# INTERSECTING MOTION VECTOR PREDICTION IN VIDEO CODING

**[0001]** This application claims the priority to U.S. Patent Application No. 19/173,166, filed April 8, 2025 and U.S. Provisional Patent Application No. 63/631,718, filed April 9, 2024, the entire content of each of which are incorporated by reference. U.S. Patent Application No. 19/173,166, filed April 8, 2025 claims the benefit of U.S. Provisional Patent Application No. 63/631,718, filed April 9, 2024.

## TECHNICAL FIELD

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

## BACKGROUND

**[0003]** Digital video capabilities can be incorporated into a wide range of devices, including digital televisions, digital direct broadcast systems, wireless broadcast systems, personal digital assistants (PDAs), laptop or desktop computers, tablet computers, e-book readers, digital cameras, digital recording devices, digital media players, video gaming devices, video game consoles, cellular or satellite radio telephones, so-called "smart phones," video teleconferencing devices, video streaming devices, and the like. Digital video devices implement video coding techniques, such as those described in the standards defined by MPEG-2, MPEG-4, ITU-T H.263, ITU-T H.264/MPEG-4, Part 10, Advanced Video Coding (AVC), ITU-T H.265/High Efficiency Video Coding (HEVC), ITU-T H.266/Versatile Video Coding (VVC), and extensions of such standards, as well as proprietary video codecs/formats such as AOMedia Video 1 (AV1) that was developed by the Alliance for Open Media. The video devices may transmit, receive, encode, decode, and/or store digital video information more efficiently by implementing such video coding techniques.

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

B) slice of a picture may use spatial prediction with respect to reference samples in neighboring blocks in the same picture or temporal prediction with respect to reference samples in other reference pictures. Pictures may be referred to as frames, and reference pictures may be referred to as reference frames.

## SUMMARY

**[0005]** In general, this disclosure describes techniques for motion vector prediction and reference picture derivation. In some current implementations and draft standards, a motion vector predictor (MVP) can be derived from a spatial neighbor, a temporal neighbor with scaling, a constructive motion vector with average of two motion vector, or a zero motion vector. However, such current techniques of deriving an MVP are not efficient for all situations. For example, in some cases, an MVP candidate is derived from a constructive motion vector or a zero motion vector, which is not efficient to model the temporal information for an inter predicted block.

**[0006]** The techniques of this disclosure consider a motion vector using both a spatial and a temporal neighbor, and/or a motion vector that is from an already decoded picture. The techniques of this disclosure may improve MVP derivation, resulting in more bandwidth efficient encoding and/or higher quality video decoding. For example, the techniques of this disclosure may provide new MVPs which may not exist with current techniques. Such new MVPs may provide a better predictor for coding a block of video data, thereby improving coding efficiency and/or quality. Such techniques may include the use of intersecting motion vectors, temporal motion vector prediction using a spatial neighbor motion vector, reference picture derivation techniques, and the adding of a motion vector predictor having a different reference picture.

**[0007]** In one example, a method includes: determining, for each of a plurality of subblocks of a current block of a current picture of the video data, a corresponding intersecting motion vector list, each corresponding intersecting motion vector list comprising at least one intersecting motion vector, the at least one intersecting motion vector comprising a source motion vector associated with a source block located in a first reference picture and a destination motion vector associated with a destination block located in a second reference picture, and wherein the source block has a predictor that is a subset of the destination block, the first reference picture being already decoded and temporally prior to the current picture and the second reference picture being already

decoded and temporally after the current picture; determining an intersecting motion vector for the current block based on the corresponding intersecting motion vector lists; and decoding the current block based on the intersecting motion vector.

**[0008]** In one example, a device includes: one or more memories for storing the video data; and one or more processors operatively coupled to the one or more memories, the one or more processors configured to: determine, for each of a plurality of subblocks of a current block of a current picture of the video data, a corresponding intersecting motion vector list, each corresponding intersecting motion vector list comprising at least one intersecting motion vector, the at least one intersecting motion vector comprising a source motion vector associated with a source block located in a first reference picture and a destination motion vector associated with a destination block located in a second reference picture, and wherein the source block has a predictor that is a subset of the destination block, the first reference picture being already decoded and temporally prior to the current picture and the second reference picture being already decoded and temporally after the current picture; determine an intersecting motion vector for the current block based on the corresponding intersecting motion vector lists; and decode the current block based on the intersecting motion vector.

**[0009]** In one example, a method includes: determining, for each of a plurality of subblocks of a current block of a current picture of the video data, a corresponding intersecting motion vector list, each corresponding intersecting motion vector list comprising at least one intersecting motion vector, the at least one intersecting motion vector comprising a source motion vector associated with a source block located in a first reference picture and a destination motion vector associated with a destination block located in a second reference picture, and wherein the source block has a predictor that is a subset of a destination block, the first reference picture being already decoded and temporally prior to the current picture and the second reference picture being already decoded and temporally after the current picture; determining an intersecting motion vector for the current block in accordance with the corresponding intersecting motion vector lists; and encoding the current block based on the intersecting motion vector.

**[0010]** In one example, a device includes: one or more memories for storing the video data; and one or more processors operatively coupled to the one or more memories, the one or more processors configured to: determine, for each of a plurality of subblocks of a current block of a current picture of video data, a corresponding intersecting motion vector list, each corresponding intersecting motion vector list comprising at least one

intersecting motion vector, the at least one intersecting motion vector comprising a source motion vector associated with a source block located in a first reference picture and a destination motion vector associated with a destination block located in a second reference picture, and wherein the source block has a predictor that is a subset of a destination block, the first reference picture being already decoded and temporally prior to the current picture and the second reference picture being already decoded and temporally after the current picture; determine an intersecting motion vector for the current block in accordance with the corresponding intersecting motion vector lists; and encode the current block based on the intersecting motion vector.

**[0011]** In one example, a method includes: determining a current reference picture for a current block of a current picture of the video data based on a reference picture of a collocated block, a reference picture list of the current block, and a picture order count (POC) of a collocated picture, wherein the collocated block is collocated with the current block and is in the collocated picture; and decoding the current block based on a motion vector associated with the current reference picture.

**[0012]** In one example, a device includes: one or more memories for storing the video data; and one or more processors operatively coupled to the one or more memories, the one or more processors configured to: determine a current reference picture for a current block of a current picture of the video data based on a reference picture of a collocated block, a reference picture list of the current block, and a picture order count (POC) of a collocated picture, wherein the collocated block is collocated with the current block and is in the collocated picture; and decode the current block based on a motion vector associated with the current reference picture.

**[0013]** In one example, a method includes: determining a current reference picture for a current block of a current picture of the video data based on a reference picture of a collocated block, a reference picture list of the current block, and a picture order count (POC) of a collocated picture, wherein the collocated block is collocated with the current block and is in the collocated picture; and encoding the current block in accordance with a motion vector associated with the current reference picture.

**[0014]** In one example, a method includes: determining at least one intersecting motion vector for a current block of a current picture of the video data, the intersecting motion vector comprising a source motion vector associated with a first reference picture and a destination motion vector associated with a second reference picture, the first reference picture being temporally prior to the current picture and the second reference picture

being temporally after the current picture; and coding the current block based on the at least one intersecting motion vector.

**[0015]** In another example, a method includes: determining a temporal motion vector predictor for a current block of a current picture of the video data based on a collocated block of a collocated picture offset by a motion vector of a spatial neighbor block, neighboring the current block in the current picture; and coding the current block based on the temporal motion vector predictor.

**[0016]** In another example, a method includes: determining a reference picture for a current block of the video data; and coding the current block of the video data based on a motion vector associated with the reference picture.

**[0017]** In another example, a method includes: determining one or more new motion vector predictors from at least one existing motion vector predictor in a motion vector predictor list for a current block of the video data; and coding the video data based on the one or more new motion vector predictors.

**[0018]** In another example, a device includes one or more memories configured to store video data and one or more processors configured to perform any of the techniques described herein.

**[0019]** In another example, a device includes one or more means configured to perform any of the techniques described herein.

**[0020]** In another example, computer-readable storage media is encoded with instructions that, when executed, cause one or more processors to perform any of the techniques described herein.

**[0021]** 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, drawings, and claims.

## BRIEF DESCRIPTION OF DRAWINGS

**[0022]** FIG. 1 is a block diagram illustrating an example video encoding and decoding system that may perform the techniques of this disclosure.

**[0023]** FIGS. 2A-2B are conceptual diagrams illustrating example spatial neighboring MV candidates for merge and AMVP modes, respectively.

**[0024]** FIG. 3A is a conceptual diagram illustrating an example of a temporal motion vector predictor.

**[0025]** FIG. 3B is a conceptual diagram illustrating an example of motion vector scaling.

**[0026]** FIG. 4 is a conceptual diagram illustrating example temporal motion information derivation.

**[0027]** FIG. 5 is a conceptual diagram illustrating an example of template matching performed on a search area around an initial MV.

**[0028]** FIG. 6 is a conceptual diagram illustrating an example of motion vector difference (MVD)0 and MVD1 that are proportional based on temporal distances.

**[0029]** FIG. 7 is a conceptual diagram illustrating an example of MVD0 and MVD1 are mirrored regardless of the temporal distances.

**[0030]** FIG. 8 is a conceptual diagram illustrating an example of the 3×3 square search pattern in the search range [-8, 8].

**[0031]** FIG. 9 is a conceptual diagram illustrating an example of decoder-side motion vector refinement.

**[0032]** FIG. 10 is a conceptual diagram illustrating example diamond regions in a search area.

**[0033]** FIG. 11 is a conceptual diagram illustrating an example of control point motion vector inheritance.

**[0034]** FIG. 12 is a conceptual diagram illustrating example locations of candidate positions for constructed affine merge mode.

**[0035]** FIG. 13 is a conceptual diagram illustrating an example of searching for non-adjacent affine CU and use motion information to derive non-refined candidates.

**[0036]** FIG. 14 is a conceptual diagram illustrating an example of sub-block motion information used to derive refined candidates.

**[0037]** FIG. 15 is a flowchart illustrating an example of AMVP-merge mode for non-LDC picture techniques.

**[0038]** FIG. 16 is a conceptual diagram illustrating an example of an intersecting MV.

**[0039]** FIG. 17 is a conceptual diagram illustrating an example of a 16×8 block having IctList0 to IctList7.

**[0040]** FIGS. 18A-18B are flowcharts illustrating example techniques for intersecting motion vector prediction according to one or more aspects of this disclosure.

**[0041]** FIGS. 19A-19B are flowcharts illustrating example techniques for reference picture derivation according to one or more aspects of this disclosure.

**[0042]** FIG. 20 is a block diagram illustrating an example video encoder that may perform the techniques of this disclosure.

**[0043]** FIG. 21 is a block diagram illustrating an example video decoder that may perform the techniques of this disclosure.

**[0044]** FIG. 22 is a flowchart illustrating an example method for encoding a current block in accordance with the techniques of this disclosure.

**[0045]** FIG. 23 is a flowchart illustrating an example method for decoding a current block in accordance with the techniques of this disclosure.

## DETAILED DESCRIPTION

**[0046]** In general, this disclosure describes techniques for motion vector prediction and reference picture derivation. In some current implementations and draft standards, a motion vector predictor (MVP) can be derived from a spatial neighbor, a temporal neighbor with scaling, a constructive motion vector with average of two motion vector, or a zero motion vector. However, such current techniques of deriving an MVP are not efficient for all situations. For example, in some cases, an MVP candidate is derived from a constructive motion vector or a zero motion vector, which is not efficient to model the temporal information for an inter predicted block.

**[0047]** The techniques of this disclosure consider a motion vector using both a spatial and a temporal neighbor, and/or a motion vector that is from an already decoded picture. The techniques of this disclosure may improve MVP derivation, resulting in more bandwidth efficient encoding and/or higher quality video decoding. Such techniques may include the use of intersecting motion vectors, temporal motion vector prediction using a spatial neighbor motion vector, reference picture derivation techniques, and the adding of a motion vector predictor having a different reference picture.

**[0048]** FIG. 1 is a block diagram illustrating an example video encoding and decoding system 100 that may perform the techniques of this disclosure. The techniques of this disclosure are generally directed to coding (encoding and/or decoding) video data. In general, video data includes any data for processing a video. Thus, video data may include raw, unencoded video, encoded video, decoded (e.g., reconstructed) video, and video metadata, such as signaling data.

**[0049]** As shown in FIG. 1, system 100 includes a source device 102 that provides encoded video data to be decoded and displayed by a destination device 116, in this example. In particular, source device 102 provides the video data to destination device 116 via a computer-readable medium 110. Source device 102 and destination device 116

may be or include any of a wide range of devices, such as desktop computers, notebook (i.e., laptop) computers, mobile devices, tablet computers, set-top boxes, telephone handsets such as smartphones, televisions, cameras, display devices, digital media players, video gaming consoles, video streaming device, broadcast receiver devices, or the like. In some cases, source device 102 and destination device 116 may be equipped for wireless communication, and thus may be referred to as wireless communication devices.

**[0050]** In the example of FIG. 1, source device 102 includes video source 104, memory 106, video encoder 200, and output interface 108. Destination device 116 includes input interface 122, video decoder 300, memory 120, and display device 118. In accordance with this disclosure, video encoder 200 of source device 102 and video decoder 300 of destination device 116 may be configured to apply the techniques for motion vector prediction and reference picture derivation. Thus, source device 102 represents an example of a video encoding device, while destination device 116 represents an example of a video decoding device. In other examples, a source device and a destination device may include other components or arrangements. For example, source device 102 may receive video data from an external video source, such as an external camera. Likewise, destination device 116 may interface with an external display device, rather than include an integrated display device.

**[0051]** System 100 as shown in FIG. 1 is merely one example. In general, any digital video encoding and/or decoding device may perform techniques for motion vector prediction and reference picture derivation. Source device 102 and destination device 116 are merely examples of such coding devices in which source device 102 generates coded video data for transmission to destination device 116. This disclosure refers to a "coding" device as a device that performs coding (encoding and/or decoding) of data. Thus, video encoder 200 and video decoder 300 represent examples of coding devices, in particular, a video encoder and a video decoder, respectively. In some examples, source device 102 and destination device 116 may operate in a substantially symmetrical manner such that each of source device 102 and destination device 116 includes video encoding and decoding components. Hence, system 100 may support one-way or two-way video transmission between source device 102 and destination device 116, e.g., for video streaming, video playback, video broadcasting, or video telephony.

**[0052]** In general, video source 104 represents a source of video data (i.e., raw, unencoded video data) and provides a sequential series of pictures (also referred to as "frames") of

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

**[0053]** Memory 106 of source device 102 and memory 120 of destination device 116 represent general purpose memories. In some examples, memories 106, 120 may store raw video data, e.g., raw video from video source 104 and raw, decoded video data from video decoder 300. Additionally, or alternatively, memories 106, 120 may store software instructions executable by, e.g., video encoder 200 and video decoder 300, respectively. Although memory 106 and memory 120 are shown separately from video encoder 200 and video decoder 300 in this example, it should be understood that video encoder 200 and video decoder 300 may also include internal memories for functionally similar or equivalent purposes. Furthermore, memories 106, 120 may store encoded video data, e.g., output from video encoder 200 and input to video decoder 300. In some examples, portions of memories 106, 120 may be allocated as one or more video buffers, e.g., to store raw, decoded, and/or encoded video data.

