub-block by averaging the motion information (e.g., motion vectors) for a left-neighboring spatial sub-block, an above-neighboring spatial sub-block, and a bottom-right temporal neighboring sub-block, as discussed below in greater detail with respect to FIG. 11A. In other examples, motion compensation unit 44 may derive the motion information for each sub-block using, e.g., one of formulas (1)–(4). Motion compensation unit 44 may use the derived motion information for each of the sub-blocks to determine prediction data for the sub-blocks. By retrieving this prediction data for each of the sub-blocks, motion compensation unit 44 produces a predicted block for the current block using the sub-block motion information derivation mode.

**[0135]** Intra-prediction unit 46 may intra-predict a current block, as an alternative to the inter-prediction performed by motion estimation unit 42 and motion compensation unit 44, as described above. In particular, intra-prediction unit 46 may determine an intra-prediction mode to use to encode a current block. In some examples, intra-prediction unit 46 may encode a current block using various intra-prediction modes, e.g., during separate encoding passes, and intra-prediction unit 46 (or mode select unit 40, in some examples) may select an appropriate intra-prediction mode to use from the tested modes.

**[0136]** For example, intra-prediction unit 46 may calculate rate-distortion values using a rate-distortion analysis for the various tested intra-prediction modes, and select the intra-prediction mode having the best rate-distortion characteristics among the tested modes. Rate-distortion analysis generally determines an amount of distortion (or error) between an encoded block and an original, unencoded block that was encoded to

produce the encoded block, as well as a bitrate (that is, a number of bits) used to produce the encoded block. Intra-prediction unit 46 may calculate ratios from the distortions and rates for the various encoded blocks to determine which intra-prediction mode exhibits the best rate-distortion value for the block.

**[0137]** After selecting an intra-prediction mode for a block, intra-prediction unit 46 may provide information indicative of the selected intra-prediction mode for the block to entropy encoding unit 56. Entropy encoding unit 56 may encode the information indicating the selected intra-prediction mode. Video encoder 20 may include in the transmitted bitstream configuration data, which may include a plurality of intra-prediction mode index tables and a plurality of modified intra-prediction mode index tables (also referred to as codeword mapping tables), definitions of encoding contexts for various blocks, and indications of a most probable intra-prediction mode, an intra-prediction mode index table, and a modified intra-prediction mode index table to use for each of the contexts.

**[0138]** Video encoder 20 forms a residual video block by subtracting the prediction data from mode select unit 40 from the original video block being coded. Summer 50 represents the component or components that perform this subtraction operation. Transform processing unit 52 applies a transform, such as a discrete cosine transform (DCT) or a conceptually similar transform, to the residual block, producing a video block comprising residual transform coefficient values. Transform processing unit 52 may perform other transforms which are conceptually similar to DCT. Wavelet transforms, integer transforms, sub-band transforms or other types of transforms could also be used. In any case, transform processing unit 52 applies the transform to the residual block, producing a block of residual transform coefficients. The transform may convert the residual information from a pixel value domain to a transform domain, such as a frequency domain. Transform processing unit 52 may send the resulting transform coefficients to quantization unit 54. Quantization unit 54 quantizes the transform coefficients to further reduce bit rate. The quantization process may reduce the bit depth associated with some or all of the coefficients. The degree of quantization may be modified by adjusting a quantization parameter. In some examples, quantization unit 54 may then perform a scan of the matrix including the quantized transform coefficients. Alternatively, entropy encoding unit 56 may perform the scan.

**[0139]** Following quantization, entropy encoding unit 56 entropy codes the quantized transform coefficients. For example, entropy encoding unit 56 may perform context

adaptive variable length coding (CAVLC), context adaptive binary arithmetic coding (CABAC), syntax-based context-adaptive binary arithmetic coding (SBAC), probability interval partitioning entropy (PIPE) coding or another entropy coding technique. In the case of context-based entropy coding, context may be based on neighboring blocks. Following the entropy coding by entropy encoding unit 56, the encoded bitstream may be transmitted to another device (e.g., video decoder 30) or archived for later transmission or retrieval.

**[0140]** Moreover, entropy encoding unit 56 may encode various other syntax elements for the various blocks of video data. For example, entropy encoding unit 56 may encode syntax elements representative of a prediction mode for each PU of each CU of the video data. When inter-prediction is indicated for a PU, entropy encoding unit 56 may encode motion information, which may include whether a motion vector is encoded using merge mode or advanced motion vector prediction (AMVP). In either case, video encoder 20 forms a candidate list including candidates (spatial and/or temporal neighboring blocks to the PU) from which the motion information may be predicted. In accordance with the techniques of this disclosure, the candidate list may include a candidate that indicates that a sub-block motion information derivation mode is to be used for the PU. Furthermore, entropy encoding unit 56 may encode a candidate index into the candidate list that indicates which of the candidates is to be used. Thus, if the sub-block motion information derivation mode is selected, entropy encoding unit 56 encodes a candidate index referring to the candidate that represents the sub-block motion information derivation mode.

**[0141]** Inverse quantization unit 58 and inverse transform unit 60 apply inverse quantization and inverse transformation, respectively, to reconstruct the residual block in the pixel domain, e.g., for later use as a reference block. Motion compensation unit 44 may calculate a reference block by adding the residual block to a predictive block of one of the frames of reference picture memory 64. Motion compensation unit 44 may also apply one or more interpolation filters to the reconstructed residual block to calculate sub-integer pixel values for use in motion estimation. Summer 62 adds the reconstructed residual block to the motion compensated prediction block produced by motion compensation unit 44 to produce a reconstructed video block for storage in reference picture memory 64. The reconstructed video block may be used by motion estimation unit 42 and motion compensation unit 44 as a reference block to inter-code a block in a subsequent video frame.

**[0142]** In this manner, video encoder 20 represents an example of a video encoder configured to determine that a motion prediction candidate for a current block of video data indicates that motion information is to be derived for sub-blocks of the current block, and in response to the determination: partition the current block into the sub-blocks, for each of the sub-blocks, derive motion information using motion information for at least two neighboring blocks, and decode the sub-blocks using the respective derived motion information. That is, video encoder 20 both encodes and decodes blocks of video data using the techniques of this disclosure.

**[0143]** FIG. 3 is a block diagram illustrating an example of video decoder 30 that may implement techniques for advanced temporal motion vector prediction (ATMVP). In the example of FIG. 3, video decoder 30 includes an entropy decoding unit 70, motion compensation unit 72, intra prediction unit 74, inverse quantization unit 76, inverse transformation unit 78, reference picture memory 82 and summer 80. Video decoder 30 may, in some examples, perform a decoding pass generally reciprocal to the encoding pass described with respect to video encoder 20 (FIG. 2). Motion compensation unit 72 may generate prediction data based on motion vectors received from entropy decoding unit 70, while intra-prediction unit 74 may generate prediction data based on intra-prediction mode indicators received from entropy decoding unit 70.

**[0144]** During the decoding process, video decoder 30 receives an encoded video bitstream that represents video blocks of an encoded video slice and associated syntax elements from video encoder 20. Entropy decoding unit 70 of video decoder 30 entropy decodes the bitstream to generate quantized coefficients, motion vectors or intra-prediction mode indicators, and other syntax elements. Entropy decoding unit 70 forwards the motion vectors and other syntax elements to motion compensation unit 72. Video decoder 30 may receive the syntax elements at the video slice level and/or the video block level.

**[0145]** When the video slice is coded as an intra-coded (I) slice, intra prediction unit 74 may generate prediction data for a video block of the current video slice based on a signaled intra prediction mode and data from previously decoded blocks of the current frame or picture. When the video frame is coded as an inter-coded (i.e., B, P or GPB) slice, motion compensation unit 72 produces predictive blocks for a video block of the current video slice based on the motion vectors and other syntax elements received from entropy decoding unit 70. The predictive blocks may be produced from one of the reference pictures within one of the reference picture lists. Video decoder 30 may

construct the reference frame lists, List 0 and List 1, using default construction techniques based on reference pictures stored in reference picture memory 82.

**[0146]** Motion compensation unit 72 determines prediction information for a video block of the current video slice by parsing the motion vectors and other syntax elements, and uses the prediction information to produce the predictive blocks for the current video block being decoded. For example, motion compensation unit 72 uses some of the received syntax elements to determine a prediction mode (e.g., intra- or inter-prediction) used to code the video blocks of the video slice, an inter-prediction slice type (e.g., B slice, P slice, or GPB slice), construction information for one or more of the reference picture lists for the slice, motion vectors for each inter-encoded video block of the slice, inter-prediction status for each inter-coded video block of the slice, and other information to decode the video blocks in the current video slice.

**[0147]** Motion compensation unit 72 may also perform interpolation based on interpolation filters. Motion compensation unit 72 may use interpolation filters as used by video encoder 20 during encoding of the video blocks to calculate interpolated values for sub-integer pixels of reference blocks. In this case, motion compensation unit 72 may determine the interpolation filters used by video encoder 20 from the received syntax elements and use the interpolation filters to produce predictive blocks.

**[0148]** In accordance with the techniques of this disclosure, entropy decoding unit 70 decodes a value for a candidate index that refers to a candidate list, and passes the value of the candidate index to motion compensation unit 72, when a block, such as a PU, is predicted using inter-prediction. The value of the candidate index may refer to a candidate in the candidate list that represents that the block is predicted using a sub-block motion information derivation mode. If the value of the candidate index does refer to candidate in the candidate list that represents that the block is predicted using a sub-block motion information derivation mode, motion compensation unit 72 may generate a predicted block for the block using the sub-block motion information derivation mode.

**[0149]** More particularly, using the sub-block motion information derivation mode, motion compensation unit 72 may derive motion information for sub-blocks of the block. For example, motion compensation unit 72 may, for each sub-block, determine motion information for two or more neighboring sub-blocks and derive motion information for the sub-block from the motion information for the neighboring sub-blocks. The neighboring sub-blocks may include, for example, spatial and/or temporal

neighboring sub-blocks. In one example, motion compensation unit 72 derives motion information for each sub-block by averaging the motion information (e.g., motion vectors) for a left-neighboring spatial sub-block, an above-neighboring spatial sub-block, and a bottom-right temporal neighboring sub-block, as discussed below in greater detail with respect to FIG. 11A. In other examples, motion compensation unit 72 may derive the motion information for each sub-block using, e.g., one of formulas (1)–(4). Motion compensation unit 72 may use the derived motion information for each of the sub-blocks to determine prediction data for the sub-blocks. By retrieving this prediction data for each of the sub-blocks, motion compensation unit 72 produces a predicted block for the current block using the sub-block motion information derivation mode.

**[0150]** Inverse quantization unit 76 inverse quantizes, i.e., de-quantizes, the quantized transform coefficients provided in the bitstream and decoded by entropy decoding unit 70. The inverse quantization process may include use of a quantization parameter QPY calculated by video decoder 30 for each video block in the video slice to determine a degree of quantization and, likewise, a degree of inverse quantization that should be applied.