**[0054]** Computer-readable medium 110 may represent any type of medium or device capable of transporting the encoded video data from source device 102 to destination device 116. In one example, computer-readable medium 110 represents a communication medium to enable source device 102 to transmit encoded video data directly to destination device 116 in real-time, e.g., via a radio frequency network or computer-based network. Output interface 108 may modulate a transmission signal including the encoded video data, and input interface 122 may demodulate the received transmission signal, according to a communication standard, such as a wireless communication protocol. The communication medium may include any wireless or wired communication medium, such as a radio frequency (RF) spectrum or one or more physical transmission lines. The

communication medium may form part of a packet-based network, such as a local area network, a wide-area network, or a global network such as the Internet. The communication medium may include routers, switches, base stations, or any other equipment that may be useful to facilitate communication from source device 102 to destination device 116.

**[0055]** In some examples, source device 102 may output encoded data from output interface 108 to storage device 112. Similarly, destination device 116 may access encoded data from storage device 112 via input interface 122. Storage device 112 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.

**[0056]** In some examples, source device 102 may output encoded video data to file server 114 or another intermediate storage device that may store the encoded video data generated by source device 102. Destination device 116 may access stored video data from file server 114 via streaming or download.

**[0057]** File server 114 may be any type of server device capable of storing encoded video data and transmitting that encoded video data to the destination device 116. File server 114 may represent a web server (e.g., for a website), a server configured to provide a file transfer protocol service (such as File Transfer Protocol (FTP) or File Delivery over Unidirectional Transport (FLUTE) protocol), a content delivery network (CDN) device, a hypertext transfer protocol (HTTP) server, a Multimedia Broadcast Multicast Service (MBMS) or Enhanced MBMS (eMBMS) server, and/or a network attached storage (NAS) device. File server 114 may, additionally or alternatively, implement one or more HTTP streaming protocols, such as Dynamic Adaptive Streaming over HTTP (DASH), HTTP Live Streaming (HLS), Real Time Streaming Protocol (RTSP), HTTP Dynamic Streaming, or the like.

**[0058]** Destination device 116 may access encoded video data from file server 114 through any standard data connection, including an Internet connection. This may include a wireless channel (e.g., a Wi-Fi connection), a wired connection (e.g., digital subscriber line (DSL), cable modem, etc.), or a combination of both, that is suitable for accessing encoded video data stored on file server 114. Input interface 122 may be configured to operate according to any one or more of the various protocols discussed above for retrieving or receiving media data from file server 114, or other such protocols for retrieving media data.

**[0059]** Output interface 108 and input interface 122 may represent wireless transmitters/receivers, modems, wired networking components (e.g., Ethernet cards), wireless communication components that operate according to any of a variety of IEEE 802.11 standards, or other physical components. In examples where output interface 108 and input interface 122 include wireless components, output interface 108 and input interface 122 may be configured to transfer data, such as encoded video data, according to a cellular communication standard, such as 4G, 4G-LTE (Long-Term Evolution), LTE Advanced, 5G, or the like. In some examples where output interface 108 includes a wireless transmitter, output interface 108 and input interface 122 may be configured to transfer data, such as encoded video data, according to other wireless standards, such as an IEEE 802.11 specification, an IEEE 802.15 specification (e.g., ZigBee™), a Bluetooth™ standard, or the like. In some examples, source device 102 and/or destination device 116 may include respective system-on-a-chip (SoC) devices. For example, source device 102 may include an SoC device to perform the functionality attributed to video encoder 200 and/or output interface 108, and destination device 116 may include an SoC device to perform the functionality attributed to video decoder 300 and/or input interface 122.

**[0060]** The techniques of this disclosure may be applied to video coding in support of any of a variety of multimedia applications, such as over-the-air television broadcasts, cable television transmissions, satellite television transmissions, Internet streaming video transmissions, such as dynamic adaptive streaming over HTTP (DASH), digital video that is encoded onto a data storage medium, decoding of digital video stored on a data storage medium, or other applications.

**[0061]** Input interface 122 of destination device 116 receives an encoded video bitstream from computer-readable medium 110 (e.g., a communication medium, storage device 112, file server 114, or the like). The encoded video bitstream may include signaling information defined by video encoder 200, which is also used by video decoder 300, such as syntax elements having values that describe characteristics and/or processing of video blocks or other coded units (e.g., slices, pictures, groups of pictures, sequences, or the like). Display device 118 displays decoded pictures of the decoded video data to a user. Display device 118 may represent any of a variety of display devices such as a liquid crystal display (LCD), a plasma display, an organic light emitting diode (OLED) display, or another type of display device.

**[0062]** Although not shown in FIG. 1, in some examples, video encoder 200 and video decoder 300 may each be integrated with an audio encoder and/or audio decoder (e.g., audio codec), and may include appropriate MUX-DEMUX units, or other hardware and/or software, to handle multiplexed streams including both audio and video in a common data stream. Example audio codecs may include AAC, AC-3, AC-4, ALAC, ALS, AMBE, AMR, AMR-WB (G.722.2), AMR-WB+, aptx (various versions), ATRAC, BroadVoice (BV16, BV32), CELT, Enhanced AC-3 (E-AC-3), EVS, FLAC, G.711, G.722, G.722.1, G.722.2 (AMR-WB). G.723.1, G.726, G.728, G.729, G.729.1, GSM-FR, HE-AAC, iLBC, iSAC, LA Lyra, Monkey's Audio, MP1, MP2 (MPEG-1, 2 Audio Layer II), MP3, Musepack, Nellymoser Asao, OptimFROG, Opus, Sac, Satin, SBC, SILK, Siren 7, Speex, SVOPC, True Audio (TTA), TwinVQ, USAC, Vorbis (Ogg), WavPack, and Windows Media Aud.

**[0063]** Video encoder 200 and video decoder 300 each may be implemented as any of a variety of suitable encoder and/or decoder circuitry that includes a processing system, such as one or more microprocessors, digital signal processors (DSPs), application specific integrated circuits (ASICs), field programmable gate arrays (FPGAs), discrete logic, software, hardware, firmware or any combinations thereof. When the techniques are implemented partially in software, a device may store instructions for the software in a suitable, non-transitory computer-readable medium and execute the instructions in hardware using one or more processors to perform the techniques of this disclosure. Each of video encoder 200 and video decoder 300 may be included in one or more encoders or decoders, either of which may be integrated as part of a combined encoder/decoder (CODEC) in a respective device. A device including video encoder 200 and/or video decoder 300 may implement video encoder 200 and/or video decoder 300 in processing circuitry such as an integrated circuit and/or a microprocessor. Such a device may be a wireless communication device, such as a cellular telephone, or any other type of device described herein.

**[0064]** Video encoder 200 and video decoder 300 may operate according to a video coding standard, such as ITU-T H.265, also referred to as High Efficiency Video Coding (HEVC) or extensions thereto, such as the multi-view and/or scalable video coding extensions. Alternatively, video encoder 200 and video decoder 300 may operate according to other proprietary or industry standards, such as ITU-T H.266, also referred to as Versatile Video Coding (VVC). In other examples, video encoder 200 and video decoder 300 may operate according to a proprietary video codec/format, such as

AOMedia Video 1 (AV1), extensions of AV1, and/or successor versions of AV1 (e.g., AV2). In other examples, video encoder 200 and video decoder 300 may operate according to other proprietary formats or industry standards. The techniques of this disclosure, however, are not limited to any particular coding standard or format. In general, video encoder 200 and video decoder 300 may be configured to perform the techniques of this disclosure in conjunction with any video coding techniques that use motion vector prediction and/or reference picture derivation.

**[0065]** In general, video encoder 200 and video decoder 300 may perform block-based coding of pictures. The term "block" generally refers to a structure including data to be processed (e.g., encoded, decoded, or otherwise used in the encoding and/or decoding process). For example, a block may include a two-dimensional matrix of samples of luminance and/or chrominance data. In general, video encoder 200 and video decoder 300 may code video data represented in a YUV (e.g., Y, Cb, Cr) format. That is, rather than coding red, green, and blue (RGB) data for samples of a picture, video encoder 200 and video decoder 300 may code luminance and chrominance components, where the chrominance components may include both red hue and blue hue chrominance components. In some examples, video encoder 200 converts received RGB formatted data to a YUV representation prior to encoding, and video decoder 300 converts the YUV representation to the RGB format. Alternatively, pre- and post-processing units (not shown) may perform these conversions.

**[0066]** This disclosure may generally refer to coding (e.g., encoding and decoding) of pictures to include the process of encoding or decoding data of the picture. Similarly, this disclosure may refer to coding of blocks of a picture to include the process of encoding or decoding data for the blocks, e.g., prediction and/or residual coding. An encoded video bitstream generally includes a series of values for syntax elements representative of coding decisions (e.g., coding modes) and partitioning of pictures into blocks. Thus, references to coding a picture or a block should generally be understood as coding values for syntax elements forming the picture or block.

**[0067]** HEVC defines various blocks, including coding units (CUs), prediction units (PUs), and transform units (TUs). According to HEVC, a video coder (such as video encoder 200) partitions a coding tree unit (CTU) into CUs according to a quadtree structure. That is, the video coder partitions CTUs and CUs into four equal, non-overlapping squares, and each node of the quadtree has either zero or four child nodes. Nodes without child nodes may be referred to as "leaf nodes," and CUs of such leaf nodes

may include one or more PUs and/or one or more TUs. The video coder may further partition PUs and TUs. For example, in HEVC, a residual quadtree (RQT) represents partitioning of TUs. In HEVC, PUs represent inter-prediction data, while TUs represent residual data. CUs that are intra-predicted include intra-prediction information, such as an intra-mode indication.

**[0068]** As another example, video encoder 200 and video decoder 300 may be configured to operate according to VVC. According to VVC, a video coder (such as video encoder 200) partitions a picture into a plurality of CTUs. Video encoder 200 may partition a CTU according to a tree structure, such as a quadtree-binary tree (QTBT) structure or Multi-Type Tree (MTT) structure. The QTBT structure removes the concepts of multiple partition types, such as the separation between CUs, PUs, and TUs of HEVC. A QTBT structure includes two levels: a first level partitioned according to quadtree partitioning, and a second level partitioned according to binary tree partitioning. A root node of the QTBT structure corresponds to a CTU. Leaf nodes of the binary trees correspond to CUs.

**[0069]** In an MTT partitioning structure, blocks may be partitioned using a quadtree (QT) partition, a binary tree (BT) partition, and one or more types of triple tree (TT) (also called ternary tree (TT)) partitions. A triple or ternary tree partition is a partition where a block is split into three sub-blocks. In some examples, a triple or ternary tree partition divides a block into three sub-blocks without dividing the original block through the center. The partitioning types in MTT (e.g., QT, BT, and TT), may be symmetrical or asymmetrical.

**[0070]** When operating according to the AV1 codec, video encoder 200 and video decoder 300 may be configured to code video data in blocks. In AV1, the largest coding block that can be processed is called a superblock. In AV1, a superblock can be either 128x128 luma samples or 64x64 luma samples. However, in successor video coding formats (e.g., AV2), a superblock may be defined by different (e.g., larger) luma sample sizes. In some examples, a superblock is the top level of a block quadtree. Video encoder 200 may further partition a superblock into smaller coding blocks. Video encoder 200 may partition a superblock and other coding blocks into smaller blocks using square or non-square partitioning. Non-square blocks may include N/2xN, NxN/2, N/4xN, and NxN/4 blocks. Video encoder 200 and video decoder 300 may perform separate prediction and transform processes on each of the coding blocks.

**[0071]** AV1 also defines a tile of video data. A tile is a rectangular array of superblocks that may be coded independently of other tiles. That is, video encoder 200 and video decoder 300 may encode and decode, respectively, coding blocks within a tile without

using video data from other tiles. However, video encoder 200 and video decoder 300 may perform filtering across tile boundaries. Tiles may be uniform or non-uniform in size. Tile-based coding may enable parallel processing and/or multi-threading for encoder and decoder implementations.

**[0072]** In some examples, video encoder 200 and video decoder 300 may use a single QTBT or MTT structure to represent each of the luminance and chrominance components, while in other examples, video encoder 200 and video decoder 300 may use two or more QTBT or MTT structures, such as one QTBT/MTT structure for the luminance component and another QTBT/MTT structure for both chrominance components (or two QTBT/MTT structures for respective chrominance components).

**[0073]** Video encoder 200 and video decoder 300 may be configured to use quadtree partitioning, QTBT partitioning, MTT partitioning, superblock partitioning, or other partitioning structures.

**[0074]** In some examples, a CTU includes a coding tree block (CTB) of luma samples, two corresponding CTBs of chroma samples of a picture that has three sample arrays, or a CTB of samples of a monochrome picture or a picture that is coded using three separate color planes and syntax structures used to code the samples. A CTB may be an NxN block of samples for some value of N such that the division of a component into CTBs is a partitioning. A component is an array or single sample from one of the three arrays (luma and two chroma) that compose a picture in 4:2:0, 4:2:2, or 4:4:4 color format or the array or a single sample of the array that compose a picture in monochrome format. In some examples, a coding block is an MxN block of samples for some values of M and N such that a division of a CTB into coding blocks is a partitioning.

**[0075]** The blocks (e.g., CTUs or CUs) may be grouped in various ways in a picture. As one example, a brick may refer to a rectangular region of CTU rows within a particular tile in a picture. A tile may be a rectangular region of CTUs within a particular tile column and a particular tile row in a picture. A tile column refers to a rectangular region of CTUs having a height equal to the height of the picture and a width specified by syntax elements (e.g., such as in a picture parameter set). A tile row refers to a rectangular region of CTUs having a height specified by syntax elements (e.g., such as in a picture parameter set) and a width equal to the width of the picture.

**[0076]** In some examples, a tile may be partitioned into multiple bricks, each of which may include one or more CTU rows within the tile. A tile that is not partitioned into multiple bricks may also be referred to as a brick. However, a brick that is a true subset

of a tile may not be referred to as a tile. The bricks in a picture may also be arranged in a slice. A slice may be an integer number of bricks of a picture that may be exclusively contained in a single network abstraction layer (NAL) unit. In some examples, a slice includes either a number of complete tiles or only a consecutive sequence of complete bricks of one tile.

**[0077]** This disclosure may use "NxN" and "N by N" interchangeably to refer to the sample dimensions of a block (such as a CU or other video block) in terms of vertical and horizontal dimensions, e.g., 16x16 samples or 16 by 16 samples. In general, a 16x16 CU will have 16 samples in a vertical direction (y = 16) and 16 samples in a horizontal direction (x = 16). Likewise, an NxN CU generally has N samples in a vertical direction and N samples in a horizontal direction, where N represents a nonnegative integer value. The samples in a CU may be arranged in rows and columns. Moreover, CUs need not necessarily have the same number of samples in the horizontal direction as in the vertical direction. For example, CUs may include NxM samples, where M is not necessarily equal to N.