**[0151]** Inverse transform unit 78 applies an inverse transform, e.g., an inverse DCT, an inverse integer transform, or a conceptually similar inverse transform process, to the transform coefficients in order to produce residual blocks in the pixel domain.

**[0152]** After motion compensation unit 72 generates the predictive block for the current video block based on the motion vectors and other syntax elements, video decoder 30 forms a decoded video block by summing the residual blocks from inverse transform unit 78 with the corresponding predictive blocks generated by motion compensation unit 72. Summer 80 represents the component or components that perform this summation operation. If desired, a deblocking filter may also be applied to filter the decoded blocks in order to remove blockiness artifacts. Other loop filters (either in the coding loop or after the coding loop) may also be used to smooth pixel transitions, or otherwise improve the video quality. The decoded video blocks in a given frame or picture are then stored in reference picture memory 82, which stores reference pictures used for subsequent motion compensation. Reference picture memory 82 also stores decoded video for later presentation on a display device, such as display device 32 of FIG. 1.

**[0153]** In this manner, video decoder 30 represents an example of a video decoder configured to determine that a motion prediction candidate for a current block of video

data indicates that motion information is to be derived for sub-blocks of the current block, and in response to the determination: partition the current block into the sub-blocks, for each of the sub-blocks, derive motion information using motion information for at least two neighboring blocks, and decode the sub-blocks using the respective derived motion information.

**[0154]** FIG. 4 is a conceptual diagram illustrating spatial neighboring candidates in HEVC. Spatial MV candidates are derived from the neighboring blocks shown on FIG. 4, for a specific PU (PU0), although the methods of generating the candidates from the blocks differ for merge and AMVP modes.

**[0155]** In merge mode, up to four spatial MV candidates can be derived with the orders shown in FIG. 4(a) 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. 4 (a). That is, in FIG. 4(a), block 100 includes PU0 104A and PU1 104B. When a video coder is to code motion information for PU0 104A using merge mode, the video coder adds motion information from spatial neighboring blocks 108A, 108B, 108C, 108D, and 108E to a candidate list, in that order. Blocks 108A, 108B, 108C, 108D, and 108E may also be referred to as, respectively, blocks A1, B1, B0, A0, and B2, as in HEVC.

**[0156]** 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. 4 (b). These blocks are labeled, respectively, as blocks 110A, 110B, 110C, 110D, and 110E in FIG. 4(b). In particular, in FIG. 4(b), block 102 includes PU0 106A and PU1 106B, and blocks 110A, 110B, 110C, 110D, and 110E represent spatial neighbors to PU0 106A. 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.

**[0157]** FIG. 5 is a conceptual diagram illustrating temporal motion vector prediction in HEVC. In particular, FIG. 5(a) illustrates an example CU 120 including PU0 122A and PU 1 122B. PU0 122A includes a center block 126 for PU 122A and a bottom-right block 124 to PU0 122A. FIG. 5(a) also shows an external block 128 for which motion information may be predicted from motion information of PU0 122A, as discussed

below. FIG. 5(b) illustrates a current picture 130 including a current block 138 for which motion information is to be predicted. In particular, FIG. 5(b) illustrates a collocated picture 134 to current picture 130 (including collocated block 140 to current block 138), a current reference picture 132, and a collocated reference picture 136. Collocated block 140 is predicted using motion vector 144, which is used as a temporal motion vector predictor (TMVP) 142 for motion information of block 138.

**[0158]** A video coder may add a TMVP candidate (e.g., TMVP candidate 142) into the MV candidate list after any spatial motion vector candidates if TMVP is enabled and the TMVP candidate is available. The process of motion vector derivation for the 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 set to 0, according to HEVC.

**[0159]** The primary block location for the TMVP candidate derivation is the bottom right block outside of the collocated PU, as shown in FIG. 5 (a) as block 124 to PU0 122A, to compensate the bias to the above and left blocks used to generate spatial neighboring candidates. However, if block 124 is located outside of the current CTB row or motion information is not available for block 124, the block is substituted with center block 126 of the PU as shown in FIG. 5(a).

**[0160]** The motion vector for TMVP candidate 142 is derived from co-located block 140 of collocated picture 134, as indicated in slice level information.

**[0161]** Similar to temporal direct mode in AVC, a motion vector of the TMVP candidate may be subject to motion vector scaling, which is performed to compensate picture order count (POC) distance differences between current picture 130 and current reference picture 132, and collocated picture 134 and collocated reference picture 136. That is, motion vector 144 may be scaled to produce TMVP candidate 142, based on these POC differences.

**[0162]** Several aspects of merge and AMVP modes of HEVC are discussed below.

**[0163]** Motion vector scaling: It is assumed that the value of a motion vector is proportional to the distance between pictures in 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 used by video encoder 20 or video decoder 30 to predict another motion vector, the distance between the containing picture and the reference picture is calculated based on Picture Order Count (POC) values.

**[0164]** For a motion vector to be predicted, its associated containing picture and reference picture are different. That is, there are two POC difference values for two distinct motion vectors: a first motion vector to be predicted, and a second motion vector used to predict the first motion vector. Moreover, the first POC difference is the difference between the current picture and the reference picture of the first motion vector, and the second POC difference is the difference between the picture containing the second motion vector and the reference picture to which the second motion vector refers. The second motion vector may be 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.

**[0165]** Artificial motion vector candidate generation: If a motion vector candidate list is not complete, artificial motion vector candidates may be generated and inserted at the end of the list until the list includes a predetermined number of candidates.

**[0166]** In merge mode, there are two types of artificial MV candidates: combined candidates derived only for B-slices and zero candidates used only for AMVP if the first type does not provide enough artificial candidates.

**[0167]** 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.

**[0168]** The following is a description of an example pruning process for candidate insertion. Candidates from different blocks may happen to be the same, which decreases the efficiency of a merge/AMVP candidate list. A pruning process may be applied to solve this problem. According to the pruning process, a video coder compares one candidate to the others in the current candidate list to avoid inserting an identical candidate, to a certain extent. To reduce the complexity, only limited numbers of pruning processes are applied, instead of comparing each potential candidate with all other existing candidates already in the list.

**[0169]** FIG. 6 illustrates an example prediction structure for 3D-HEVC. 3D-HEVC is the 3D video extension of HEVC under development by JCT-3V. The key techniques related to the techniques of this disclosure are described in this sub-section.

**[0170]** FIG. 6 shows a multiview prediction structure for a three-view case. V3 denotes the base view and a picture in a non-base view (V1 or V5) can be predicted from pictures in a dependent (base) view of the same time instance.

**[0171]** It is worth mentioning that the inter-view sample prediction (from reconstructed samples) is supported in MV-HEVC, a typical prediction structure of which is shown in FIG. 8.

**[0172]** Both MV-HEVC and 3D-HEVC are compatible with HEVC in a way that the base (texture) view is decodable by an HEVC (version 1) decoder. A test model for MV-HEVC and 3D-HEVC is described in Zhang et al., "Test Model 6 of 3D-HEVC and MV-HEVC," JCT-3V document ISO/IEC JTC1/SC29/WG11 N13940, available at mpeg.chiariglione.org/standards/mpeg-h/high-efficiency-video-coding/test-model-6-3d-hevc-and-mv-hevc as of January 26, 2015.

**[0173]** In MV-HEVC, a current picture in a non-base view may be predicted by both pictures in the same view and pictures in a reference view of the same time instance, by putting all of these pictures in reference picture lists of the picture. Therefore, a reference picture list of the current picture contains both temporal reference pictures and inter-view reference pictures.

**[0174]** A motion vector associated with a reference index corresponding to a temporal reference picture is denoted as a temporal motion vector.

**[0175]** A motion vector associated with a reference index corresponding to a inter-view reference picture is denoted as a disparity motion vector.

**[0176]** 3D-HEVC supports all features in MV-HEVC; therefore, the inter-view sample prediction as mentioned above is enabled.

**[0177]** In addition more advanced texture only coding tools and depth related/dependent coding tools are supported.

**[0178]** The texture only coding tools often requires the identification of the corresponding blocks (between views) that may belong to the same object. Therefore disparity vector derivation is a basic technology in 3D-HEVC.

**[0179]** FIG. 7 is a conceptual diagram illustrating sub-PU based inter-view motion prediction in 3D-HEVC. FIG. 7 shows current picture 160 of a current view (V1) and a collocated picture 162 in a reference view (V0). Current picture 160 includes a current PU 164 including four sub-Pus 166A–166D (sub-PUs 166). Respective disparity vectors 174A–174D (disparity vectors 174) identify corresponding sub-PUs 168A–168D to sub-PUs 166 in collocated picture 162. 3D-HEVC describes a sub-PU level

inter-view motion prediction method for the inter-view merge candidate, i.e., the candidate derived from a reference block in the reference view.

**[0180]** When such a mode is enabled, current PU 164 may correspond to a reference area (with the same size as the current PU identified by the disparity vector) in the reference view and the reference area may have richer motion information than needed for generation of one set of motion information typically for a PU. Therefore, a sub-PU level inter-view motion prediction (SPIVMP) method may be used, as shown in FIG. 7.

**[0181]** This mode may also be signaled as a special merge candidate. Each of the sub-PUs contains a full set of motion information. Therefore, a PU may contain multiple sets of motion information.

**[0182]** Sub-PU based motion parameter inheritance (MPI) in 3D-HEVC: Similarly, in 3D-HEVC, the MPI candidate can also be extended in a way similar to sub-PU level inter-view motion prediction. For example, if the current depth PU has a co-located region which contains multiple PUs, the current depth PU may be separated into sub-PUs, and each PU may have a different set of motion information. This method is called sub-PU MPI. That is, motion vectors 172A–172D of corresponding sub-PUs 168A–168D may be inherited by sub-PUs 166A–166D, as motion vectors 170A–170D, as shown in FIG. 7.

**[0183]** Sub-PU related information for 2D video coding: In U.S. Application No. 14/497,128, filed September 25, 2014, published as U.S. Publication No. 2015/0086929 on Mar. 26, 2015, a sub-PU based advanced TMVP design is described. In single-layer coding, a two-stage advanced temporal motion vector prediction design is proposed.

**[0184]** A first stage is to derive a vector identifying the corresponding block of the current prediction unit (PU) in a reference picture, and a second stage is to extract multiple sets motion information from the corresponding block and assign them to sub-PUs of the PU. Each sub-PU of the PU therefore is motion compensated separately. The concept of the ATMVP is summarized as follows:

1. The vector in the first stage can be derived from spatial and temporal neighboring blocks of the current PU.
2. This process may be achieved as activating a merge candidate among all the other merge candidates.

**[0185]** Applicable to single-layer coding and sub-PU temporal motion vector prediction, a PU or CU may have motion refinement data to be conveyed on top of the predictors.

**[0186]** Several design aspects of the 14/497,128 application are highlighted as follows:

1. The first stage of vector derivation can also be simplified by just a zero vector.

2. The first stage of vector derivation may include identifying jointly the motion vector and its associated picture. Various ways of selecting the associated picture, and further deciding the motion vector to be the first stage vector, have been proposed.