**[0078]** Video encoder 200 encodes video data for CUs representing prediction and/or residual information, and other information. The prediction information indicates how the CU is to be predicted in order to form a prediction block for the CU. The residual information generally represents sample-by-sample differences between samples of the CU prior to encoding and the prediction block.

**[0079]** To predict a CU, video encoder 200 may generally form a prediction block for the CU through inter-prediction or intra-prediction. Inter-prediction generally refers to predicting the CU from data of a previously coded picture, whereas intra-prediction generally refers to predicting the CU from previously coded data of the same picture. To perform inter-prediction, video encoder 200 may generate the prediction block using one or more motion vectors. Video encoder 200 may generally perform a motion search to identify a reference block that closely matches the CU, e.g., in terms of differences between the CU and the reference block. Video encoder 200 may calculate a difference metric using a sum of absolute difference (SAD), sum of squared differences (SSD), mean absolute difference (MAD), mean squared differences (MSD), or other such difference calculations to determine whether a reference block closely matches the current CU. In some examples, video encoder 200 may predict the current CU using uni-directional prediction or bi-directional prediction.

**[0080]** Some examples of VVC also provide an affine motion compensation mode, which may be considered an inter-prediction mode. In affine motion compensation mode, video encoder 200 may determine two or more motion vectors that represent non-translational motion, such as zoom in or out, rotation, perspective motion, or other irregular motion types.

**[0081]** To perform intra-prediction, video encoder 200 may select an intra-prediction mode to generate the prediction block. Some examples of VVC provide sixty-seven intra-prediction modes, including various directional modes, as well as planar mode and DC mode. In general, video encoder 200 selects an intra-prediction mode that describes neighboring samples to a current block (e.g., a block of a CU) from which to predict samples of the current block. Such samples may generally be above, above and to the left, or to the left of the current block in the same picture as the current block, assuming video encoder 200 codes CTUs and CUs in raster scan order (left to right, top to bottom).

**[0082]** Video encoder 200 encodes data representing the prediction mode for a current block. For example, for inter-prediction modes, video encoder 200 may encode data representing which of the various available inter-prediction modes is used, as well as motion information for the corresponding mode. For uni-directional or bi-directional inter-prediction, for example, video encoder 200 may encode motion vectors using advanced motion vector prediction (AMVP) or merge mode. Video encoder 200 may use similar modes to encode motion vectors for affine motion compensation mode.

**[0083]** AV1 includes two general techniques for encoding and decoding a coding block of video data. The two general techniques are intra prediction (e.g., intra frame prediction or spatial prediction) and inter prediction (e.g., inter frame prediction or temporal prediction). In the context of AV1, when predicting blocks of a current frame of video data using an intra prediction mode, video encoder 200 and video decoder 300 do not use video data from other frames of video data. For most intra prediction modes, video encoder 200 encodes blocks of a current frame based on the difference between sample values in the current block and predicted values generated from reference samples in the same frame. Video encoder 200 determines predicted values generated from the reference samples based on the intra prediction mode.

**[0084]** Following prediction, such as intra-prediction or inter-prediction of a block, video encoder 200 may calculate residual data for the block. The residual data, such as a residual block, represents sample by sample differences between the block and a prediction block for the block, formed using the corresponding prediction mode. Video

encoder 200 may apply one or more transforms to the residual block, to produce transformed data in a transform domain instead of the sample domain. For example, video encoder 200 may apply a discrete cosine transform (DCT), an integer transform, a wavelet transform, or a conceptually similar transform to residual video data. Additionally, video encoder 200 may apply a secondary transform following the first transform, such as a mode-dependent non-separable secondary transform (MDNSST), a signal dependent transform, a Karhunen-Loeve transform (KLT), or the like. Video encoder 200 produces transform coefficients following application of the one or more transforms.

**[0085]** As noted above, following any transforms to produce transform coefficients, video encoder 200 may perform quantization of the transform coefficients. Quantization generally refers to a process in which transform coefficients are quantized to possibly reduce the amount of data used to represent the transform coefficients, providing further compression. By performing the quantization process, video encoder 200 may reduce the bit depth associated with some or all of the transform coefficients. For example, video encoder 200 may round an $n$-bit value down to an $m$-bit value during quantization, where $n$ is greater than $m$. In some examples, to perform quantization, video encoder 200 may perform a bitwise right-shift of the value to be quantized.

**[0086]** Following quantization, video encoder 200 may scan the transform coefficients, producing a one-dimensional vector from the two-dimensional matrix including the quantized transform coefficients. The scan may be designed to place higher energy (and therefore lower frequency) transform coefficients at the front of the vector and to place lower energy (and therefore higher frequency) transform coefficients at the back of the vector. In some examples, video encoder 200 may utilize a predefined scan order to scan the quantized transform coefficients to produce a serialized vector, and then entropy encode the quantized transform coefficients of the vector. In other examples, video encoder 200 may perform an adaptive scan. After scanning the quantized transform coefficients to form the one-dimensional vector, video encoder 200 may entropy encode the one-dimensional vector, e.g., according to context-adaptive binary arithmetic coding (CABAC). Video encoder 200 may also entropy encode values for syntax elements describing metadata associated with the encoded video data for use by video decoder 300 in decoding the video data.

**[0087]** To perform CABAC, video encoder 200 may assign a context within a context model to a symbol to be transmitted. The context may relate to, for example, whether

neighboring values of the symbol are zero-valued or not. The probability determination may be based on a context assigned to the symbol.

**[0088]** Video encoder 200 may further generate syntax data, such as block-based syntax data, picture-based syntax data, and sequence-based syntax data, to video decoder 300, e.g., in a picture header, a block header, a slice header, or other syntax data, such as a sequence parameter set (SPS), picture parameter set (PPS), or video parameter set (VPS). Video decoder 300 may likewise decode such syntax data to determine how to decode corresponding video data.

**[0089]** In this manner, video encoder 200 may generate a bitstream including encoded video data, e.g., syntax elements describing partitioning of a picture into blocks (e.g., CUs) and prediction and/or residual information for the blocks. Ultimately, video decoder 300 may receive the bitstream and decode the encoded video data.

**[0090]** In general, video decoder 300 performs a reciprocal process to that performed by video encoder 200 to decode the encoded video data of the bitstream. For example, video decoder 300 may decode values for syntax elements of the bitstream using CABAC in a manner substantially similar to, albeit reciprocal to, the CABAC encoding process of video encoder 200. The syntax elements may define partitioning information for partitioning of a picture into CTUs, and partitioning of each CTU according to a corresponding partition structure, such as a QTBT structure, to define CUs of the CTU. The syntax elements may further define prediction and residual information for blocks (e.g., CUs) of video data.

**[0091]** The residual information may be represented by, for example, quantized transform coefficients. Video decoder 300 may inverse quantize and inverse transform the quantized transform coefficients of a block to reproduce a residual block for the block. Video decoder 300 uses a signaled prediction mode (intra- or inter-prediction) and related prediction information (e.g., motion information for inter-prediction) to form a prediction block for the block. Video decoder 300 may then combine the prediction block and the residual block (on a sample-by-sample basis) to reproduce the original block. Video decoder 300 may perform additional processing, such as performing a deblocking process to reduce visual artifacts along boundaries of the block.

**[0092]** This disclosure may generally refer to "signaling" certain information, such as syntax elements. The term "signaling" may generally refer to the communication of values for syntax elements and/or other data used to decode encoded video data. That is, video encoder 200 may signal values for syntax elements in the bitstream. In general,

signaling refers to generating a value in the bitstream. As noted above, source device 102 may transport the bitstream to destination device 116 substantially in real time, or not in real time, such as might occur when storing syntax elements to storage device 112 for later retrieval by destination device 116.

**[0093]** In accordance with the techniques of this disclosure, a method includes: determining, for each of a plurality of subblocks of a current block of a current picture of the video data, a corresponding intersecting motion vector list, each corresponding intersecting motion vector list comprising at least one intersecting motion vector, the at least one intersecting motion vector comprising a source motion vector associated with a source block located in a first reference picture and a destination motion vector associated with a destination block located in a second reference picture, and wherein the source block has a predictor that is a subset of the destination block, the first reference picture being already decoded and temporally prior to the current picture and the second reference picture being already decoded and temporally after the current picture; determining an intersecting motion vector for the current block based on the corresponding intersecting motion vector lists; and decoding the current block based on the intersecting motion vector.

**[0094]** In another example, a method includes: determining a current reference picture for a current block of a current picture of the video data based on a reference picture of a collocated block, a reference picture list of the current block, and a picture order count (POC) of a collocated picture, wherein the collocated block is collocated with the current block and is in the collocated picture; and

decoding the current block based on a motion vector associated with the current reference picture.

**[0095]** In another example, a method includes determining at least one intersecting motion vector for a current block of a current picture of the video data, the intersecting motion vector comprising a source motion vector associated with a first reference picture and a destination motion vector associated with a second reference picture, the first reference picture being temporally prior to the current picture and the second reference picture being temporally after the current picture; and coding the current block based on the at least one intersecting motion vector.

**[0096]** In another example, a method includes: determining a temporal motion vector predictor for a current block of a current picture of the video data based on a collocated block of a collocated picture offset by a motion vector of a spatial neighbor block,

neighboring the current block in the current picture; and coding the current block based on the temporal motion vector predictor.

**[0097]** In another example, a method includes: determining a reference picture for a current block of the video data; and coding the current block of the video data based on a motion vector associated with the reference picture.

**[0098]** In another example, a method includes: determining one or more new motion vector predictors from at least one existing motion vector predictor in a motion vector predictor list for a current block of the video data; and coding the video data based on the one or more new motion vector predictors.

**[0099]** In another example, a device includes one or more memories configured to store video data and one or more processors configured to perform any of the techniques described herein.

**[0100]** In another example, a device includes one or more means configured to perform any of the techniques described herein.

**[0101]** In another example, computer-readable storage media is encoded with instructions that, when executed, cause one or more processors to perform any of the techniques described herein.

**[0102]** This disclosure describes motion vector prediction and reference picture derivation techniques. These techniques may be applied to any of the existing video codecs, such as HEVC (High Efficiency Video Coding), VVC (Versatile Video Coding), Essential Video Coding (EVC) or be an efficient coding tool in any future video coding standards. HEVC and JEM techniques and on-going works in Versatile Video Coding (VVC) related to the techniques of this disclosure are now reviewed.

**[0103]** Video coding standards include ITU-T H.261, ISO/IEC MPEG-1 Visual, ITU-T H.262 or ISO/IEC MPEG-2 Visual, ITU-T H.263, ISO/IEC MPEG-4 Visual and ITU-T H.264 (also known as ISO/IEC MPEG-4 AVC), including its Scalable Video Coding (SVC) and Multi-view Video Coding (MVC) extensions.

**[0104]** In addition, a new video coding standard, namely High Efficiency Video Coding (HEVC) or ITU-T H.265, including its range extension, multiview extension (MV-HEVC) and scalable extension (SHVC), has recently been developed by the Joint Collaboration Team on Video Coding (JCT-VC) as well as Joint Collaboration Team on 3D Video Coding Extension Development (JCT-3V) of ITU-T Video Coding Experts Group (VCEG) and ISO/IEC Motion Picture Experts Group (MPEG).

**[0105]** The latest HEVC draft specification, and referred to as HEVC WD hereinafter, is available from phenix.int-evry.fr/jct/doc_end_user/documents/14_Vienna/wg11/JCTVC-N1003-v1.

**[0106]** ITU-T VCEG (Q6/16) and ISO/IEC MPEG (JTC 1/SC 29/WG 11) are now studying the potential need for standardization of future video coding technology with a compression capability that significantly exceeds that of the current HEVC standard (including its current extensions and near-term extensions for screen content coding and high-dynamic-range coding). The groups are working together on this exploration activity in a joint collaboration effort known as the Joint Video Exploration Team (JVET) to evaluate compression technology designs proposed by their experts in this area. A version of reference software, i.e., VVC Test Model 10 (VTM 10.0) could be downloaded from: vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM.

**[0107]** The Versatile Video Coding (VVC) draft specification, Bross et al., "Versatile Video Coding Editorial Refinements on Draft 10," Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 20th Meeting: by teleconference, 7-16 October 2020, JVET- T2001, may be referred to herein as JVET-T2001.

**[0108]** Algorithm description of Chen et al. "Algorithm description for Versatile Video Coding and Test Model 11 (VTM 11)," Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 20th Meeting: by teleconference, 7-16 October 2020, JVET- T2002, may be referred to herein as JVET-T2002.

**[0109]** The CU structure and motion vector prediction in HEVC are now described. 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.

**[0110]** 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 to as small as 8x8. Each coding unit is coded with one mode, e.g., inter or intra. 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 partitioning does not 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.

**[0111]** When the CU is inter coded, each PU has one set of motion information, which is derived with a unique inter prediction mode.

**[0112]** Motion vector prediction is now described. In HEVC standard, there are two inter prediction modes, named merge (skip is considered as a special case of merge) and

advanced motion vector prediction (AMVP) modes respectively for a prediction unit (PU).

**[0113]** 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.

**[0114]** 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 used for the prediction of the current blocks, as well as the associated motion vectors are determined. On the other hand, 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 MV predictor (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.

**[0115]** The candidates for both modes are derived similarly from the same spatial and temporal neighboring blocks.

**[0116]** Spatial neighboring candidates are now described. FIGS. 2A-2B are conceptual diagrams illustrating example spatial neighboring MV candidates for merge and AMVP modes, respectively. For example, video encoder 200 or video decoder 300 may derive spatial MV candidates from neighboring blocks shown in FIGS. 2A-2B, for a specific PU (PU0), although the techniques video encoder 200 or video decoder 300 may use to generate the candidates from the blocks differ for merge and AMVP modes.

**[0117]** In merge mode, up to four spatial MV candidates for PU0 130 can be derived with the orders shown in FIG. 2A as follows: left (0, A1), above (1, B1), above right (2, B0), below left (3, A0), and above left (4, B2).

**[0118]** In AMVP mode, the neighboring blocks of PU0 132 are divided into two groups: left group including block 0 and 1, and above group including blocks 2, 3, and 4 as shown on FIG. 2B. For each group, the potential candidate in a neighboring block that refers 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 for PU0 132. 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. For example, video encoder 200 or video decoder 300 may select the first available candidate and scale that first available candidate to form a final candidate to compensate for temporal distance differences.

**[0119]** Temporal motion vector prediction in HEVC is now described. Video coders, such as video encoder 200 or video decoder 300 may implement HEVC. Temporal motion vector predictor (TMVP) candidate, if enabled and available, is added into the MV candidate list after spatial motion vector candidates. The process of motion vector derivation for TMVP candidate is the same for both merge and AMVP modes, however the target reference index for the TMVP candidate in the merge mode is always set to 0.

**[0120]** FIG. 3A is a conceptual diagram illustrating an example of a temporal motion vector predictor. FIG. 3A shows example TMVP candidates for block 156 (PU0) and FIG. 3B shows a motion vector scaling process. The primary block location for TMVP candidate derivation is the bottom right block outside of the collocated PU. This candidate is shown in FIG. 3A as a block "T" 154. The location of block T 154 is used to compensate the bias to the above and left blocks used to generate spatial neighboring candidates. However, if that block is located outside of the current CTB row (shown as block T 152), or motion information is not available (shown as block T 150), the block is substituted with a center block of the PU.