3. If the motion information during the above process is unavailable, the "first stage vector" is used for substitution.

4. A motion vector identified from a temporal neighbor has to be scaled to be used for the current sub-PU, in a way similar to motion vector scaling in TMVP. However, which reference picture such a motion vector may be scaled to can be designed with one of the following ways:

    a. The picture is identified by a fixed reference index of the current picture.

    b. The picture is identified to be the reference picture of the corresponding temporal neighbor, if also available in a reference picture list of the current picture.

    c. The picture is set to be the collocated picture identified in the first stage and from where the motion vectors are retrieved.

**[0187]** FIG. 8 is a conceptual diagram illustrating sub-PU motion prediction from a reference picture. In this example, current picture 180 includes a current PU 184 (e.g., a PU). In this example, motion vector 192 identifies PU 186 of reference picture 182 relative to PU 184. PU 186 is partitioned into sub-PUs 188A–188D, each having respective motion vectors 190A–190D. Thus, although current PU 184 is not actually partitioned into separate sub-PUs, in this example, current PU 184 may be predicted using motion information from sub-PUs 188A–188D. In particular, a video coder may code sub-PUs of current PU 184 using respective motion vectors 190A–190D. However, the video coder need not code syntax elements indicating that current PU 184 is split into sub-PUs. In this manner, current PU 184 may be effectively predicted using multiple motion vectors 190A–190D, inherited from respective sub-PUs 188A–188D, without the signaling overhead of syntax elements used to split current PU 184 into multiple sub-PUs.

**[0188]** FIG. 9 is a conceptual diagram illustrating relevant pictures in ATMVP (similar to TMVP). In particular, FIG. 9 illustrates current picture 204, motion source picture 206, and reference pictures 200, 202. More particularly, current picture 204 includes current block 208. Temporal motion vector 212 identifies corresponding block 210 of motion

source picture 206 relative to current block 208. Corresponding block 210, in turn, includes motion vector 214, which refers to reference picture 202 and acts as an advanced temporal motion vector predictor for at least a portion of current block 208, e.g., a sub-PU of current block 208. That is, motion vector 214 may be added as a candidate motion vector predictor for current block 208. If selected, at least a portion of current block 208 may be predicted using a corresponding motion vector, namely, motion vector 216, which refers to reference picture 200.

**[0189]** FIG. 10 is a flowchart illustrating an example method in accordance with the techniques of this disclosure. The method of FIG. 10 may be performed by video encoder 20 and/or video decoder 30. For generality, the method of FIG. 10 is explained as being performed by a "video coder," which again, may correspond to either of video encoder 20 or video decoder 30.

**[0190]** Initially, a video coder obtains an available motion field from spatial or temporal neighboring blocks for a current sub-PU of a PU (230). The video coder then derives motion information from the obtained neighboring motion field (232). The video coder then determines whether motion information for all sub-PUs of the PU has been derived (234). If not ("NO" branch of 234), the video coder derives motion information for a remaining sub-PU (230). On the other hand, if motion information for all sub-PUs has been derived ("YES" branch of 234), the video coder determines availability of a spatial-temporal sub-PU motion predictor (236), e.g., as explained above. The video coder inserts the spatial-temporal sub-PU motion predictor into a merge list, if the spatial-temporal sub-PU motion predictor is available (238).

**[0191]** Although not shown in the method of FIG. 10, the video coder may then code the PU (e.g., each of the sub-PUs of the PU) using the merge candidate list. For instance, when performed by video encoder 20, video encoder 20 may calculate residual block(s) for the PU (e.g., for each sub-PU) using the sub-PUs as predictors, transform and quantize the residual block(s), and entropy encode the resulting quantized transform coefficients. Video decoder 30, similarly, may entropy decode received data to reproduce quantized transform coefficients, inverse quantize and inverse transform these coefficients to reproduce the residual block(s), and then combine the residual block(s) with the corresponding sub-PUs to decode a block corresponding to the PU.

**[0192]** FIGS. 11A and 11B are conceptual diagrams illustrating examples of blocks including sub-blocks that are predicted using derived motion information. In particular, FIG. 11A illustrates block 250 (e.g., a PU) including sub-blocks 254A–254P (sub-

blocks 254), which may represent sub-PUs when block 250 is a PU. Neighboring sub-blocks 256A–256I (neighboring sub-blocks 256) to block 250 are also shown in FIG. 11A and shaded light grey.

**[0193]** In general, a video coder, such as video encoder 20 or video decoder 30, may derive motion information for sub-blocks 254 of block 250 using motion information from two or more neighboring blocks. The neighboring blocks may include spatially neighboring and/or temporally neighboring blocks. For example, a video coder may derive motion information for sub-block 254J from spatially neighboring sub-blocks 254F and 254I, and from a temporally neighboring block corresponding to the position of sub-block 254O. The temporally neighboring block may be from a previously coded picture that is co-located with sub-block 254O. To derive a motion vector of the motion information for sub-block 254J, the video coder may average motion vectors for sub-block 254F, sub-block 254I, and the temporally neighboring block that is co-located with sub-block 254O. Alternatively, the video coder may derive the motion vector using one of formulas (1)–(4) as discussed above.

**[0194]** In some examples, the video coder may be configured to always derive motion information from sub-blocks outside of block 250, e.g., neighboring sub-blocks 256 and/or temporally neighboring sub-blocks. Such a configuration may allow sub-blocks 254 to be coded in parallel. For example, the video coder may derive motion information for sub-block 254A from motion information of sub-blocks 256B and 256F, and a temporally neighboring sub-block that is co-located with sub-block 254F. The video coder may also derive motion information for sub-block 254B in parallel with sub-block 254A, using motion information of sub-blocks 256C, 256B, 256F, and temporally neighboring sub-blocks that are co-located with sub-blocks 254F and 254G.

**[0195]** FIG. 11B illustrates block 260 (e.g., a PU) including sub-blocks 264A–264D (sub-blocks 264), which again may represent sub-PUs. FIG. 11B also illustrates neighboring sub-blocks 266A–266I (neighboring sub-blocks 266). In general, the example of FIG. 11B indicates that sub-blocks of a block, such as block 260, may be of a variety of sizes, and may be larger than neighboring blocks used to derive motion information. In this example, sub-blocks 264 are larger than neighboring sub-blocks 266. Nevertheless, video coders (such as video encoder 20 and video decoder 30) may be configured to apply similar techniques to sub-blocks 264 as those discussed above with respect to sub-blocks 254.

**[0196]** FIG. 12 is a flowchart illustrating an example method of encoding video data in accordance with the techniques of this disclosure. The method of FIG. 12 is described with respect to video encoder 20 (FIGS. 1 and 2) and the components thereof, for purposes of explanation and example. However, it should be understood that other video encoding devices may be configured to perform these or similar techniques. Moreover, certain steps may be omitted, performed in a different order, and/or performed in parallel.

**[0197]** Initially, video encoder 20 partitions a coding unit (CU) into one or more prediction units (PUs) (270). Video encoder 20 may then test a variety of prediction modes (e.g., spatial or intra-prediction, temporal or inter-prediction, and sub-block motion derivation prediction) for each of the PUs (272). In particular, mode select unit 40 may test a variety of prediction modes, and select one of the modes for a PU that yields the best rate-distortion characteristics for the PU. It is assumed for purposes of example that video encoder 20 selects the sub-PU motion derivation mode (274) for a PU of the CU.

**[0198]** According to the sub-PU motion derivation mode, video encoder 20 partitions the PU into sub-PUs (276). In general, the sub-PUs are distinguishable from the PU in that separate information, such as motion information, is not coded for the sub-PUs. Instead, according to the techniques of this disclosure, video encoder 20 derives motion information for the sub-PUs from neighboring sub-PUs (278). The neighboring sub-PUs may include spatial and/or temporal neighboring sub-PUs. For example, the neighboring sub-PUs may be selected as discussed with respect to FIG. 11A. That is, in this example, for each sub-PU, video encoder 20 derives motion information from neighboring sub-PUs including an above-neighboring spatial sub-PU, a left-neighboring spatial sub-PU, and a bottom-right temporal neighboring sub-PU. Video encoder 20 may calculate the derived motion information as an average of the motion information of the neighboring sub-PUs or according to formulas (1)–(4) discussed above.

**[0199]** Video encoder 20 may then predict the sub-PUs using the derived motion information (280). That is, motion compensation unit 44 of video encoder 20 may retrieve predicted information for each of the sub-PUs of the PU using the derived motion information for the respective sub-PUs. Video encoder 20 may form a predicted block for the PU as an assembly of each of the predicted sub-PUs in their respective positions of the PU.

**[0200]** Video encoder 20 may then calculate a residual block for the PU (282). For example, summer 50 may calculate pixel-by-pixel differences between the original version of the PU and the predicted block, forming the residual block. Then, transform processing unit 52 and quantization unit 54 of video encoder 20 may transform and quantize the residual block, respectively, to produce quantized transform coefficients (284). Entropy encoding unit 56 may then entropy encode the quantized transform coefficients (286).

**[0201]** Moreover, entropy encoding unit 56 may entropy encode a candidate index for the PU that indicates that the PU is predicted using the sub-PU motion derivation mode (286). In particular, entropy encoding unit 56 may construct a candidate list including a plurality of motion prediction candidates, as well as a candidate representing the sub-PU motion derivation mode. Thus, when video encoder 20 selects the sub-PU motion information derivation mode, entropy encoding unit 56 entropy encodes a value representing an index that identifies the position of the candidate representing the sub-PU motion derivation mode in the candidate list for the PU.

**[0202]** After encoding the PU in the manner described above, video encoder 20 also decodes the PU in a substantially similar, albeit reciprocal, manner. Although not shown in FIG. 12, video encoder 20 also inverse transforms and inverse quantizes the quantized transform coefficients to reproduce the residual block, and combines the residual block with the predicted block to decode the PU, for use as a reference block during subsequent prediction (e.g., intra- and/or inter-prediction).

**[0203]** In this manner, the method of FIG. 12 represents an example of a method including determining that a motion prediction candidate for a current block of video data indicates that motion information is to be derived for sub-blocks of the current block, and in response to the determination: partitioning the current block into the sub-blocks, for each of the sub-blocks, deriving motion information using motion information for at least two neighboring blocks, and encoding (and decoding) the sub-blocks using the respective derived motion information.

**[0204]** FIG. 13 is an example of a method of decoding video data in accordance with the techniques of this disclosure. The method of FIG. 13 is described with respect to video decoder 30 (FIGS. 1 and 3) and the components thereof, for purposes of explanation and example. However, it should be understood that other video decoding devices may be configured to perform these or similar techniques. Moreover, certain steps may be omitted, performed in a different order, and/or performed in parallel.

**[0205]** Initially, entropy decoding unit 70 of video decoder 30 entropy decodes a candidate index of a candidate in a candidate list that indicates that sub-PU motion derivation mode is used for a prediction unit (290). Although not shown, it should be understood that, initially, video decoder 30 constructs the candidate list and adds candidates to the candidate list. In this example, for purposes of explanation, the candidate index refers to the candidate that represents the sub-PU motion derivation mode. However, in general, it should be understood that the candidate index could refer to any of the candidates in the candidate list for the PU.