**[0121]** FIG. 3B is a conceptual diagram illustrating an example of motion vector scaling. Similar to temporal direct mode in AVC, to derive the TMVP candidate motion vector, the collocated MV may be scaled to compensate the temporal distance differences, as shown in FIG. 3B. Video encoder 200 and video decoder 300 may derive a motion vector for the TMVP candidate from the collocated PU of the collocated picture, indicated in the slice level. The motion vector for the collocated PU is called a collocated MV. Similar to temporal direct mode in AVC, to derive TMVP candidate motion vector 157, collocated MV 158 may be scaled to compensate the temporal distance differences, as shown in FIG. 3B.

**[0122]** Temporal motion information derivation is now described. In VVC, a TMVP for the AMVP and merge mode is derived by fetching the motion information from the center or the bottom-right of the collocated block in a signaled collocated picture. Similarly, for the Subblock-based Temporal Motion Vector Prediction (SbTMVP) mode, the motion information from the left neighboring position is used as a motion shift, which is then employed to obtain TMVPs at a sub-CU level.

**[0123]** Video encoder 200 or video decoder 300 may implement the VVC or Enhanced Compression Model (ECM) TMVP techniques. In ECM, to further improve the coding efficiency of temporal motion vector prediction, two aspects are modified. Firstly, two collocated pictures are utilized which are the two reference frames with the least picture order count (POC) distance relative to the to-be-coded frame. Secondly, the motion shift to locate a TMVP is adaptively determined from multiple locations according to template costs. More specifically, two motion shift candidate lists are constructed respectively for two collocated frames. The motion shifts with the minimum template matching cost are used to derive SbTMVP or TMVP candidates. At most 4 SbTMVP candidates are included in the sub-block-based merge list. The SbTMVP candidate with the least template matching cost derived from the first collocated frame is placed in the first entry without reordering, while other SbTMVP candidates are sorted together with affine candidates. In addition, the prediction direction of each subblock template is determined based on the center subblock.

**[0124]** FIG. 4 is a conceptual diagram illustrating example temporal motion information derivation. As illustrated in FIG. 4, if the center subblock 140 is uni-predicted, then all the subblock templates 142A-142H are uni-predicted, and vice versa. If the motion vector of corresponding adjacent subblock at the determined reference list is not available for a subblock template, a zero MV is used for that subblock template.

**[0125]** Other aspects of motion prediction in HEVC are now described. Several aspects of merge and AMVP modes are worth mentioning as follows.

**[0126]** Motion vector scaling: It is assumed that the value of motion vectors is proportional to the distance of pictures in the presentation time. A motion vector associates two pictures, the reference picture, and the picture containing the motion vector (namely the containing picture). When a motion vector is utilized to predict the other motion vector, the distance of the containing picture and the reference picture is calculated based on the POC values.

**[0127]** For a motion vector to be predicted, both its associated containing picture and reference picture may be different. Therefore, a new distance (based on POC) is calculated. And the motion vector is scaled based on these two POC distances. For a spatial neighboring candidate, the containing pictures for the two motion vectors are the same, while the reference pictures are different. In HEVC, motion vector scaling applies to both TMVP and AMVP for spatial and temporal neighboring candidates.

**[0128]** Artificial motion vector candidate generation: If a motion vector candidate list is not complete, artificial motion vector candidates are generated and inserted at the end of the list until it will have all candidates.

**[0129]** In merge mode, there are two types of artificial MV candidates: combined candidate derived only for B-slices and zero candidates used only for AMVP if the first type does not provide enough artificial candidates.

**[0130]** 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.

**[0131]** 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. The pruning process compares one candidate against the others in the current candidate list to avoid inserting identical candidate in certain extent. To reduce the complexity, only limited numbers of pruning process is applied instead of comparing each potential one with all the other existing ones.

**[0132]** Template matching (TM) prediction is now described. Video encoder 200 or video decoder 300 may implement TM prediction. TM prediction is a special merge mode based on Frame-Rate Up Conversion (FRUC) techniques. With this TM prediction mode, motion information of a block is not signaled, but is derived at the decoder side. For example, video encoder 200 may not signal motion information with TM prediction mode, and video decoder 300 may derive the motion information. TM prediction mode may be applied to both AMVP mode and regular merge mode. In AMVP mode, MVP candidate selection may be determined based on template matching to select a candidate which has the minimal difference between a current block template and a reference block template. In regular merge mode, a TM mode flag may be signaled to indicate the use of TM and then TM may be applied to the merge candidate indicated by merge index for MV refinement.

**[0133]** FIG. 5 is a conceptual diagram illustrating an example of template matching performed on a search area around an initial MV. As shown in FIG. 5, template matching is used to derive motion information of current CU 160 by finding the closest match between a current template 162 in the current picture and a reference template 164, which may be the same size as current template 162, in a reference picture. Both current template 162 and reference template 164 may include respective above and left portions, as shown.

With an AMVP candidate selected based on initial matching error, video encoder 200 and video decoder 300 may refine the MVP using template matching. With a merge candidate indicated by signaled merge index, video encoder 200 and video decoder 300 may be configured to refine MVs corresponding to L0 and L1 independently by template matching and then further refine the less accurate MV based on the more accurate MV.

**[0134]** Video encoder 200 and video decoder 300 may be configured to implement a cost function. When a motion vector points to a fractional sample position, motion compensated interpolation is needed. To reduce complexity, bi-linear interpolation instead of regular 8-tap DCT-IF interpolation is used for both template matching to generate templates on reference pictures. The matching cost C of template matching is calculated as follows:

$$C = SAD + w \cdot (|MV_x - MV_x^s| + |MV_y - MV_y^s|)$$

where w is a weighting factor which is empirically set to 4, MV and MV^s indicate the currently testing MV and the initial MV (i.e., an MVP candidate in AMVP mode or merged motion candidate in merge mode), respectively. SAD is used as the matching cost of template matching.

**[0135]** When TM is used, motion is refined by using luma samples only. The derived motion will be used for both luma and chroma for MC inter prediction. After MV is decided, final MC is performed using 8-taps interpolation filter (IF) for luma and 4-taps interpolation filter for chroma.

**[0136]** Video encoder 200 and video decoder 300 may be configured to implement a search process. MV refinement is a pattern based MV search with the criterion of template matching cost. two search patterns are supported – a diamond search and a cross search for MV refinement. The MV is directly searched at quarter luma sample MVD accuracy with diamond pattern, followed by quarter luma sample MVD accuracy with cross pattern, and then this is followed by one-eighth luma sample MVD refinement with cross pattern. The search range of MV refinement is set equal to (-8, +8) luma samples around the initial MV.

**[0137]** Bilateral Matching Prediction is now discussed. Video encoder 200 and video decoder 300 may be configured to perform bilateral matching prediction. Bilateral Matching (also known as Bilateral Merge) (BM) prediction is another merge mode base on Frame-Rate Up Conversion (FRUC) techniques. When applying BM mode to a block, video encoder 200 and video decoder 300 may derive two initial motion vectors MV0 and MV1 using a signaled merge candidate index to select the merge candidate in a

constructed merge list. When implementing bilateral matching, video encoder 200 and video decoder 300 search around the MV0 and MV1 and derive the final MV0' and MV1' based on a minimum bilateral matching cost.

**[0138]** The motion vector difference (MVD) MVD0 (denoted by MV0' – MV0) and MVD1 (denoted by MV1' – MV1) pointing to the two reference blocks may be proportional to the temporal distances (TD), e.g., TD0 and TD1, between the current picture and the two reference pictures. FIG. 4 shows an example of MVD0 and MVD1 where the distance (TD1) between current picture 170 and reference picture 172 is 4-times the distance (TD0) between current picture 170 and reference picture 174. FIG. 4 shows an example of MVD0 and MVD1 being proportional based on the temporal distances.

**[0139]** However, there is an optional design where MVD0 and MVD1 are mirrored regardless of the temporal distances TD0 and TD1. FIG. 7 shows an example of MVD0 and MVD1 being mirrored regardless of the temporal distance (TD1) between current picture 176 and reference picture 178 and the temporal distance (TD0) between current picture 176 and reference picture 180. FIG. 7 shows an example of mirrored MVD0 and MVD1, where TD1 is 4-times of TD0.

**[0140]** FIG. 8 is a conceptual diagram illustrating an example of the 3×3 square search pattern in the search range [-8, 8]. FIG. 8 shows an example of 3×3 square search patterns in the search range [-8, 8] for implementing bilateral matching. When implementing bilateral matching, video encoder 200 and video decoder 300 may be configured to perform a local search around the initial MV0 and MV1 to derive the final MV0' and MV1'. In the example of FIG. 8, the initial MV points to sample 182, and the final MV points to sample 184. The local search applies a 3×3 square search pattern to loop through the search range [-8, 8]. Samples 186 represent examples of samples in the search range around samples 182, 184, and sample 188. Sample 188 represents an example of a sample corresponding to a MV determined during an intermediate iteration of the search process. In each search iteration, the bilateral matching cost of the eight surrounding MVs in the search pattern are calculated and compared to the bilateral matching cost of center MV. The MV which has minimum bilateral matching cost becomes the new center MV in the next search iteration. The local search is terminated when the current center MV has a minimum cost within the 3×3 square search pattern or the local search reaches the pre-defined maximum search iteration.

**[0141]** Decoder-side motion vector refinement (DMVR) is now discussed. To increase the accuracy of the MVs of the merge mode, DMVR may be applied in VVC. For example, video encoder 200 or video decoder 300 may implement DMVR. In a bi-prediction operation, a refined MV is searched around the initial MVs in the reference picture list L0 and reference picture list L1. Video encoder 200 or video decoder 300 implementing the DMVR technique calculates the distortion between the two candidate blocks in the reference picture list L0 and list L1. As illustrated in FIG. 9, the SAD between blocks 190 and 192 based on each MV candidate around the initial MV is calculated. The MV candidate with the lowest SAD becomes the refined MV and used to generate the bi-predicted signal.

**[0142]** The refined MV derived by video encoder 200 or video decoder 300 implementing the DMVR techniques is used to generate the inter prediction samples and also used in temporal motion vector prediction for future pictures coding (e.g., coding of future pictures). While the original MV is used in a deblocking process and also used in spatial motion vector prediction for future CU coding.

**[0143]** DMVR is a sub-block based merge mode with a pre-defined maximum processing unit of 16x16 luma samples. When the width and/or height of a CU are larger than 16 luma samples, the CU will be further split into subblocks with width and/or height equal to 16 luma samples.

**[0144]** A searching scheme is now discussed. In DMVR, the search points are surrounding the initial MV and the MV offset by the MV difference mirroring rule. In other words, any points that are checked by DMVR, denoted by candidate MV pair (MV0, MV1) follow the two equations:

$$MV0' = MV0 + MV_offset$$

$$MV1' = MV1 - MV_offset$$

Where $MV_offset$ represents the refinement offset between the initial MV and the refined MV in one of the reference pictures. The refinement search range is two integer luma samples from the initial MV. The searching includes the integer sample offset search stage and fractional sample refinement stage.

**[0145]** A 25 points full search is applied for integer sample offset searching. The SAD of the initial MV pair is first calculated. If the SAD of the initial MV pair is smaller than a threshold, the integer sample stage of DMVR is terminated. Otherwise SADs of the remaining 24 points are calculated and checked in raster scanning order. The point with the smallest SAD is selected as the output of integer sample offset searching stage. To

reduce the penalty of the uncertainty of DMVR refinement, the original MV may be favored during the DMVR process. The SAD between the reference blocks referred by the initial MV candidates is decreased by 1/4 of the SAD value.

**[0146]** The integer sample search is followed by a fractional sample refinement. To save on calculational complexity, the fractional sample refinement is derived by using a parametric error surface equation, instead of an additional search with a SAD comparison. The fractional sample refinement is conditionally invoked based on the output of the integer sample search stage. When the integer sample search stage is terminated with center having the smallest SAD in either the first iteration or the second iteration search, the fractional sample refinement is further applied.

**[0147]** In a parametric error surface based sub-pixel offsets estimation, the center position cost and the costs at four neighboring positions from the center are used to fit a 2-D parabolic error surface equation of the following form:

$$E(x,y) = A(x - x_{min})^2 + B(y - y_{min})^2 + C$$

where $(x_{min}, y_{min})$ corresponds to the fractional position with the least cost and C corresponds to the minimum cost value. By solving the above equations by using the cost value of the five search points, the $(x_{min}, y_{min})$ is computed as:

$$x_{min} = (E(-1,0) - E(1,0))/(2(E(-1,0) + E(1,0) - 2E(0,0)))$$

$$y_{min} = (E(0,-1) - E(0,1))/(2((E(0,-1) + E(0,1) - 2E(0,0))))$$

**[0148]** The value of $x_{min}$ and $y_{min}$ are automatically constrained to be between − 8 and 8 since all cost values are positive and the smallest value is $E(0,0)$. This corresponds to a half-pel offset with 1/16th-pel MV accuracy in VVC. The computed fractional $(x_{min}, y_{min})$ are added to the integer distance refinement MV to get the sub-pixel accurate refinement delta MV.

**[0149]** Bilinear-interpolation and sample padding is now described. Video decoder 300, may implement bilinear-interpolation and/or sample padding. The samples at the fractional positions are interpolated using an 8-tap interpolation filter. In DMVR, the search points are surrounding the initial fractional-pel MV with an integer sample offset, therefore the samples of those fractional positions need to be interpolated for the DMVR search process. To reduce the calculation complexity, the bi-linear interpolation filter is used to generate the fractional samples for the searching process in DMVR. Another important effect of using the bi-linear filter is that with a 2-sample search range, the DMVR does not access more reference samples compared to the normal motion

compensation process. After the refined MV is attained with DMVR search process, the normal 8-tap interpolation filter is applied to generate the final prediction. In order to not access more reference samples than the normal MC process, the samples, which are not needed for the interpolation process based on the original MV but are needed for the interpolation process based on the refined MV, will be padded from those available samples.

**[0150]** Enabling conditions for DMVR are now discussed. DMVR is enabled if the following conditions are satisfied (e.g., all satisfied): 1) CU level merge mode with bi-prediction MV; 2) One reference picture is in the past and another reference picture is in the future with respect to the current picture; 3) The distances (e.g., POC differences) from both reference pictures to the current picture are same; 4) CU has more than 64 luma samples; 5) Both CU height and CU width are larger than or equal to 8 luma samples; 6) bi-prediction with CU-based weighting (BCW) weight index indicates equal weight; 7) weighted prediction (WP) is not enabled for the current block; and 8) combined intra-inter prediction (CIIP) mode is not used for the current block.

**[0151]** Multi-pass decoder-side motion vector refinement in ECM is now discussed. Video encoder 200 or video decoder 300 may implement multi-pass decoder-side motion vector refinement. A multi-pass decoder-side motion vector refinement may be applied. In the first pass, bilateral matching (BM) is applied to the coding block. In the second pass, BM is applied to each 16x16 subblock within the coding block. In the third pass, a MV in each 8x8 subblock is refined by applying bi-directional optical flow (BDOF). The refined MVs are stored for both spatial and temporal motion vector prediction.

**[0152]** First pass – block based bilateral matching MV refinement is now discussed. In the first pass, a refined MV is derived by applying BM to a coding block. Similar to DMVR, in bi-prediction operation, a refined MV is searched around the two initial MVs (MV0 and MV1) in the reference picture lists L0 and L1. The refined MVs (MV0_pass1 and MV1_pass1) are derived around the initial MVs based on the minimum bilateral matching cost between the two reference blocks in L0 and L1.

**[0153]** Video encoder 200 or video decoder 300 implementing BM performs a local search to derive integer sample precision intDeltaMV. The local search applies a 3×3 square search pattern to loop through the search range [–sHor, sHor] in a horizontal direction and [–sVer, sVer] in a vertical direction, wherein, the values of sHor and sVer are determined by the block dimension, and the maximum value of sHor and sVer is 8.

**[0154]** The bilateral matching cost may be calculated as: bilCost = mvDistanceCost + sadCost. When the block size cbW * cbH is greater than 64, mean reduced sum of average difference (MRSAD) cost function is applied to remove the DC effect of distortion between reference blocks. When the bilCost at the center point of the 3×3 search pattern has the minimum cost, the intDeltaMV local search is terminated. Otherwise, the current minimum cost search point becomes the new center point of the 3×3 search pattern and video encoder 200 or video decoder 300 may continue to search for the minimum cost, until video encoder 200 or video decoder 300 reaches the end of the search range.

**[0155]** The existing fractional sample refinement is further applied to derive the final deltaMV. The refined MVs after the first pass is then derived as:

MV0_pass1 = MV0 + deltaMV

MV1_pass1 = MV1 – deltaMV

**[0156]** Second pass – Subblock based bilateral matching MV refinement is now discussed. In the second pass, a refined MV is derived by applying BM to a 16×16 grid subblock. For each subblock, a refined MV is searched around the two MVs (MV0_pass1 and MV1_pass1), obtained on the first pass, in the reference picture list L0 and L1. The refined MVs (MV0_pass2(sbIdx2) and MV1_pass2(sbIdx2)) are derived based on the minimum bilateral matching cost between the two reference subblocks in L0 and L1.

**[0157]** For each subblock, video encoder 200 or video decoder 300 implementing BM performs full search to derive integer sample precision intDeltaMV. The full search has a search range [–sHor, sHor] in horizontal direction and [– sVer, sVer] in vertical direction, wherein, the values of sHor and sVer are determined by the block dimension, and the maximum value of sHor and sVer is 8.

**[0158]** FIG. 10 is a conceptual diagram illustrating example diamond regions in a search area. The bilateral matching cost is calculated by applying a cost factor to the sum of absolute transformed differences (SATD) cost between two reference subblocks, as: bilCost = satdCost * costFactor. The search area (2*sHor + 1) * (2*sVer + 1) is divided up to 5 diamond shape search regions in search area 196 as shown on FIG. 9. Each search region is assigned a costFactor, which is determined by the distance (intDeltaMV) between each search point and the starting MV, and each diamond region is processed in the order starting from the center of the search area. In each region, the search points are processed in the raster scan order starting from the top left going to the bottom right corner of the region. When the minimum bilCost within the current search region is less than a

threshold equal to sbW * sbH, the int-pel full search is terminated, otherwise, the int-pel full search continues to the next search region until all search points are examined.

**[0159]** The existing VVC DMVR fractional sample refinement is further applied to derive the final deltaMV(sbIdx2). The refined MVs at second pass are then derived as:

MV0_pass2(sbIdx2) = MV0_pass1 + deltaMV(sbIdx2)

MV1_pass2(sbIdx2) = MV1_pass1 – deltaMV(sbIdx2)

**[0160]** The third pass – Subblock based bi-directional optical flow MV refinement – is now described. In the third pass, a refined MV is derived by applying BDOF to an 8×8 grid subblock. For each 8×8 subblock, BDOF refinement is applied to derive scaled Vx and Vy without clipping starting from the refined MV of the parent subblock of the second pass. The derived bioMv(Vx, Vy) is rounded to 1/16 sample precision and clipped between -32 and 32.

**[0161]** The refined MVs (MV0_pass3(sbIdx3) and MV1_pass3(sbIdx3)) at third pass may be derived as:

MV0_pass3(sbIdx3) = MV0_pass2(sbIdx2) + bioMv

MV1_pass3(sbIdx3) = MV0_pass2(sbIdx2) – bioMv

**[0162]** Regular merge, affine merge, set BCW default, local illumination compensation (LIC) weight, and LIC BDOF are now described. An affine motion model can be described as

$$\begin{cases} v_x = ax + by + e \\ v_y = cx + dy + f \end{cases}$$

wherein $(v_x, v_y)$ is the motion vector at the coordinate $(x, y)$, and a, b, c, d, e, and f are the six affine parameters. We will refer to this affine motion model as 6-parameters affine motion model. In a typical video coder, a picture is partitioned into blocks for block-based coding. The affine motion model for a block can also be described by the 3 motion vectors (MVs) $\vec{v}_0 = (v_{0x}, v_{0y})$, $\vec{v}_1 = (v_{1x}, v_{1y})$, and $\vec{v}_2 = (v_{2x}, v_{2y})$ at 3 different locations that are not in the same line. The 3 locations are usually referred to as *control-point*s, the 3 motion vectors are referred to as *control-point motion vectors* (CPMVs). In the case when the 3 control-points are at the 3 corners of the block, the affine motion can be described as

$$\begin{cases} v_x = \dfrac{(v_{1x} - v_{0x})}{blkW} x + \dfrac{(v_{2x} - v_{0x})}{blkH} y + v_{0x} \\ v_y = \dfrac{(v_{1y} - v_{0y})}{blkW} x + \dfrac{(v_{2y} - v_{0y})}{blkH} y + v_{0y} \end{cases}$$

wherein $blkW$ and $blkH$ are the width and height of the block.

**[0163]** In affine mode, different motion vectors can be derived for each pixel in the block according to the associate affine motion model. Therefore, motion compensation can be performed in pixel-by-pixel. However, to reduce the complexity, subblock based motion compensation is usually adopted, wherein the block is partitioned into multiple subblocks (that have smaller block size) and each subblock is associate with one motion vector for block-based motion compensation. The motion vector for each subblock is derived using the representative coordinate of the subblock. Typically, the center position is used. In one example, the block is partitioned into non-overlapping subblocks. The block width is blkW, block height is blkH, the subblock width is sbW and subblock height is sbH, then there's blkH/sbH rows of subblocks and blkW/sbW subblocks in each row. For a six-parameter affine motion model, the motion vector for the subblock (referred to as subblock MV) at $i_{th}$ row (0<=i<blkW/sbW) and $j_{th}$ (0<=j<blkH/sbH) column is derived as

$$\begin{cases} v_x = \frac{(v_{1x} - v_{0x})}{blkW}(j * sbW + \frac{sbW}{2}) + \frac{(v_{2x} - v_{0x})}{blkH}(i * sbH + \frac{sbH}{2}) + v_{0x} \\ v_y = \frac{(v_{1y} - v_{0y})}{blkW}(j * sbW + \frac{sbW}{2}) + \frac{(v_{2y} - v_{0y})}{blkH}(i * sbH + \frac{sbH}{2}) + v_{0y} \end{cases}$$

**[0164]** The subblock MVs are rounded to the predefined precision and stored in the motion buffer for motion compensation and motion vector prediction.

**[0165]** A simplified 4-parameters affine model (for zoom and rotational motion) is described as

$$\begin{cases} v_x = ax - by + e \\ v_y = bx + ay + f \end{cases}$$

**[0166]** Similarly, the 4-parameters affine model for a block can be described by 2 CPMVs $\vec{v}_0 = (v_{0x}, v_{0y})$ and $\vec{v}_1 = (v_{1x}, v_{1y})$ at the 2 corners (typically top-left and top-right) of the block. The motion field is then described as

$$\begin{cases} v_x = \frac{(v_{1x} - v_{0x})}{blkW}x - \frac{(v_{1y} - v_{0y})}{blkW}y + v_{0x} \\ v_y = \frac{(v_{1y} - v_{0y})}{blkW}x + \frac{(v_{1x} - v_{0x})}{blkW}y + v_{0y} \end{cases}$$

**[0167]** The subblock MV at $i_{th}$ row and $j_{th}$ column is derived as

$$\begin{cases} v_x = \frac{(v_{1x} - v_{0x})}{blkW}\left(j * sbW + \frac{sbW}{2}\right) - \frac{(v_{1y} - v_{0y})}{blkW}(i * sbH + \frac{sbH}{2}) + v_{0x} \\ v_y = \frac{(v_{1y} - v_{0y})}{blkW}(j * sbW + \frac{sbW}{2}) + \frac{(v_{1x} - v_{0x})}{blkW}(i * sbH + \frac{sbH}{2}) + v_{0y} \end{cases}$$

**[0168]** Prediction refinement for affine mode is now described. After the sub-block based affine motion compensation is performed, the prediction signal can be refined by adding an offset derived based on the pixel-wise motion and the gradient of the prediction signal. The offset at location $(m, n)$ can be calculated as:

$$\Delta I(m,n) = g_x(m,n) * \Delta v_x(m,n) + g_y(m,n) * \Delta v_y(m,n)$$

wherein $g_x(m,n)$ is the horizontal gradient and $g_y(m,n)$ is the vertical gradient of the prediction signal, respectively. $\Delta v_x(m,n)$ and $\Delta v_y(m,n)$ are the differences in x and y components between the motion vector calculated at location pixel location $(m,n)$ and the subblock MV. Let the coordinate of the top-left sample of the subblock be (0,0), the center of the subblock is $(\frac{sbW}{2}, \frac{sbH}{2})$. Given the affine motion parameters a, b, c, and d, $\Delta v_x(m,n)$ and $\Delta v_y(m,n)$ can be derived as:

$$\Delta v_x(m,n) = a * (\mathrm{m} - \frac{sbW}{2}) + b * (\mathrm{n} - \frac{sbH}{2})$$

$$\Delta v_y(m,n) = c * (\mathrm{m} - \frac{sbW}{2}) + d * (\mathrm{n} - \frac{sbH}{2})$$

**[0169]** In the control-points based affine motion model, the affine motion parameters a, b, c, and d are calculated from the CPMVs as

$$a = \frac{(v_{1x} - v_{0x})}{blkW}$$

$$b = \frac{(v_{2x} - v_{0x})}{blkH}$$

$$c = \frac{(v_{1y} - v_{0y})}{blkW}$$

$$d = \frac{(v_{2y} - v_{0y})}{blkH}$$

**[0170]** Affine merge mode is now described. In affine merge mode of VVC, the CPMVs of the current CU is generated based on the motion information of the spatial neighboring CUs. There can be up to five candidates and an index is signalled to indicate the one to be used for the current CU. The following three types of candidate are used to form the affine merge candidate list: Inherited affine merge candidates that extrapolated from the CPMVs of the neighbour CUs; Constructed affine merge candidates that are derived using the translational MVs of the neighbour CUs; and Zero MVs.

**[0171]** FIG. 11 is a conceptual diagram illustrating an example of control point motion vector inheritance. In VVC, when a neighboring affine CU is identified, its control point motion vectors are used to derived the inherited affine merge candidate in the affine merge

list of the current CU. As shown in FIG. 11, if the neighbor left bottom block 402 is coded in affine mode, the motion vectors $v_2$ , $v_3$ and $v_4$ of the top left corner, above right corner and left bottom corner of the CU 404 which contains block 402 are attained. When block 402 is coded with 4-parameter affine model, the two CPMVs of current CU 400 are calculated according to $v_2$, and $v_3$. In case that block 402 is coded with 6-parameter affine model, the three CPMVs of current CU 400 are calculated according to $v_2$ , $v_3$ and $v_4$.

**[0172]** Constructed affine candidate means the candidate is constructed by combining the neighbor translational motion information of each control point. FIG. 12 is a conceptual diagram illustrating example locations of candidate positions for constructed affine merge mode. The motion information for the control points is derived from the specified spatial neighbors and temporal neighbor of a current block 406 shown in FIG. 12. CPMVk (k=1, 2, 3, 4) represents the k-th control point. For CPMV1, the B2->B3->A2 blocks are checked and the MV of the first available block is used. For CPMV2, the B1->B0 blocks are checked and for CPMV3, the A1->A0 blocks are checked. TMVP is used as CPMV4 if TMVP is available. The following combinations of control point MVs are used to construct affine merge candidates in a given order with at most 6 different candidates: {CPMV1, CPMV2, CPMV3}, {CPMV1, CPMV2, CPMV4}, {CPMV1, CPMV3, CPMV4}, {CPMV2, CPMV3, CPMV4}, { CPMV1, CPMV2}, { CPMV1, CPMV3}. The combination of 3 CPMVs constructs a 6-parameter affine merge candidate and the combination of 2 CPMVs constructs a 4-parameter affine merge candidate. To avoid a motion scaling process, if the reference indices of control points are different, the related combination of control point MVs may be discarded.

**[0173]** Video encoder 200 or video decoder 300 may also add zero MVs to the subblock merge candidate list, after inherited affine merge candidates and constructed affine merge candidate are checked, if the list is not full. Zero MVs are inserted until the list is full.

**[0174]** Affine AMVP mode is now described. In VVC, affine flag in coding unit (CU) level is signalled in the bitstream to indicate whether affine AMVP mode is used and then another flag is signalled to indicate whether 4-parameter affine or 6-parameter affine. In affine AMVP mode, the motion vector difference (MVD) between the CPMVs of current CU and their predictors CPMVPs is signalled in the bitstream together with the index of predictors and the index of the selected reference picture for each of the applicable prediction direction. In case of 4-parameter affine, two MVDs are signaled per applicable prediction direction. In case of 6-paramter affine, three MVDs are signaled per applicable prediction direction. When coding the 2$^{nd}$ and 3$^{rd}$ (in case of 6-paramter affine) MVD is

further predicted by the 1st MVD. Therefore, the difference between 2nd and 1st MVD instead of the 2nd MVD is signaled in the bitstream, and the difference between 3rd and 1st MVD in stead of the 3rd MVD is signaled in the bitstream for the 6-parater affine. Note that inter prediction direction is signaled beforehand to indicate whether it is bi-prediction, uni-prediction from reference picture list 0 or uni-prediction from reference picture list 1.

**[0175]** In VVC, the affine AMVP candidate list size is generated by using the following four types of CPMVP candidate in order: 1) Inherited affine AMVP candidates that extrapolated from the CPMVs of the neighbor CUs; 2) Constructed affine AMVP candidates CPMVPs that are derived using the translational MVs of the neighbor CUs; 3) Translational MVs from neighboring CUs; and 4) Zero MVs.

**[0176]** The checking order of inherited affine AMVP candidates is same to the checking order of inherited affine merge candidates. The only difference is that, for AMVP candidate, only the affine CU that has the same reference picture as in current block is considered. No pruning process is applied when inserting an inherited affine motion predictor into the candidate list.