**[0206]** In this example, because the candidate index refers to the candidate representing that sub-PU motion derivation mode is to be used for the PU, video decoder 30 partitions the PU into sub-PUs (292). Motion compensation unit 72 of video decoder 30 then derives motion information for each of the sub-PUs from neighboring sub-PUs (294). The neighboring sub-PUs may include spatial and/or temporal neighboring sub-PUs. For example, the neighboring sub-PUs may be selected as discussed with respect to FIG. 11A. That is, in this example, for each sub-PU, video decoder 30 derives motion information from neighboring sub-PUs including an above-neighboring spatial sub-PU, a left-neighboring spatial sub-PU, and a bottom-right temporal neighboring sub-PU. Video decoder 30 may calculate the derived motion information as an average of the motion information of the neighboring sub-PUs or according to formulas (1)–(4) discussed above.

**[0207]** Video decoder 30 may then predict the sub-PUs using the derived motion information (296). That is, motion compensation unit 72 of video decoder 30 may retrieve predicted information for each of the sub-PUs of the PU using the derived motion information for the respective sub-PUs. Video decoder 30 may form a predicted block for the PU as an assembly of each of the predicted sub-PUs in their respective positions of the PU.

**[0208]** Entropy decoding unit 70 of video decoder 30 may further entropy decode quantized transform coefficients (298) of the PU. Inverse quantization unit 76 and inverse transform unit 78 may inverse quantize and inverse transform, respectively, the quantized transform coefficients to produce a residual block for the PU (300). Video decoder 30 may then decode the prediction unit using the predicted block and the residual block (302). In particular, summer 80 may combine the predicted block and the residual block on a pixel-by-pixel basis to decode the prediction unit.

**[0209]** In this manner, the method of FIG. 13 represents an example of a method of decoding video data including determining that a motion prediction candidate for a current block of video data indicates that motion information is to be derived for sub-blocks of the current block, and in response to the determination: partitioning the current block into the sub-blocks, for each of the sub-blocks, deriving motion information using motion information for at least two neighboring blocks, and decoding the sub-blocks using the respective derived motion information.

**[0210]** 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.

**[0211]** 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.

**[0212]** 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.

**[0213]** 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 term "processor," as used herein may refer to any of the foregoing structure 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.

**[0214]** 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.

**[0215]** Various examples have been described. These and other examples are within the scope of the following claims.

**WHAT IS CLAIMED IS:**

1. A method of decoding video data, the method comprising:

determining that a motion prediction candidate for a current block of video data indicates that motion information is to be derived for sub-blocks of the current block;

in response to the determination:

partitioning the current block into the sub-blocks;

for each of the sub-blocks, deriving motion information using motion information for at least two neighboring blocks; and

decoding the sub-blocks using the respective derived motion information.

2. The method of claim 1, wherein the at least two neighboring blocks are selected from a group including an above-neighboring block, a left-neighboring block, and a temporally neighboring block.

3. The method of claim 2, wherein the above-neighboring block comprises an above-neighboring sub-block within the current block.

4. The method of claim 2, wherein the left-neighboring block comprises a left-neighboring sub-block within the current block.

5. The method of claim 2, wherein the above-neighboring block comprises an above-neighboring sub-block outside of the current block.

6. The method of claim 2, wherein the left-neighboring block comprises a left-neighboring sub-block outside of the current block.

7. The method of claim 2, wherein the temporally neighboring block comprises a block in a previously decoded picture that neighbors a block that is co-located with the current block in the previously decoded picture.

8. The method of claim 2, wherein deriving the motion information comprises deriving the motion information using an average of the motion information for the above-neighboring block, the left-neighboring block, and the temporally neighboring block.

9. The method of claim 1, wherein the neighboring blocks have sizes equal to or less than sizes of the sub-blocks.

10. The method of claim 1, further comprising decoding data representative of a size of the sub-blocks.

11. The method of claim 10, wherein decoding the data representative of the size of the sub-blocks comprises decoding the data in at least one of a slice header, a sequence parameter set (SPS), or a picture parameter set (PPS).

12. The method of claim 1, wherein deriving the motion information comprises scaling motion information for the neighboring blocks to a common reference picture.

13. The method of claim 1, wherein deriving the motion information comprises deriving a motion vector (MV) for each of the sub-blocks according to the following formula, where $w_i$ represents a horizontal weighting factor, $w_j$ represents a vertical weighting factor, $O_i$ represents a horizontal offset value, and $O_j$ represents a vertical offset value:

$$(MV_x, MV_y) = ((\sum_{i=0}^{2} MV_{xi} + sign(\sum_{i=0}^{2} MV_{xi}) * 1) * 43/128, (\sum_{i=0}^{2} MV_{yi} + sign(-\sum_{i=0}^{2} MV_{yi}) * 1) * 43/128)).$$

14. The method of claim 1, further comprising adding the motion prediction candidate to a candidate list.

15. The method of claim 1, further comprising encoding the sub-blocks prior to decoding the sub-blocks, the method further comprising encoding data that identifies the motion prediction candidate for the current block.

16. The method of claim 1, the method being executable on a wireless communication device, wherein the device comprises:

a memory configured to store the video data;

a processor configured to execute instructions to process the video data stored in the memory; and

a receiver configured to receive an encoded version of the video data.

17. The method of claim 16, wherein the wireless communication device is a cellular telephone and the encoded video data is received by the receiver and modulated according to a cellular communication standard.

18. A device for decoding video data, the device comprising:
a memory configured to store video data; and
a video decoder configured to:
determine that a motion prediction candidate for a current block of the video data indicates that motion information is to be derived for sub-blocks of the current block;
in response to the determination:
partition the current block into the sub-blocks;
for each of the sub-blocks, derive motion information using motion information for at least two neighboring blocks; and
decode the sub-blocks using the respective derived motion information.

19. The device of claim 18, wherein to derive the motion information, the video decoder is configured to, for each of the sub-blocks, derive the motion information using an above-neighboring sub-block, a left-neighboring sub-block, and a temporally neighboring sub-block.

20. The device of claim 19,
wherein the above-neighboring sub-block comprises an above-neighboring sub-block within the current block or outside the current block, and
wherein the left-neighboring sub-block comprises a left-neighboring sub-block within the current block or outside the current block.

21. The device of claim 18, wherein the video decoder is further configured to decode data representative of a size of the sub-blocks from at least one of a slice header, a sequence parameter set (SPS), or a picture parameter set (PPS).

22. The device of claim 18, wherein the video decoder is further configured to scale motion information for the neighboring blocks to a common reference picture.

23. The device of claim 18, further comprising a video encoder configured to encode the sub-blocks before the video decoder decodes the sub-blocks.

24. The device of claim 18, wherein the device is a wireless communication device, further comprising:

a receiver configured to receive an encoded version of the video data.

25. The device of claim 24, wherein the wireless communication device is a cellular telephone and the encoded video data is received by the receiver and modulated according to a cellular communication standard.

26. A device for decoding video data, the device comprising:

means for determining that a motion prediction candidate for a current block of video data indicates that motion information is to be derived for sub-blocks of the current block;

means for partitioning the current block into the sub-blocks in response to the determination;

means for deriving, for each of the sub-blocks, motion information using motion information for at least two neighboring blocks in response to the determination; and

means for decoding the sub-blocks using the respective derived motion information in response to the determination.

27. The device of claim 26, wherein the means for deriving the motion information comprises means for deriving, for each of the sub-blocks, the motion information using an above-neighboring sub-block, a left-neighboring sub-block, and a temporally neighboring sub-block.

28. The device of claim 27,

wherein the above-neighboring sub-block comprises an above-neighboring sub-block within the current block or outside the current block, and

wherein the left-neighboring sub-block comprises a left-neighboring sub-block within the current block or outside the current block.

29. The device of claim 26, further comprising means for encoding the sub-blocks before the means for decoding decodes the sub-blocks.

30. A computer-readable storage medium having stored thereon instructions that, when executed, cause a processor to:

determine that a motion prediction candidate for a current block of video data indicates that motion information is to be derived for sub-blocks of the current block;

in response to the determination:

partition the current block into the sub-blocks;

for each of the sub-blocks, derive motion information using motion information for at least two neighboring blocks; and

decode the sub-blocks using the respective derived motion information.

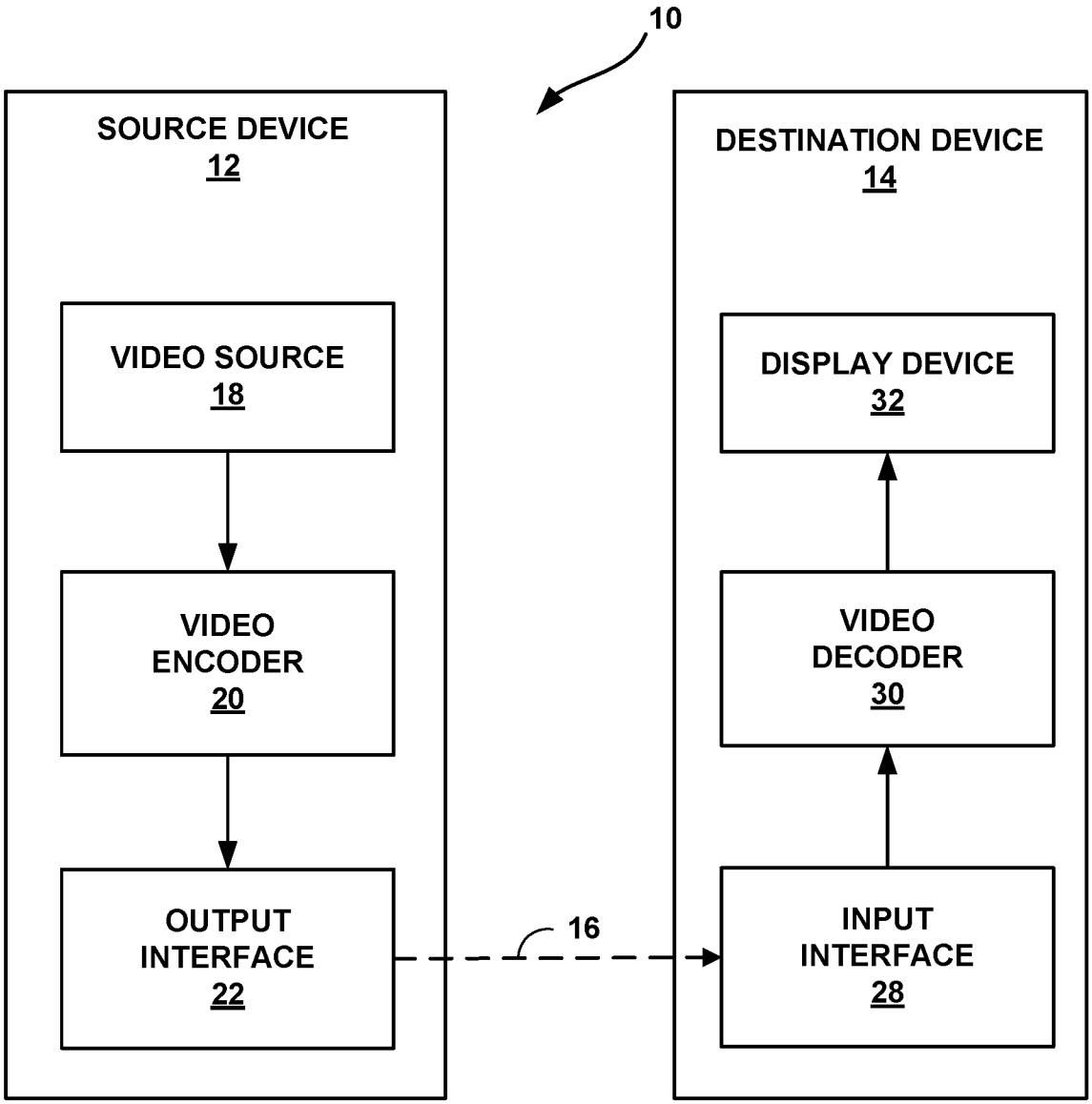
10
SOURCE DEVICE
12
VIDEO SOURCE
18
VIDEO
ENCODER
20
OUTPUT
INTERFACE
22
16
DESTINATION DEVICE
14
DISPLAY DEVICE
32
VIDEO
DECODER
30
INPUT
INTERFACE
28