**[0177]** A constructed AMVP candidate is derived from the specified spatial neighbors. The same checking order is used as done in affine merge candidate construction. In addition, reference picture index of the neighboring block is also checked. The first block in the checking order that is inter coded and has the same reference picture as in current CUs is used. There is only one when the current CU is coded with 4-parameter affine mode, and $mv_0$ and $mv_1$ are both available, they are added as one candidate in the affine AMVP list. When the current CU is coded with 6-parameter affine mode, and all three CPMVs are available, they are added as one candidate in the affine AMVP list. Otherwise, constructed AMVP candidate is set as unavailable.

**[0178]** If a number of affine AMVP list candidates is still less than maximum number after valid inherited affine AMVP candidates and constructed AMVP candidate are inserted, $mv_0$, $mv_1$, and $mv_2$ will be added, in order, as the translational MVs to predict all control point MVs of the current CU, when available. Finally, zero MVs are used to fill the affine AMVP list if it is still not full.

**[0179]** Linear regression based affine merge candidates are now described. In ECM6.0, linear regression based affine merge candidate derivation method proposed in Zhang, et al., "EE2-2.1: Regression based affine candidate derivation," Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 27th Meeting: by

teleconference, 13-22 July 2022, JVET- AA0107 was adopted. In the proposal, two types of linear regression based affine merge candidates are derived, the non-refined and refined candidates. For both types of candidates, the derivation process is the same with only different sub-block motion information are used as the input. Y (hereinafter "VVC Draft 10").

**[0180]** FIG. 13 is a conceptual diagram illustrating example non-adjacent spatial neighboring blocks used to derive non-adjacent affine candidates. In the example of FIG. 13, current block 420 is depicted, along with adjacent neighbors 422 (shown with cross hatching) and non-adjacent neighbors 424 (shown with shading). For the non-refined candidates, only the sub-block motion information from a non-adjacent affine CU is used as the input to the linear regression process. A non-adjacent affine CU would be a non-adjacent neighbor 424 that is coded using affine mode.

**[0181]** FIG. 13 shows an example of inputs to the linear regression process to derive the non-refined linear regression based affine merge candidates. As described above with respect to BDOF, certain scan patterns may be used in searching for non-adjacent affine CUs. Once a non-adjacent affine CU is identified, as in the example of FIG. 13, each of a sub-block's motion information including the sub-block's motion vectors denoted by $\{(mv_{x0}, mv_{y0}), (mv_{x1}, mv_{y1}), \ldots, (mv_{xN-1}, mv_{yN-1})\}$ and central coordinates denoted by $\{(x_0, y_0), (x_1, y_1), \ldots, (x_{N-1}, y_{N-1})\}$ are input to the linear regression process to derive non-refined affine merge candidates.

**[0182]** FIG. 14 is a conceptual diagram illustrating an example of sub-block motion information used to derive refined candidates. For the refined candidates, in addition to the motion information from sub-blocks 430 in the non-adjacent affine CU (as shown in FIG. 13), motion information from the template sub-blocks 432 may additionally be included as input to the linear regression process.

**[0183]** The linear regression process for deriving both the non-refined as well as the refined candidates are the same which follows the mathematical derivation as explained above. The only difference is which of the sub-blocks' information should be used as the input to the linear regression process. For example, video encoder 200 or video decoder 300 may employ such a linear regression process for deriving both the non-refined, as well as the refined candidates.

**[0184]** Bilateral matching AMVP-merge mode in ECM is now described. The bi-directional predictor is composed of an AMVP predictor in one direction and a merge predictor in the other direction. The mode can be enabled to a coding block when the

selected merge predictor and the AMVP predictor satisfy DMVR condition, where there is at least one reference picture from the past and one reference picture from the future relatively to the current picture and the distances from two reference pictures to the current picture are the same, the bilateral matching MV refinement is applied for the merge MV candidate and AMVP MVP as a starting point. Otherwise, if template matching functionality is enabled, template matching MV refinement is applied to the merge predictor or the AMVP predictor which has a higher template matching cost. The pipeline of AMVP-merge mode is illustrated in FIG. 15.

**[0185]** FIG. 15 is a flowchart illustrating an example of AMVP-merge mode for non-LDC picture techniques. In FIG. 15, the AMVP-merge mode process begins at 500. Then video encoder 200 and video decoder 300 may construct a reference picture pair for the AMVP-merge mode (502). Video encoder 200 and video decoder 300 may generate an AMVP candidate list (504) for one prediction direction, and may generate a merge candidate list (506) for the other prediction direction. In some examples, video encoder 200 and video decoder 300 may perform bilateral matching-based merge candidate list reordering (508) on the AMVP candidate list and/or the merge candidate list. Video encoder 200 and video decoder 300 may perform bilateral matching-based refinement (510) if the candidates have equal POC distance, and may perform template matching-based refinement if the candidates have unequal POC distance (512). The term "true-bi equal POC distance" refers to the situation where one reference picture has a POC that is less than the POC of the current picture, and the other reference picture has a POC that is greater than the POC of the current picture.

**[0186]** AMVP part of the mode is signaled as a regular uni-directional AMVP, i.e., reference index and MVD are signaled, and it has a derived MVP index if template matching is used or MVP index is signaled when template matching is disabled.

**[0187]** For AMVP direction LX, X can be 0 or 1, the merge part in the other direction (1 – LX) is implicitly derived by minimizing the bilateral matching cost between the AMVP predictor and a merge predictor, i.e., for a pair of the AMVP and a merge motion vectors. For every merge candidate in the merge candidate list which has that other direction (1 – LX) motion vector, the bilateral matching cost is calculated using the merge candidate MV and the AMVP MV. The merge candidate with the smallest cost is selected. The bilateral matching refinement is applied to the coding block with the selected merge candidate MV and the AMVP MV as a starting point.

**[0188]** The third pass of multi pass DMVR which is 8x8 sub-PU BDOF refinement of the multi-pass DMVR is enabled to AMVP-merge mode coded block.

**[0189]** The mode is indicated by a flag, if the mode is enabled AMVP direction LX is further indicated by a flag.

**[0190]** Local illumination compensation (LIC) is now described. LIC is an inter prediction technique to model local illumination variation between current block and its prediction block as a function of that between current block template and reference block template. The parameters of the function can be denoted by a scale $\alpha$ and an offset $\beta$, which forms a linear equation, that is, $\alpha$*p[x]+$\beta$ to compensate illumination changes, where p[x] is a reference sample pointed to by MV at a location x on reference picture. Since $\alpha$ and $\beta$ can be derived based on current block template and reference block template, no signaling overhead is required for them, except that an LIC flag is signaled for AMVP mode to indicate the use of LIC.

**[0191]** The local illumination compensation proposed in Seregin et al, "CE4-3.1a and CE4-3.1b: Unidirectional local illumination compensation with affine prediction," Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 15th Meeting: Gothenburg, 3-12 July 2019, JVET-O0066 is used for uni-prediction inter CUs with the following modifications: Intra neighbor samples can be used in LIC parameter derivation; LIC is disabled for blocks with less than 32 luma samples; For both non-subblock and affine modes, LIC parameter derivation is performed based on the template block samples corresponding to the current CU, instead of partial template block samples corresponding to first top-left 16x16 unit; Samples of the reference block template are generated by using MC with the block MV without rounding it to integer-pel precision.

**[0192]** In Xiu et al., "EE2-Test2.7: Improvements on local illumination compensation," Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 30th Meeting: Antalya, 21-28 April 2023, JVET-AD0213, LIC mode is extended to bi-predictive CUs and is adopted into ECM, where two different linear models are applied to the two prediction blocks which are then combined to generate the bi-prediction samples of the current CU, i.e.,

$$P'[x,y] = (1-\omega)\cdot p_0'[x,y] + \omega \cdot p_1'[x,y]$$

and

$$p_0'[x,y] = \alpha_0 \cdot P_0[x,y] + \beta_0$$
$$p_1'[x,y] = \alpha_1 \cdot P_1[x,y] + \beta_1$$

where $\alpha_0$ and $\beta_0$, and $\alpha_1$ and $\beta_1$ indicate the scales and the offsets in L0 and L1, respectively; $\omega$ indicates the weight (as indicated by the CU-level BCW index) for the weighted combination of L0 and L1 predictions.

**[0193]** The method firstly derives the L0 parameters by minimizing difference between L0 template prediction $T_0$ and the template $T$ and the samples in $T$ are updated by subtracting the corresponding samples in $T_0$. Then, the L1 parameters are calculated that minimizes the difference between L1 template prediction $T_1$ and the updated template. Finally, the L0 parameter is refined again in the same way.

**[0194]** Following the current LIC design, one flag is signaled for AMVP bi-predicted CUs for the indication of the LIC mode while the flag is inherited for merge related inter CUs. Additionally, the LIC is disabled for DMVR and BDOF.

**[0195]** In ECM, the derived LIC model parameter is stored in the CUs since when performing overlapped block motion compensation (OBMC), the LIC model parameter will additionally be compared to decide whether the OBMC need to be performed. Hence, for a CU with LIC flag equals to true, a set of LIC model parameters is stored and available for future usage.

**[0196]** In Xiu et al., "Non-EE2: Enhancements on local illumination compensation," Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 32nd Meeting: Hanover, 13-20 October 2023, JVET-AF0191, non-local illumination compensation (NLIC) is proposed. For this method, instead of using the template samples, the samples of the previously coded CUs are utilized for deriving the linear model used for the motion compensation of the current block. Specifically, after the reconstruction of each inter CU (except for geometry-based prediction mode (GPM) and SbTMVP CUs), one linear model is derived by minimizing the difference between the reconstruction and prediction samples of the block. The derived LIC model parameters are also stored. This LIC model is derived irrespective of the LIC flag value of the block. If the LIC flag is true for the CU, then two set of LIC model parameters are stored. One set is derived between the current template and reference template. Another set is derived by minimizing the difference between the reconstruction and prediction samples of the block. If the LIC flag is false, then only set of LIC parameters are stored which is derived between the reconstruction and prediction samples of the block.

**[0197]** In Zhang et al., "EE2-3.2: LIC flag derivation for merge candidates with template costs," Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 32nd Meeting: Hanover, 13-20 October 2023, JVET-AF0128, a template

matching cost based LIC flag derivation method is proposed and is adopted to the ECM reference software. In these techniques, the template matching cost is computed twice for the same merge candidate with LIC flag set to true or false each time. The two template matching costs will be compared, and a predefined threshold is used to decide whether the LIC flag will be modified. Currently this method is only applied to uni-predicted merge candidates.

**[0198]** In a draft ECM standard, an MVP can be derived from a spatial neighbor, a temporal neighbor with scaling, a constructive motion vector with an average of two motion vectors, or a zero motion vector.

**[0199]** The current techniques of deriving an MVP do not fully consider a motion vector using both a spatial and a temporal neighbor. The current techniques of deriving an MVP also do not fully consider a motion vector that is from an already decoded picture. In some cases, an MVP candidate is derived from a constructive motion vector or a zero motion vector, which is not efficient to model the temporal information for an inter predicted block.

**[0200]** In this disclosure, various techniques are described that may improve motion vector prediction derivation. The techniques of this disclosure may apply to certain types of slices or pictures. In one example, the techniques are only applied to one or more of a random-access (RA), a low-delay B (LDB), and/or a low-delay P (LDP) slice and/or picture.

**[0201]** Intersecting motion vector prediction is now described. Video encoder 200 or video decoder 300 may determine an intersecting motion vector using the techniques described herein. In one example, given a W×H block with a top-left position of (X, Y) at a current picture, an intersecting motion vector for the block is defined as a pair of motion vectors including a source motion vector and a destination motion vector. The source motion vector (srcItcMvHor, srcItcMvVer) is a motion vector located at an already decoded picture A. The destination motion vector (dstItcMvHor, dstItcMvVer) is a motion vector located at an already decoded picture B. A source block (srcRefBlk) (e.g., a W×H source block) related to the source motion vector with top-left position of (srcX, srcY) is located at picture A and has a predictor where the predictor is a subset of a destination block (dstRefBlk) (e.g., a W×H destination block) related to the destination motion vector with top-left position of (dstX, dstY) located at picture B. For example, a portion of the destination block may be a predictor for the source block. At least one sample position of the source block is determined based on the source motion vector, and

at least one sample position of the destination block is determined based on the destination motion vector. For example, srcX is equal to X + srcItcMvHor, srcY is equal to Y + srcItcMvVer, dstX is equal to X + dstItcMvHor, dstY is equal to Y + dstItcMvVer, wherein the block size and the position are in a same precision as the motion vector. In one example, the precision is set to 1/16-pel. The srcRefBlk may be an inter prediction block. For example, the predictor may be a subset of a destination block in that the predictor may be located in the destination block and be smaller than a full size of the destination block. That is, the source block (e.g., a part of the source block) is predicted at least by the subset of the W×H destination block. For example, a source block may be an inter block, which has a motion vector that points to the destination block. In other words, the source block has a predictor that is derived from the destination block by using motion compensation with respect to the motion vector which may be in a fractional pel precision, such as in 1/16-pel precision.

**[0202]** FIG. 16 is a conceptual diagram illustrating an example of an intersecting MV. Current block 600 is located in current picture 610. Source reference picture A 612 may be a picture that is already decoded (e.g., decoded prior to current block 600) and is temporally prior to current picture 610. For example, source reference picture A 612 may have a lower POC count than current picture 610. Source reference picture A 612 includes source block 602.

**[0203]** Destination reference picture B 614 may be a picture that is already decoded (e.g., decoded prior to current block 600) and is temporally after current picture 610. For example, destination reference picture B 614 may have a higher POC count than current picture 610. Destination reference picture B 614 includes destination block 604.

**[0204]** Source block 602 may include a predictor that is a subset of destination block 604. Motion vector 622 may be a source motion vector and motion vector 624 may be a destination motion vector. Together motion vector 622 and motion vector 624 may be considered an intersecting motion vector. An intersecting motion vector as defined in the present disclosure may be regarded as having two components: a source motion vector and a destination motion vector. An intersecting motion vector may therefore, in some ways, be regarded as an "intersecting motion vector pair."

**[0205]** A size of source block 602, a size of destination block 604, a position of source block 602, a position of destination block 604, motion vector 622, and motion vector 624 may all have a same precision. For example, a block size of source block 602, a block size of destination block 604, a position of source block 602, and a position of destination

block 604 may be of a same precision as the source motion vector (e.g., motion vector 622) and the destination motion vector (e.g., motion vector 624). In some examples, each of the size of source block 602, the size of destination block 604, the position of source block 602, the position of destination block 604, motion vector 622, and motion vector 624 may have a $1/16^{th}$ pel precision.

**[0206]** Given a W×H source block 602 (srcRefBlk) with top-left position of (srcX, srcY) located at source reference picture A 612, source reference picture A 612 is from the past with respect to current picture 610, the srcRefBlk is a inter prediction block and has a motion vector (srcHor, srcVer) located at destination reference picture B 614, destination reference picture B 614 is from the future with respect to current picture 610, the temporal distance between source reference picture A 612 and current picture 610 is pocDistAtoCur, and the temporal distance between source reference picture A 612 and destination reference picture B 614 is pocDistAtoB.

**[0207]** The position of (X, Y), the motion vector of (srcItcMvHor, srcItcMvVer) and (dstItcMvHor, dstItcMvVer) for current block 600 are derived as follows:

Deriving the top-left position (dstX, dstY) of dstRefBlk as dstX = srcX + srcHor and dstY = srcY + srcVer.

Deriving the top-left position (X, Y) of the current block by solving the temporal scale equation of: $\frac{Y-srcY}{dstY-srcY} = \frac{pocDistAtoCur}{pocDistAtoB}$ and $\frac{X-srcX}{dstX-srcX} = \frac{pocDistAtoCur}{pocDistAtoB}$

Video encoder 200 or video decoder 300 may derive the portion of the intersecting motion vector located at reference picture A (e.g., the source motion vector) as: srcItcMvHor = srcX – X and srcItcMvVer = srcY – Y. Video encoder 200 or video decoder 300 may derive the portion of the intersecting motion vector located at reference picture B (e.g., the destination motion vector) as: dstItcMvHor = dstX – X and dstItcMvVer = dstY – Y. X and Y may be rounded to be pel-precision to represent the current block top-left position. In one example, the pel-precision of X and Y are further rounded to be a multiple of a minimum block size, e.g. X is a multiple of 4 and Y is a multiple of 4.

**[0208]** For a current block (or subblock), a set of intersecting motion vectors is derived by checking all srcRefBlk (from top-left to bottom-right of a picture) in all decoded reference pictures. For example, video encoder 200 or video decoder 300 may derive the set of intersecting motion vectors.

**[0209]** In one example, picture A and B (source reference picture A 612 and destination reference picture B) has at least one motion vector from the past to the current picture and at least one motion vector from the future to the current picture.

**[0210]** In one example, block width W is equal to the value of a predefined minimum block size, e.g. 4. In one example, block height H is equal to the value of a predefined minimum block size, e.g. 4.

**[0211]** For example, video encoder 200 or video decoder 300 may determine, for each of a plurality of subblocks (e.g., 4x4 subblocks) of a current block of a current picture of the video data, a corresponding intersecting motion vector list. Each corresponding intersecting motion vector list may include at least one intersecting motion vector. The at least one intersecting motion vector may include a source motion vector associated with a source block located in a first reference picture and a destination motion vector associated with a destination block located in a second reference picture. The source block may have a predictor that is a subset of the destination block. The first reference picture may already be decoded and temporally prior to the current picture in display order and the second reference picture be already decoded and temporally after the current picture in display order. Video encoder 200 or video decoder 300 may determine an intersecting motion vector for the current block based on the corresponding intersecting motion vector lists. Video encoder 200 or video decoder 300 may decode the current block based on the intersecting motion vector.

**[0212]** In one example, at least P% of the srcRefBlk predictors are from the dstRefBlk, e.g., when P is equal to 50, at least half of the srcRefBlk predictors are from the dstRefBlk.

**[0213]** In some examples, video encoder 200 or video decoder 300 may determine whether an intersecting motion vector is valid. In one example, when both the source MV and the destination MV are in a reference picture that is used as the reference picture for the current picture, the intersecting MV is valid. For example, video encoder 200 or video decoder 300 may determine which intersecting motion vectors are valid. A corresponding intersecting motion vector list may include intersecting motion vectors determined to be valid and may not include intersecting motion vectors determined to not be valid (e.g., determined to be invalid).

**[0214]** In one example, given a picture size of W×H, and an offset of T_W and T_H, the picture boundary is extended as (-T_W, -T_H) at top-left, (W + T_W, -T_H) at top-right, (-T_W, H + T_H) at bottom-left and (W + T_W, H + T_H) at bottom-right. When both the source MV and the destination MV are in the extended picture boundary, the

intersecting MV is valid. For example, give an intersecting MV (srcItcMvHor, srcItcMvVer) and (dstItcMvHor, dstItcMvVer), when -T_W < srcItcMvHor < W + T_W and -T_H < srcItcMvVer < H + T_H and -T_W < dstItcMvHor < W + T_W and -T_H < dstItcMvVer < H + T_H, the intersecting MV is valid. For example, video encoder 200 or video decoder 300 may determine a first intersecting MV is valid when the first intersecting MV includes both a first source MV and a first destination MV in an extended picture boundary. For example, the when both the first source MV and a first destination MV are anywhere inside the extended picture boundary, the first intersecting MV is valid. In one example, when a source MV (srcHor, srcVer) is from a srcRefBlk and the srcRefBlk is a uni-prediction inter block, the derived intersecting MV is valid.

**[0215]** In one example, given an intersecting MV (srcItcMvHor, srcItcMvVer) and (dstItcMvHor, dstItcMvVer), when the absolute value of srcItcMvHor and dstItcMvHor is greater than K, and/or the absolute value of srcItcMvVer and dstItcMvVer is greater than L, the intersecting MV is valid. For example, the value of K and L are equal to 1, in this case, a zero MV, e.g. (0,0) for srcItcMv and (0,0) for dstItcMv will be determined as a not valid (e.g., invalid) intersecting MV. For example, video encoder 200 or video decoder 300 may determine the first intersecting MV is valid when a first absolute value of each of a first horizontal component of the first source MV and a first horizontal component of the first destination MV is greater than a horizontal threshold (e.g., the absolute value of the horizontal component of the first source MV is greater than the horizontal threshold and the absolute value of the horizontal component of the first destination MV is also greater than the horizontal threshold). For example, video encoder 200 or video decoder 300 may determine the first intersecting MV is valid when a second absolute value of each of a first vertical component of the first source MV and a first vertical component of the first destination MV is greater than a vertical threshold (e.g., the absolute value of the vertical component of the first source MV is greater than the vertical threshold and the absolute value of the vertical component of the first destination MV is also greater than the vertical threshold)

**[0216]** In one example, when a set of intersecting MVs is derived for a current block, two intersecting MVs must have a difference that is greater than a threshold T. For example, given two intersecting MVs for a current block, (srcItcMv1, dstItcMv1) including (srcItcMvHor1, srcItcMvVer1) and (dstItcMvHor1, dstItcMvVer1), and (srcItcMv2, dstItcMv2) including (srcItcMvHor2, srcItcMvVer2) and (dstItcMvHor2, dstItcMvVer2), wherein srcItcMv1 and srcItcMv2 are both from reference picture A and dstItcMv1 and

dstItcMv2 are both from reference picture B, when both of the two intersecting MVs are valid, the following condition must be met:

In one example, absolute(srcItcMvHor1 - srcItcMvHor2) > T and absolute(srcItcMvVer1 - srcItcMvVer2) > T and absolute(dstItcMvHor1 - dstItcMvHor2) > T and absolute(dstItcMvVer1 - dstItcMvVer2) > T.

In one example, absolute(srcItcMvHor1 - srcItcMvHor2) > T or absolute(srcItcMvVer1 - srcItcMvVer2) > T or absolute(dstItcMvHor1 - dstItcMvHor2) > T or absolute(dstItcMvVer1 - dstItcMvVer2) > T. In some examples the thresholds above may be different or the same. For example, absolute(srcItcMvHor1 - srcItcMvHor2) > T1 and absolute(srcItcMvVer1 - srcItcMvVer2) > T2 and absolute(dstItcMvHor1 - dstItcMvHor2) > T3 and absolute(dstItcMvVer1 - dstItcMvVer2) > T4. In one example, absolute(srcItcMvHor1 - srcItcMvHor2) > T1 or absolute(srcItcMvVer1 - srcItcMvVer2) > T2 or absolute(dstItcMvHor1 - dstItcMvHor2) > T3 or absolute(dstItcMvVer1 - dstItcMvVer2) > T4. T1, T2, T3 and T4 may be the same or different

**[0217]** For example, video encoder 200 or video decoder 300 may determine the first intersecting MV is valid when any or all of, for a pair of intersecting MVs including the first intersecting MV and a second intersecting MV, a) a difference between the first horizontal component of the first source MV and a second horizontal component of a second source MV of the second intersecting MV is greater than a difference threshold, b) a difference between the first vertical component of the first source MV and a second vertical component of the second source MV is greater than the difference threshold, c) a difference between the first horizontal component of the first destination MV and a second horizontal component of a second destination MV of the second intersecting MV is greater than the difference threshold, and/or d) a difference between the first vertical component of the first destination MV and a second vertical component of the second destination MV is greater than the difference threshold.

**[0218]** Intersecting MV generation for a picture and/or slice is now described. Given a W×H block with a top-left position of (X, Y) at a current picture, there are N intersecting MVs determined to be valid as described above, in this example, the K intersecting MVs are further determined to be valid and stored for the block. For example, video encoder 200 or video decoder 300 may determine that N intersecting MVs are valid and store K intersecting MVs that are valid. In the context of the present disclosure, storing an intersecting MV (or list) may entail storing the intersecting MV (or list) for possible subsequent use in decoding the current block. In other words, non-

stored intersecting MVs (or lists) may be discarded and play no part in decoding the current block.

**[0219]** In one example, the N intersecting MVs are sorted based on BM cost, where an intersecting MV with a smaller BM cost is sorted before an intersecting MV with a larger BM cost. In one example, the BM cost is derived as an absolute difference between a W×H block with top-left position of (srcX, srcY) at a reference picture A (e.g., source reference picture A 612) and a W×H block with top-left position of (dstX, dstY) at a reference picture B (e.g., destination reference picture B 614). In one example, the BM cost is derived as an absolute difference between a W×H block with an MV (srcItcMvHor, srcItcMvVer) offset to the top-left position of the current block (X, Y) at reference picture A (e.g., source reference picture A 612) and a W×H block with a motion vector (dstItcMvHor, dstItcMvVer) offset to the top-left position of the current block (X, Y) at reference picture B (e.g., destination reference picture B 614), wherein, the block is derived from bilinear interpolation using srcItcMv and dstItcMv on reference picture A and B, respectively. The first K intersecting MVs in the sorted intersecting MV list are further determined to be valid and stored for the block. Video encoder 200 or video decoder 300 may sort the intersecting MV list and store valid intersecting MVs as described above. For example, as part of storing the corresponding intersecting motion vector list, video encoder 200 or video decoder 300 may store, for a first corresponding intersecting motion vector list, only a first number of intersecting motion vectors appearing first in the first corresponding intersecting motion vector list, the first number of intersecting motion vectors being less than a number of intersecting motion vectors in the first corresponding intersecting motion vector list. For example, video encoder 200 or video decoder 300 may store only a subset of the intersecting motion vectors appearing in a corresponding intersecting motion vector list, such as the first corresponding intersecting motion vector list, rather than storing all of the intersecting motion vectors appearing in the corresponding intersecting motion vector list.

**[0220]** For example, video encoder 200 or video decoder 300 may sort valid intersecting motion vectors of a corresponding subblock (e.g., a 4x4 subblock) based on BM cost so as to place an intersecting motion vector having a lower BM cost in a position closer to a front of the corresponding intersecting motion vector list than an intersecting motion vector having a higher BM cost.

**[0221]** In one example, the N intersecting MVs are sorted based on occurrence. For example, video encoder 200 or video decoder 300 may sort the N intersecting MVs based

on a number of times each intersecting MV may appear, for example, in a MV list. For example, during the derivation of the N intersecting MVs, when a new intersecting MV with srcItcMv and dstItcMv at reference picture A and B, respectively, is identical to an intersecting MV already in the list, or the difference between srcItcMv and dstItcMv to an intersecting Mv already in the list is less than a threshold, an occurrence count of the intersecting MV already in the list may be incremented by 1. The N intersecting MVs may be sorted based on the occurrence count and an intersecting MV with a larger occurrence count may be sorted to appear before an intersecting MV with a smaller occurrence count. In some examples, the first K intersecting MVs in the sorted intersecting MV list are further determined to be valid and stored for the block. Video encoder 200 or video decoder 300 may sort the intersecting MV list and store the first K valid intersecting MVs. For example, an intersecting MV list may include more than one occurrence of the same intersecting MV. For example, video encoder 200 or video decoder 300 may sort intersecting MVs in the intersecting MV list based on occurrence, such that an MV appearing 4 times is placed before an MV appearing twice in the sorted intersecting MV list.

**[0222]** For example, video encoder 200 or video decoder 300 may sort valid intersecting motion vectors of a corresponding subblock (e.g., a 4x4 subblock) based on an occurrences count of each intersecting motion vector in the corresponding intersecting motion vector list so as to place an intersecting motion vector having a higher occurrence count in a position closer to a front of the corresponding intersecting motion vector list than the intersecting motion vector having a lower occurrence count.

**[0223]** Intersecting MV derivation for a block is now described. Video encoder 200 or video decoder 300 may derive intersecting MVs for a block as described herein.

**[0224]** In this example, when a block is determined to be an inter prediction block, the MV used for prediction of the block is an intersecting MV. The intersecting MV used for prediction of the block is determined by an index which indicates an intersecting MV from an intersecting MV list for the block, where the intersecting MV list is derived from getting all intersecting MVs of the subblocks that are covered by the current block. In one example, the intersecting MVs are generated for a picture and/or slice in every W×H block, wherein, W is equal to 4 and H is equal to 4, given a current block with a size (4*M) × (4*N), there are M×N blocks of generating intersecting MVs, denoted as IctList0, IctList1, … , IctListK, where K is equal to (M×N - 1). The total number of the

intersecting MVs of the IctList0 to IctListK is P, and the total number of intersecting MVs for the current block is Q, wherein Q is equal or smaller than P.

**[0225]** For example, video encoder 200 or video decoder 300, as part of determining the intersecting motion vector, may determine an intersecting motion vector list for the current block based on all of the corresponding intersecting motion vector lists and determine a value of an index, the value of the index indicative of the intersecting motion vector in the intersecting motion vector list for the current block. In some examples, a number of intersecting motion vectors in the intersecting motion vector list for the current block (Q) is less than a total number of intersecting motion vectors in the corresponding intersecting motion vector lists (P).

**[0226]** FIG. 17 is a conceptual diagram illustrating an example of a 16×8 block having IctList0 to IctList7. Current picture 710 includes current block 700. Current block 700 has a block size of 16x8. For example, the length of current block 700 is 16 samples and the height of current block 700 is 8 samples. Current block 700 is depicted having 4x4 subblocks within current block 700. Each 4x4 subblock stores a respective intersecting MV list (e.g., an ICT (intersecting motion vector) list of ICT list0-ICT list7). In some examples, 4x4 is the minimum subblock size to store a respective intersecting MV list.

**[0227]** In some examples, the Q intersecting MVs are selected from P intersecting MVs and ordered as described according to any of the following techniques.

**[0228]** In one example, the P intersecting MVs from IctList0 to IctListK are sorted based on occurrence count (as described above), and the Q intersecting MVs with larger occurrence count are selected for the current block. For example, video encoder 200 or video decoder 300 may sort the P intersecting MVs based on a number of times each intersecting MV may appear in IctList0 to IctListK and select Q intersecting MVs with the largest occurrence count for the current block.