FIG. 1

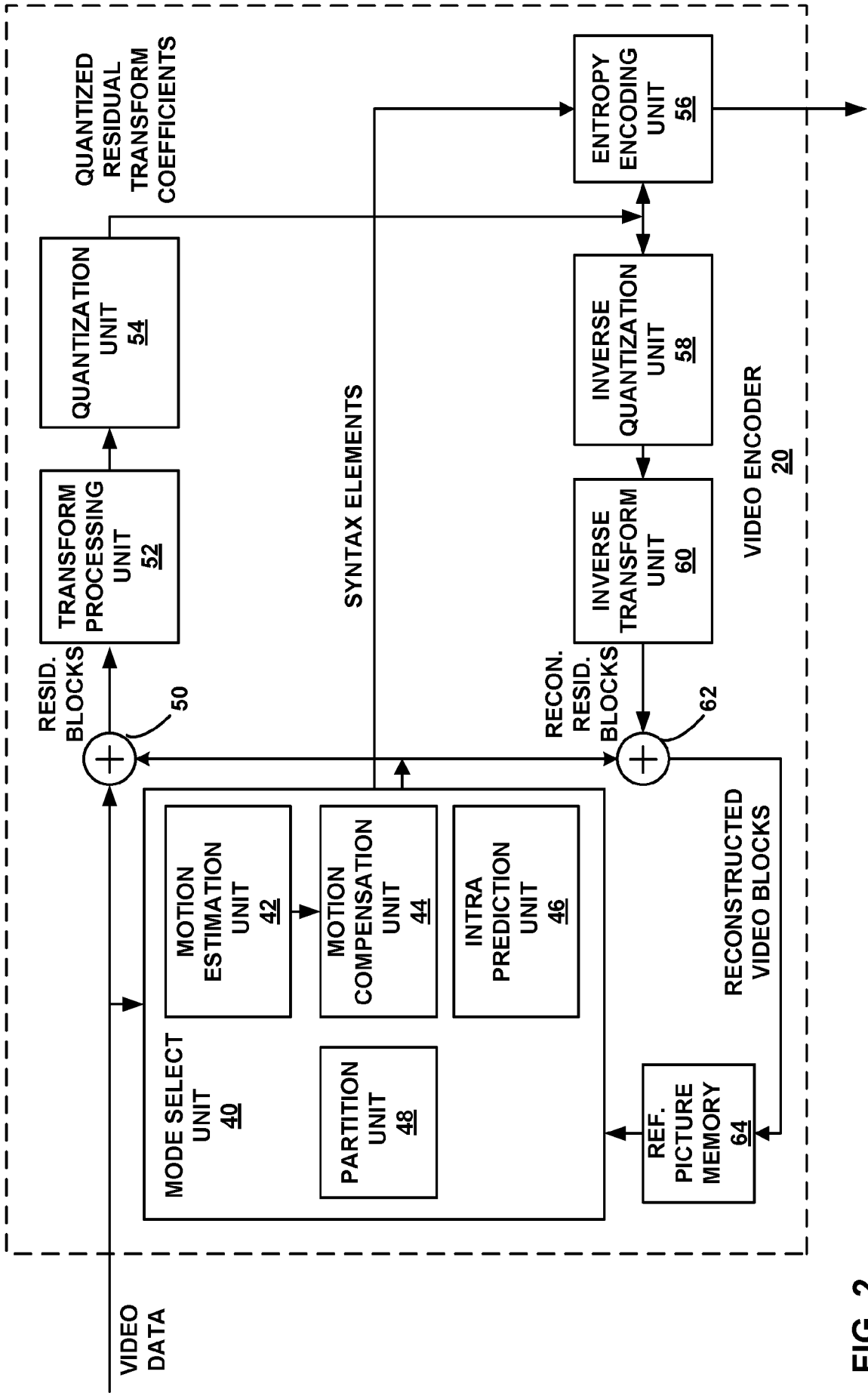
VIDEO DATA
VIDEO ENCODER 20
MODE SELECT UNIT 40
MOTION ESTIMATION UNIT 42
MOTION COMPENSATION UNIT 44
INTRA PREDICTION UNIT 46
PARTITION UNIT 48
REF. PICTURE MEMORY 64
RESID. BLOCKS
50
TRANSFORM PROCESSING UNIT 52
QUANTIZATION UNIT 54
QUANTIZED RESIDUAL TRANSFORM COEFFICIENTS
SYNTAX ELEMENTS
ENTROPY ENCODING UNIT 56
INVERSE QUANTIZATION UNIT 58
INVERSE TRANSFORM UNIT 60
RECON. RESID. BLOCKS
62
RECONSTRUCTED VIDEO BLOCKS

FIG. 2

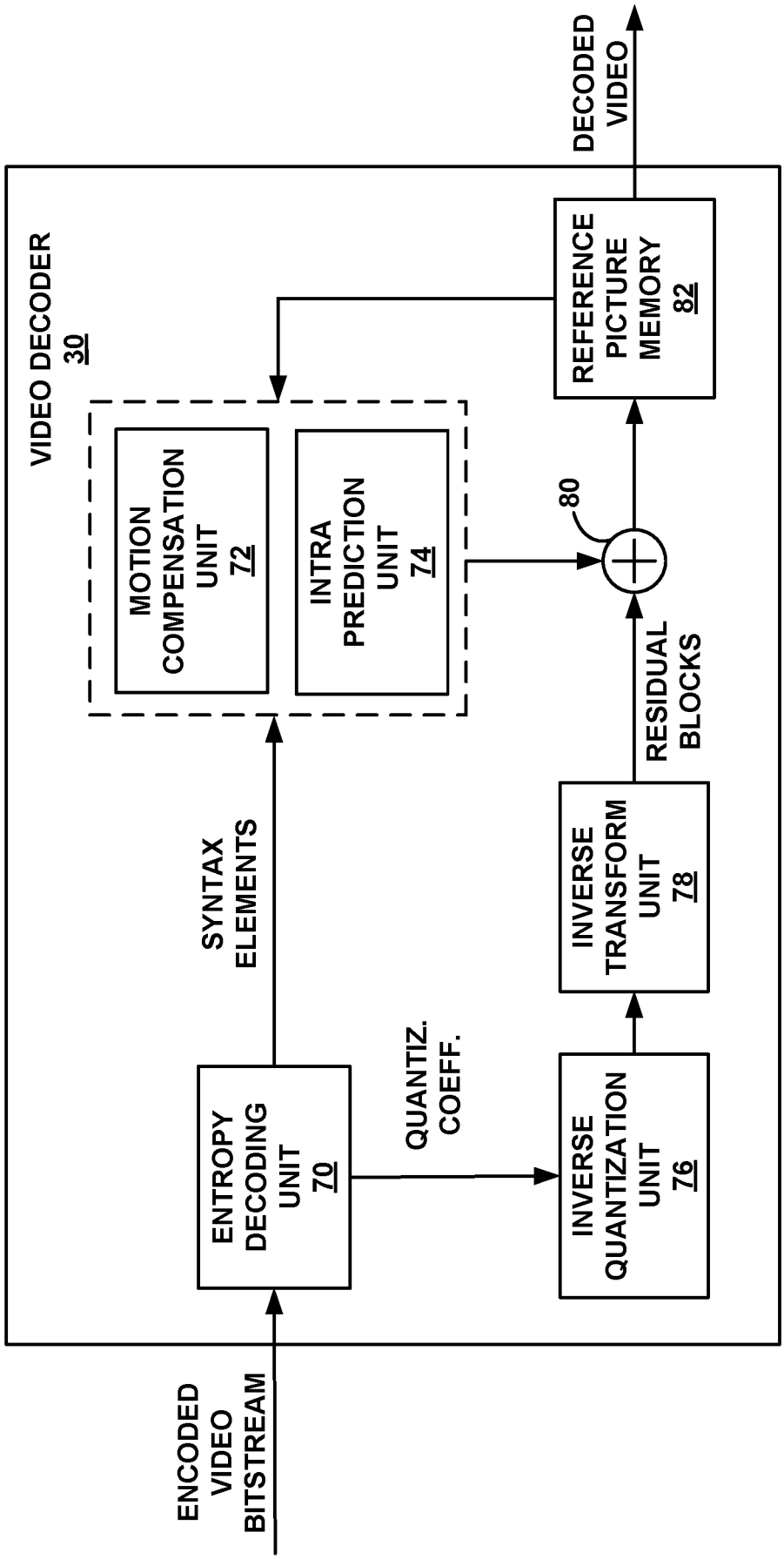
DECODED VIDEO
VIDEO DECODER 30
REFERENCE PICTURE MEMORY 82
MOTION COMPENSATION UNIT 72
INTRA PREDICTION UNIT 74
80
RESIDUAL BLOCKS
INVERSE TRANSFORM UNIT 78
SYNTAX ELEMENTS
QUANTIZ. COEFF.
ENTROPY DECODING UNIT 70
INVERSE QUANTIZATION UNIT 76
ENCODED VIDEO BITSTREAM

FIG. 3

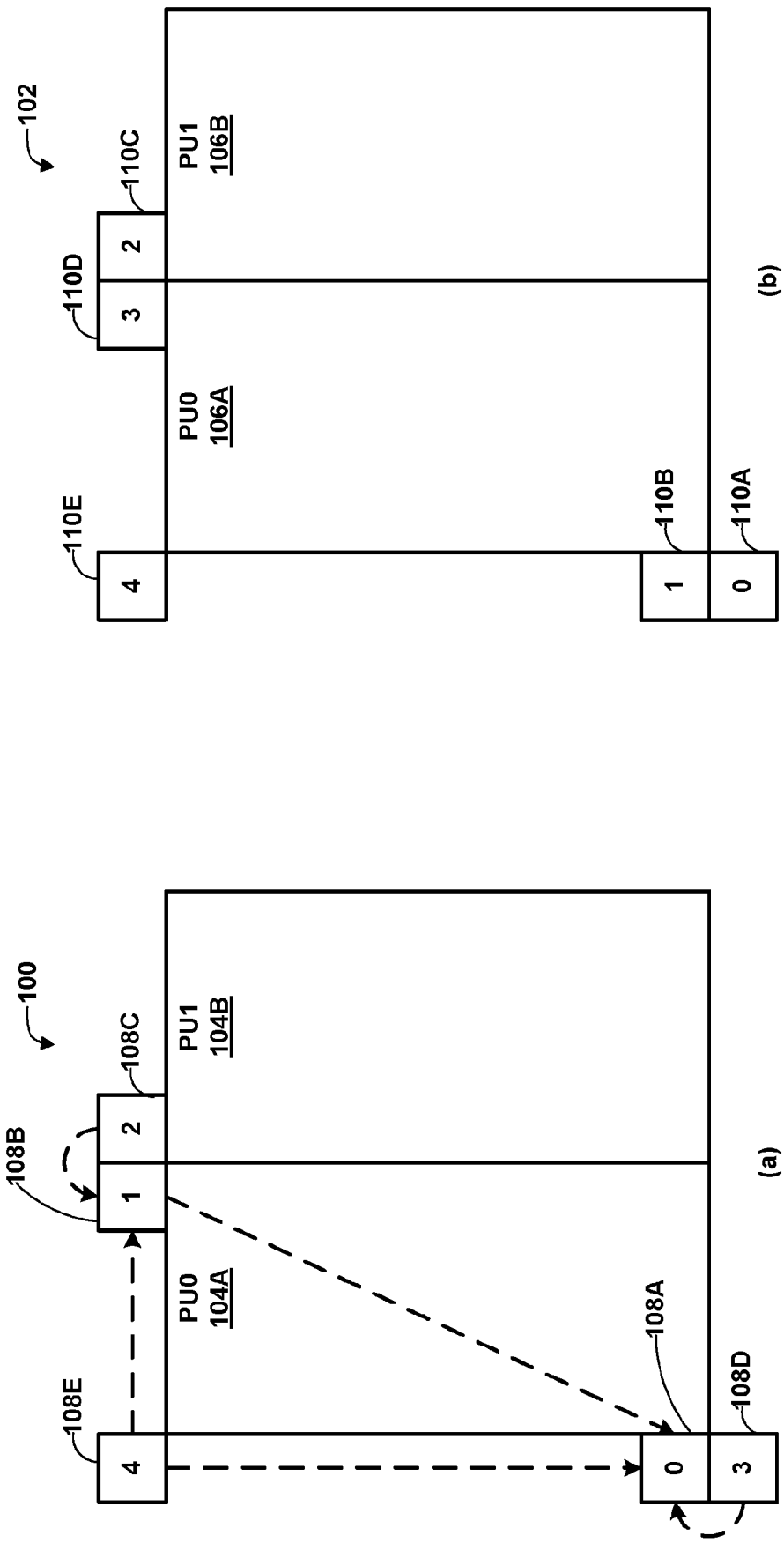
102
110C
110D
110E
PU1
106B
PU0
106A
110B
110A
(b)
100
108C
108B
108E
PU1
104B
PU0
104A
108A
108D
(a)

FIG. 4

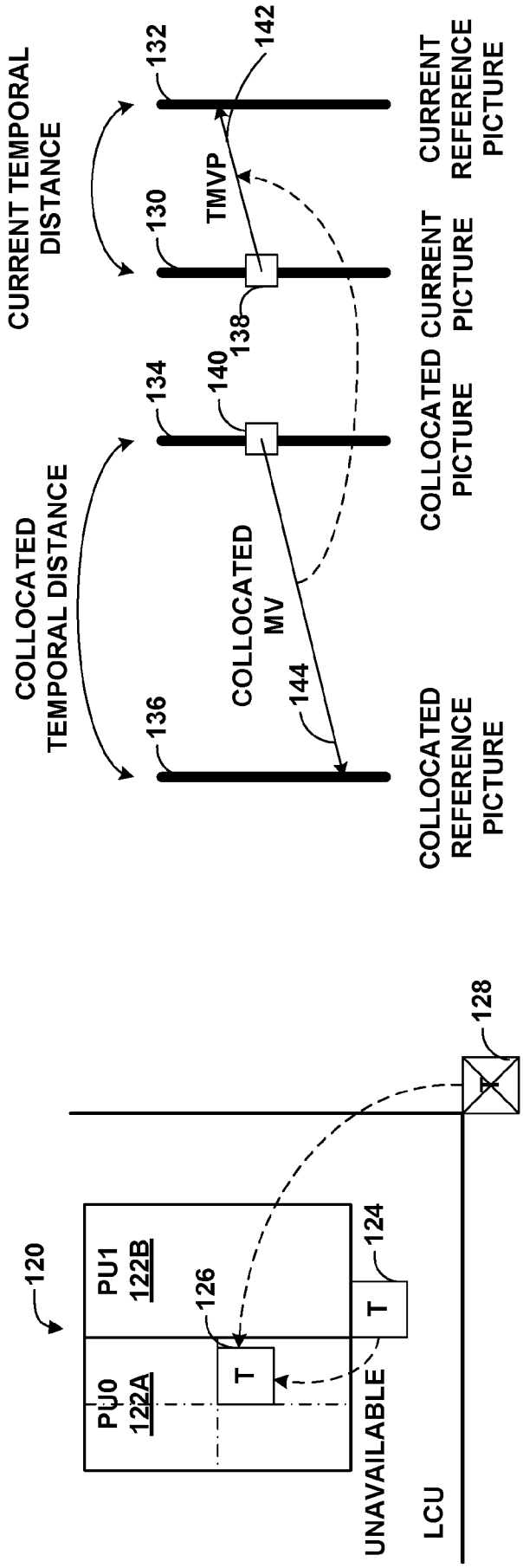
CURRENT TEMPORAL DISTANCE
132
142
TMVP
130
CURRENT REFERENCE PICTURE
138
CURRENT PICTURE
134
140
COLLOCATED PICTURE
COLLOCATED TEMPORAL DISTANCE
COLLOCATED MV
144
136
COLLOCATED REFERENCE PICTURE
(b)
120
PU1 122B
126
124
T
T
PU0 122A
UNAVAILABLE
LCU
128
(a)

FIG. 5

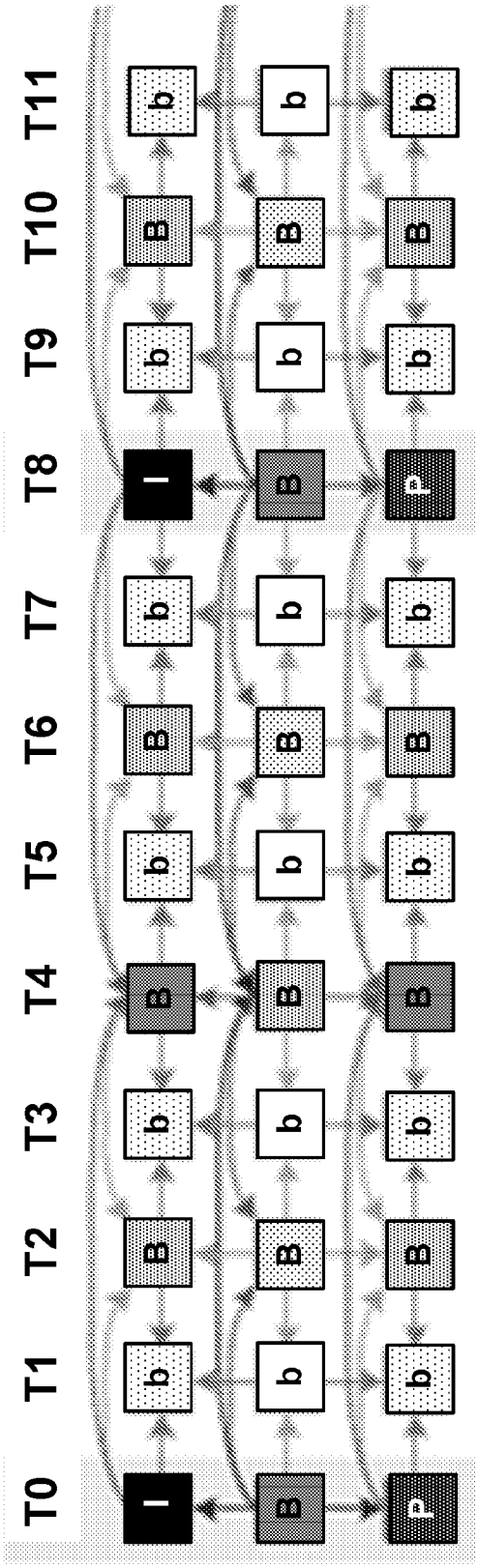


FIG. 6

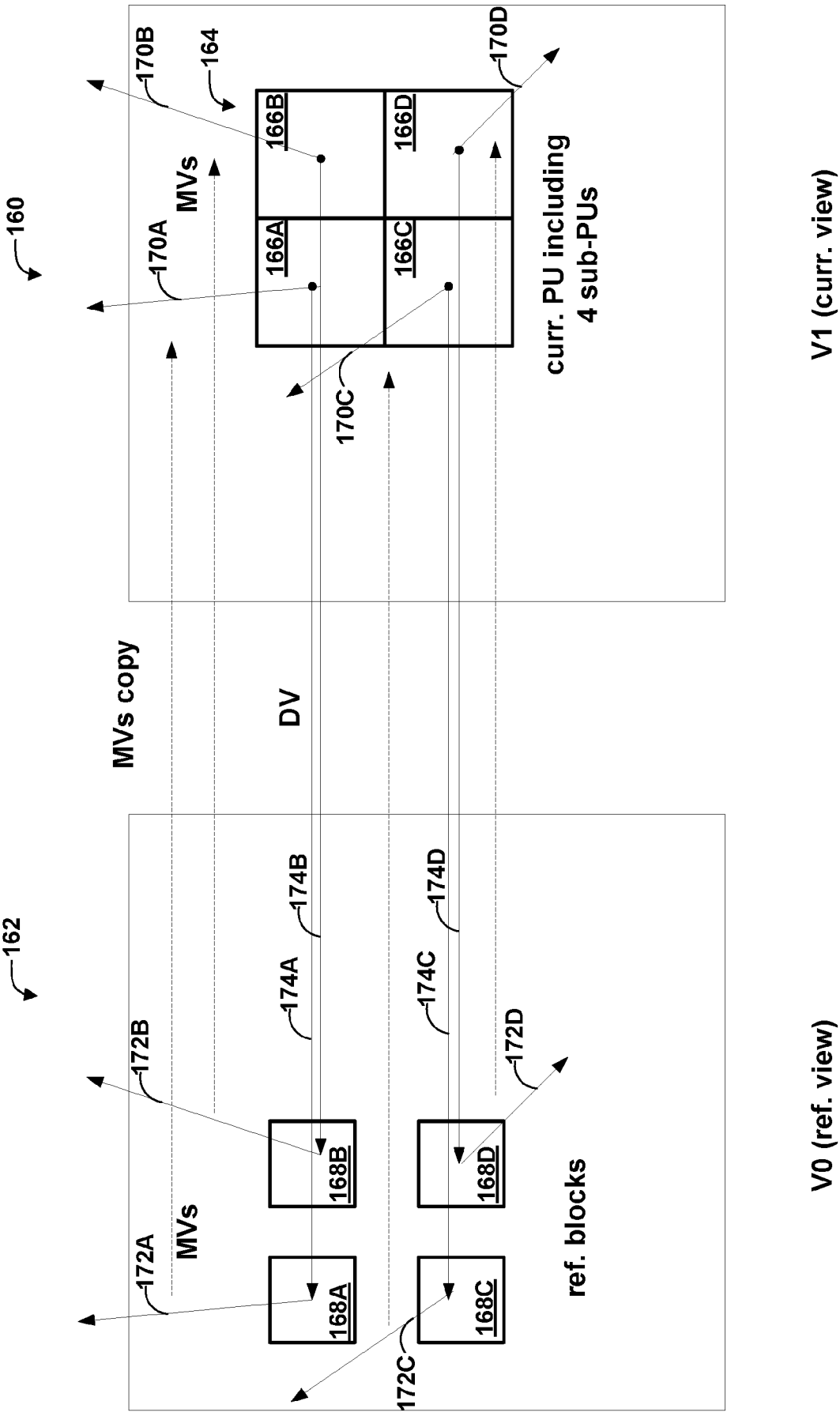
160
162
164
170A
170B
170C
170D
MVs
MVs copy
DV
curr. PU including
4 sub-PUs
166A
166B
166C
166D
V1 (curr. view)
V0 (ref. view)
ref. blocks
172A
172B
172C
172D
174A
174B
174C
174D
168A
168B
168C
168D

FIG. 7

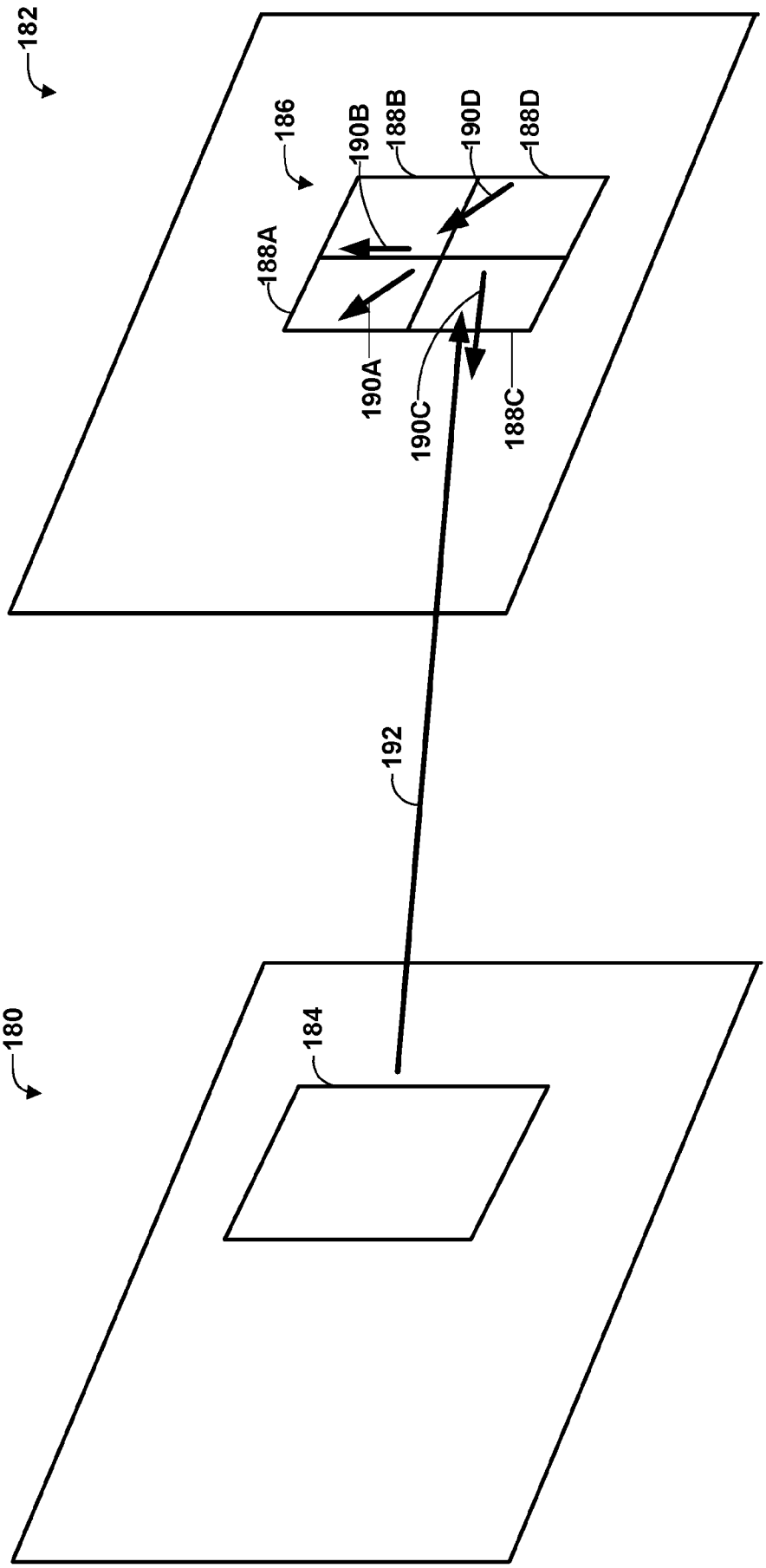


FIG. 8

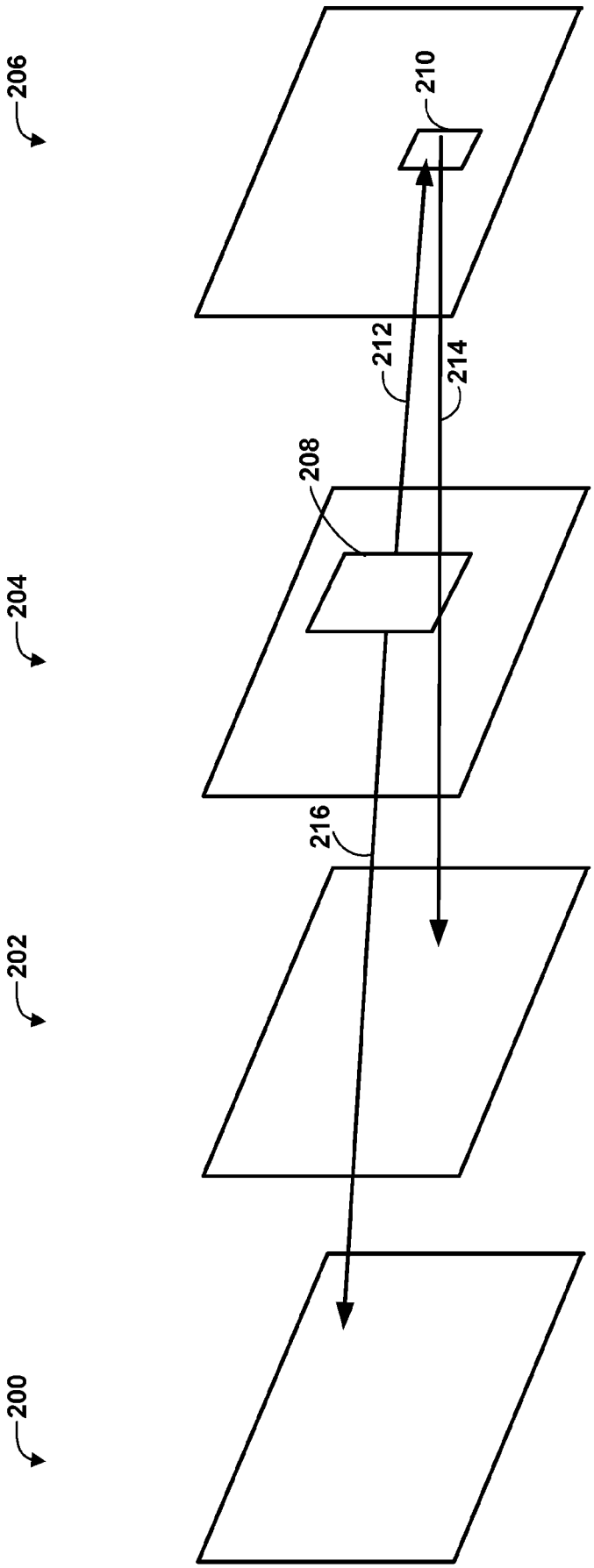


FIG. 9

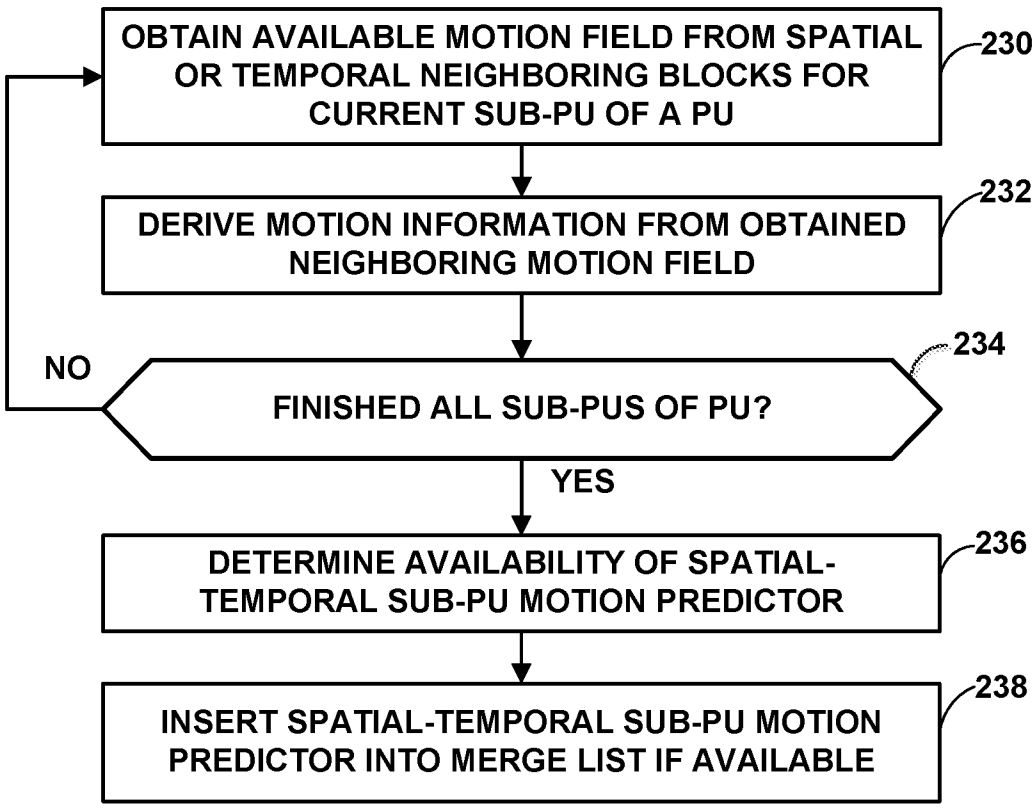
OBTAIN AVAILABLE MOTION FIELD FROM SPATIAL OR TEMPORAL NEIGHBORING BLOCKS FOR CURRENT SUB-PU OF A PU
230
DERIVE MOTION INFORMATION FROM OBTAINED NEIGHBORING MOTION FIELD
232
NO
FINISHED ALL SUB-PUS OF PU?
234
YES
DETERMINE AVAILABILITY OF SPATIAL-TEMPORAL SUB-PU MOTION PREDICTOR
236
INSERT SPATIAL-TEMPORAL SUB-PU MOTION PREDICTOR INTO MERGE LIST IF AVAILABLE
238

FIG. 10

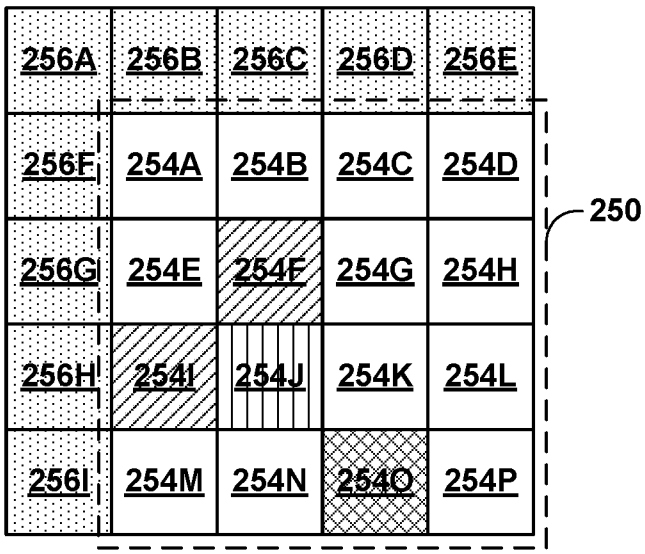


FIG. 11A

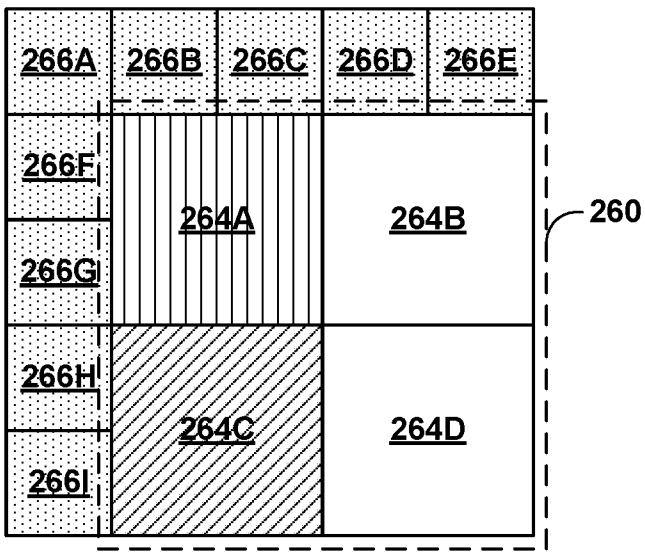


FIG. 11B

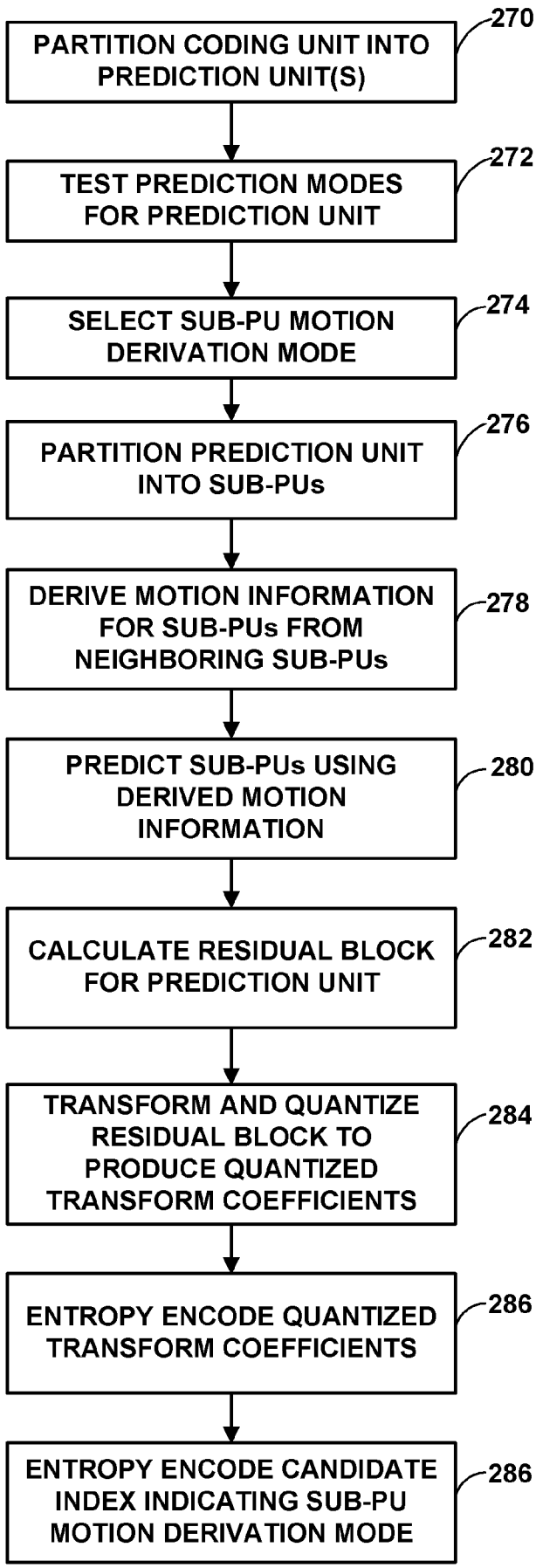
PARTITION CODING UNIT INTO PREDICTION UNIT(S)
270
TEST PREDICTION MODES FOR PREDICTION UNIT
272
SELECT SUB-PU MOTION DERIVATION MODE
274
PARTITION PREDICTION UNIT INTO SUB-PUs
276
DERIVE MOTION INFORMATION FOR SUB-PUs FROM NEIGHBORING SUB-PUs
278
PREDICT SUB-PUs USING DERIVED MOTION INFORMATION
280
CALCULATE RESIDUAL BLOCK FOR PREDICTION UNIT
282
TRANSFORM AND QUANTIZE RESIDUAL BLOCK TO PRODUCE QUANTIZED TRANSFORM COEFFICIENTS
284
ENTROPY ENCODE QUANTIZED TRANSFORM COEFFICIENTS
286
ENTROPY ENCODE CANDIDATE INDEX INDICATING SUB-PU MOTION DERIVATION MODE
286

FIG. 12

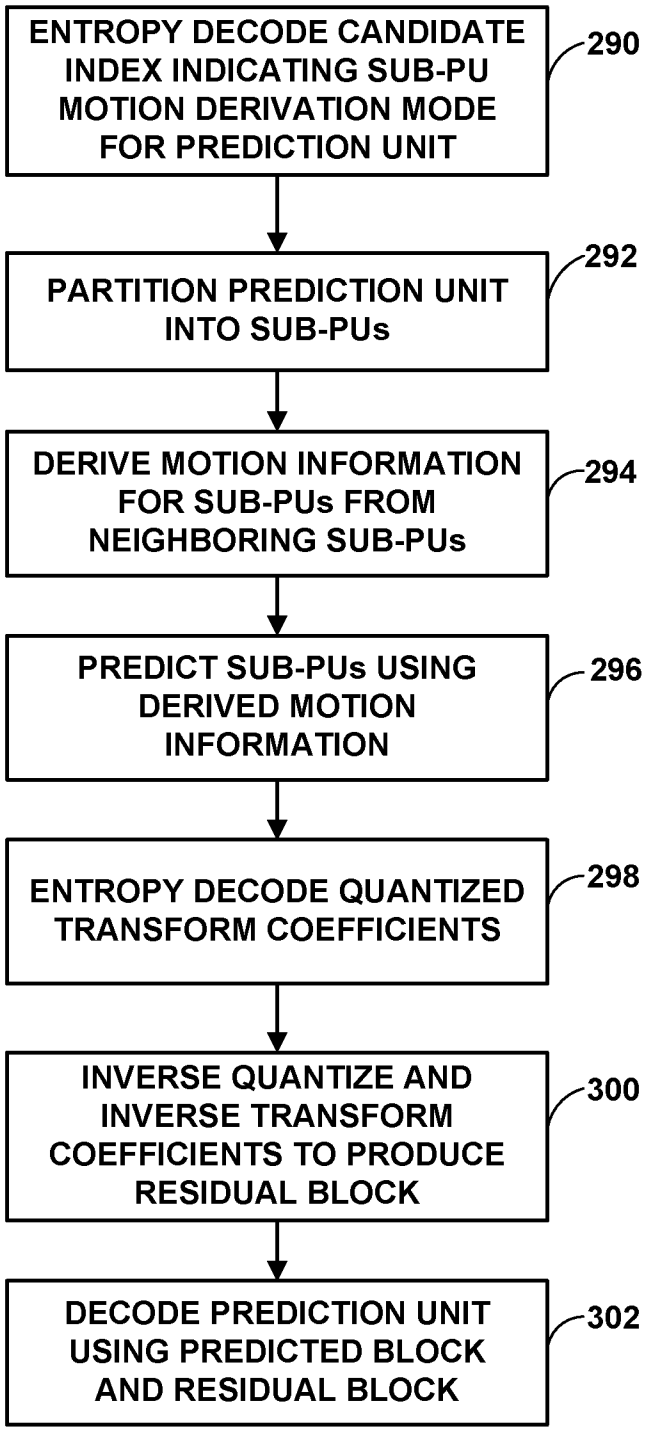
ENTROPY DECODE CANDIDATE INDEX INDICATING SUB-PU MOTION DERIVATION MODE FOR PREDICTION UNIT
290
PARTITION PREDICTION UNIT INTO SUB-PUs
292
DERIVE MOTION INFORMATION FOR SUB-PUs FROM NEIGHBORING SUB-PUs
294
PREDICT SUB-PUs USING DERIVED MOTION INFORMATION
296
ENTROPY DECODE QUANTIZED TRANSFORM COEFFICIENTS
298
INVERSE QUANTIZE AND INVERSE TRANSFORM COEFFICIENTS TO PRODUCE RESIDUAL BLOCK
300
DECODE PREDICTION UNIT USING PREDICTED BLOCK AND RESIDUAL BLOCK
302

FIG. 13