**[0229]** In one example, the P intersecting MVs from IctList0 to IctListK are sorted based on BM cost, and the Q intersecting MVs with a smaller BM cost are selected for the current block. For example, video encoder 200 or video decoder 300 may sort the P intersecting MVs based on BM cost and select Q intersecting MVs with the smallest BM cost for the current block. In one example, the BM cost for each intersecting MV is derived as an absolute difference between two reference blocks of the current block wherein, the reference blocks are derived from motion compensation, e.g., bilinear interpolation, in a reference picture using the intersecting MV. In one example, the BM cost for each intersecting MV of an IctBlk (e.g., a subblock) is already derived and stored

when generating the intersecting MV list for the IctBlk, the BM cost for the current block is estimated by scaling the BM cost of the IctBlk.

**[0230]** In one example, the P intersecting MVs from IctList0 to IctListK are sorted based on both occurrence and BM cost. In one example, an intersecting MV that has larger occurrence and smaller BM cost is prioritized (e.g., placed closer to the beginning) in the list. In one example, an intersecting MV that has a larger occurrence is prioritized, when two intersecting MVs have an equal occurrence, the MV with a smaller BM cost is prioritized in the list. In one example, an intersecting MV having a smaller BM cost is prioritized, when two intersecting MV have an equal BM cost, the MV having a larger occurrence is prioritized.

**[0231]** In one example, the P intersecting MVs from IctList0 to IctListK are sorted based on template matching (TM) cost, and the Q intersecting MVs with smaller TM cost are selected for the current block. For example, video encoder 200 or video decoder 300 may sort the P intersecting MVs based on TM cost and select Q intersecting MVs with the smallest TM cost for the current block. The TM cost for each intersecting MV may be derived as an absolute difference between the neighboring blocks to the left and to the above of the current block and the neighboring blocks to the left and to the above of the reference block, wherein, the reference block is derived from the intersecting MV.

**[0232]** For example, video encoder 200 or video decoder 300 may, as part of determining the intersecting motion vector list for the current block, sort intersecting motion vectors of each of the corresponding intersecting motion vector lists based on at least one of bilateral matching cost, occurrence count, or template matching cost.

**[0233]** Video encoder 200 or video decoder 300 may determine an intersecting MV predicted block.

**[0234]** In one example, an intersecting MV predicted block is a bi-prediction block, which has one motion vector located at a reference picture A in reference picture list 0 and one motion vector located at a reference picture B in reference picture list 1.

**[0235]** In one example, an intersecting MV predicted block is a uni-prediction block, only one motion vector of the intersecting MV at a reference picture in reference list X is used, X is equal to 0 or 1.

**[0236]** In one example, an intersecting MV predicted block is a bi-prediction block, where only one motion vector of the intersecting MV at a reference picture in reference list X is used, wherein X is equal to 0 or 1.

**[0237]** In one example, an intersecting MV predicted block is a bi-prediction block, where one motion vector of an intersecting MV A at a reference picture A in reference list X is used, and one motion vector of an intersecting MV B at a reference picture B in reference list (1 – X) is used, wherein X is equal to 0 or 1. In this example, intersecting MV A may be a different intersecting MV than intersecting MV B.

**[0238]** Video encoder 200 or video decoder 300 may use an intersecting MV predicted block.

**[0239]** In one example, an intersecting MV predicted block is used for an inter AMVP prediction block.

**[0240]** In one example, an intersecting MV predicted block is used for an inter merge prediction block.

**[0241]** In one example, an intersecting MV predicted block is used for an inter AMVP-merge prediction block.

**[0242]** Video encoder 200 or video decoder 300 may add at least one intersecting MV to an MVP list.

**[0243]** In one example, when constructing an MVP list for a block, at least one intersecting MV is added after spatial MVP construction.

**[0244]** In one example, when constructing an MVP list for a block, at least one intersecting MV is added after non-adjacent spatial MVP construction.

**[0245]** In one example, when constructing an MVP list for a block, at least one intersecting MV is added after temporal MVP (TMVP) construction.

**[0246]** In one example, when constructing an MVP list for a block, at least one intersecting MV is added after history MVP (HMVP) construction.

**[0247]** In one example, when constructing an MVP list for a block, at least one intersecting MV is added before zero MVP construction.

**[0248]** In one example, a block located at the top boundary of the current picture and/or slice or a block located at the left boundary of the current picture and/or slice has at least one MVP of an intersecting MV.

**[0249]** In one example, a block has a block width greater than T has at least one intersecting MV as an MVP. For example, a block has a block width greater than T5 has at least one intersecting MV as an MVP

**[0250]** In one example, a block has a block height greater than T has at least one intersecting MV as an MVP. For example, a block has a block height greater than T6 has at least one intersecting MV as an MVP.

**[0251]** In one example, a block has a block width smaller than T has at least one intersecting MV as an MVP. For example, a block has a block width smaller than T7 has at least one intersecting MV as an MVP.

**[0252]** In one example, a block has a block height smaller than T has at least one intersecting MV as an MVP. For example, a block has a block height smaller than T8 has at least one intersecting MV as an MVP. T5, T6, T7, and T8 may be the same or different.

**[0253]** Intersecting MV derivation for a subblock prediction block is now described. Video encoder 200 or video decoder 300 may derive an intersecting MV for subblock prediction.

**[0254]** In this example, given a current block with a size (4*M) × (4*N), there are M×N subblocks of IctBlks already having intersecting MVs lists, denoted as IctList0, IctList1, …, IctListK. The current block is predicted as an inter subblock prediction block. For each M×N subblock, denoted as subblockX, the motion vector is derived from the intersecting Mv list IctListX, wherein X is from 0 to K. In one example, the motion vector of subblockX is indicated by an index from IctListX. In one example, the motion vector of subblockX is the first intersecting MV from IctListX. In one example, a subblockD is determined as a default subblock, where there is at least one intersecting MV in IctListD, when one subblockX has 0 intersectingMv in IctListX, the intersectingMv in IctListD is used for subblockX.

**[0255]** Intersecting MV derivation for an affine model derivation is now described. Video encoder 200 or video decoder 300 may derive intersecting MV(s) for an affine model.

**[0256]** In this example, an inter prediction block is determined to be subblock coded with an affine model, where the affine model is derived from the intersecting MVs. Given a current block with a size (4*M) × (4*N), there are M×N blocks of IctBlks already having intersecting MVs lists, denoted as IctList0, IctList1, … , IctListK. For example, video encoder 200 or video decoder 300 may decode the current block using subblock prediction, wherein each subblock is predicted using a respective intersecting motion vector. In some examples, as part of decoding the current block using subblock prediction, video encoder 200 or video decoder 300 may use an affine model.

**[0257]** In one example, the intersecting MVs in IctListX has an MV located at reference picture A in reference list 0 and reference picture B in reference list 1. In one example, the reference picture A is the first reference picture in reference list0 and reference picture B is the first reference list 1.

**[0258]** The current block may be predicted as an inter subblock prediction block with an affine model. The affine model may be derived as follows: 1) The intersecting MV for each M×N subblock, denoted as subblockX, is derived from the intersecting MV list IctListX denoted as IctMvSubBlkX, wherein X is from 0 to K. In one example, IctMvSubBlkX is indicated by an index from IctListX. In one example IctMvSubBlkX is the first intersecting MV from IctListX. In one example, a subblockD is determined as a default subblock, where there is at least one intersecting MV in IctListD, when one subblockX has 0 intersectingMv in IctListX, the intersectingMv in IctListD is used for subblockX; and/or 2) The MV for each M×N subblock, denoted as subblockX, is derived from the affine model as described above denoted as AffineMvSubBlkX, wherein X is from 0 to K. The affine model is derived to minimize the difference between AffineMvSubBlkX and IctMvSubBlkX, wherein X is from 0 to K. In one example, the affine model is derived as a linear regression based affine model derivation as described above, wherein the IctMvSubBlkX is used as the input to the linear regression process.

**[0259]** Temporal motion vector prediction (TMVP) derivation using a spatial neighbor motion vector is now described. Video encoder 200 or video decoder 300 may perform a TMVP derivation using a spatial neighbor MV. In this example, a TMVP is derived from a collocated block (colBlk) located at the collocated picture (colPic). The top-left position of the colBlk is derived as the top-left position of the current block with an offset, where the offset is an MV (spatialMv) of the spatial neighbor block of the current block in current picture.

**[0260]** In one example, a spatialMv is derived from an adjacent spatial neighbor, e.g., a block sitting to the left or to the above of the current block.

**[0261]** In one example, a spatialMv is derived from a non-adjacent spatial neighbor, e.g., a block having a top-left position with an offset (M, N) to the top-left position of the current block, wherein, M and N are integers with negative values.

**[0262]** In one example, a spatialMv is derived from a merge candidate in a merge candidate list.

**[0263]** In one example, a spatialMv is the MV of the neighbor block and the MV located at a reference picture, which is same as the collocated picture.

**[0264]** In one example, a spatialMv is the MV of the neighbor block and the MV located at a reference picture which is not the collocated picture, in this case, the spatialMv is derived as the temporal scale from the reference picture to the collocated picture.

**[0265]** When the spatial MV is determined, a collocated MV (colMv) on a collocated reference picture (colRefPic) is derived by checking the motion information at the colBlk. The TMVP is derived as the temporal scale of the colMv, wherein the temporal scale rate is determined based on a rate of temporal distance between colPic and colRefPic and temporal distance between the current picture and a current reference picture. In one example, the current reference picture is pre-defined as the first reference picture in the reference list X, wherein, X is 0 or 1. In another example, the current reference picture is derived as described above.

**[0266]** Reference picture derivation is now described. Video encoder 200 or video decoder 300 may derive a reference picture as described herein. In this disclosure, one or more target reference picture derivation techniques are described and may be applied to TMVP, SbTMVP, spatial merge candidates, and/or AMVP candidates. The target reference picture derivation techniques are based on at least one of the reference pictures of the collocated block, the reference picture list of current coding block, and the POC of the collocated picture.

**[0267]** For example, video encoder 200 or video decoder 300 may determine a current reference picture for a current block of a current picture of the video data based on a reference picture of a collocated block, a reference picture list of the current block, and a POC of a collocated picture, wherein the collocated block is collocated with the current block and is in the collocated picture. Video encoder 200 or video decoder 300 may decode the current block based on a motion vector associated with the current reference picture. In some examples, the motion vector is associated with a TMVP, an SbTMVP, a spatial merge candidate, or an AMVP candidate.

**[0268]** In one example, the reference picture of the collocated block is selected as the target reference picture. For example, video encoder 200 or video decoder 300 may, as part of determining the current reference picture, determine a collocated reference picture for the collocated block.

**[0269]** In one example, video encoder 200 or video decoder 300 implementing the target reference picture derivation technique searches one or more reference pictures in the reference picture list(s) and selects the target reference picture with maximum overlap in POC dimension as follows:

```
curPOCMax = std::max(currPOC, currRefPOC);
curPOCMin = std::min(currPOC, currRefPOC);
colPOCMax = std::max(colPOC, colRefPOC);
```

colPOCMin = std::min(colPOC, colRefPOC);

Overlap = min(curPOCMax, colPOCMax) - max(curPOCMin, colPOCMin);

where currPOC is the POC of the current picture, currRefPOC is the POC of the reference picture, colPOC is the POC of the collocated picture, and colRefPOC is the POC of the collocated reference picture. A target reference picture is selected that maximizes Overlap, or maximizes

Overlap * Overlap / (|currPOC – currRefPOC| * |colPOC – colRefPOC|).

**[0270]** For example, video encoder 200 or video decoder 300 may, as part of determining the current reference picture, determine a reference picture from the reference picture list that is associated with a highest number of pictures that overlap pictures between the collocated reference picture and the collocated picture. For example, video encoder 200 or video decoder 300 may determine first pictures that are between the collocated reference picture and the collocated picture in a picture order count (POC). Video encoder 200 or video decoder 300 may determine, for each potential current reference picture, corresponding second pictures that are between the potential current reference picture and the current picture in the POC. Video encoder 200 or video decoder 300 may determine, for each potential current reference picture, a corresponding count of pictures that are both first pictures and corresponding second pictures. Video encoder 200 or video decoder 300 may determine the current reference picture based on the corresponding counts. Video encoder 200 or video decoder 300 may, as part of determining the current reference picture based on the corresponding counts, select, as the current reference picture, the potential current reference picture having a highest number of corresponding counts. In this manner, a better reference picture which may facilitate obtaining a better PU may be selected, thereby improving the coding performance.

**[0271]** For example, video encoder 200 or video decoder 300 may determine a current reference picture to use for the current block based on at least one of the reference pictures of a collocated block, the reference picture list of the current block, and the picture order count (POC) of the collocated picture. For example, video encoder 200 or video decoder 300 may select the current reference picture from the reference picture list of the current block that has the most overlap of pictures between the collocated reference picture and the collocated picture. For example, if the current picture has a POC of 16, the collocated picture has a POC of 14 and the collocated reference picture has a POC of 10, a potential current reference picture with a POC of 15 would have 0 overlap because the difference between the potential current reference picture and the current picture would not include

any of the pictures between POC 10 and POC 14. A potential current reference picture with a POC of 9 would include all of the pictures between POC 10 and POC 14 inclusive and would therefore have an overlap of 5. In the case where multiple reference pictures have a same amount of overlap, video encoder 200 or video decoder 300 may select the current reference picture from the reference picture list based on a minimum POC distance from the current picture. For example, a potential reference picture with a POC of 10 would have the same overlap as a POC of 9, but be closer in POC distance to the current picture.

**[0272]** In one example, video encoder 200 or video decoder 300 implementing the target reference picture derivation technique first selects the reference picture of the collocated block as the target reference picture. If the reference picture of the collocated block is not in a reference picture list of the current block, video encoder 200 or video decoder 300 implementing the proposed technique selects the reference picture with maximum overlap in POC dimension as the target reference picture.

**[0273]** In one example, video encoder 200 or video decoder 300 implementing the target reference picture derivation technique first selects the reference picture of the collocated block as the target reference picture. If the reference picture of the collocated block is not in the reference picture list of the current block, video encoder 200 or video decoder 300 implementing the technique selects the reference picture with same POC distance, e.g., (|currPOC – currRefPOC| = |colPOC – colRefPOC|) or (currPOC – currRefPOC = colPOC – colRefPOC).

**[0274]** In one example, video encoder 200 or video decoder 300 implementing the target reference picture derivation technique first selects the reference picture of the collocated block as the target reference picture. If the reference picture of the collocated block is not in the reference picture list of the current block, video encoder 200 or video decoder 300 implementing the technique selects the reference picture with the same POC distance. If the reference picture of the collocated block is not in the reference picture list of the current block and there is no reference picture with the same POC distance, video encoder 200 or video decoder 300 implementing the technique selects the reference picture with a maximum overlap in POC dimension as the target reference picture.

**[0275]** For example, video encoder 200 or video decoder 300 may determine that no potential current reference picture in the current reference picture list has a same POC distance from the current picture as a POC distance from the collocated picture to the collocated reference picture. Video encoder 200 or video decoder 300 may determine the

reference picture from the reference picture list that is associated with a highest number of pictures that overlap pictures between the collocated reference picture and the collocated picture based on the determination that no potential current reference picture in the current reference picture list has a same POC distance from the current picture as a POC distance from the collocated picture to the collocated reference picture.

**[0276]** In one example, the reference picture derivation technique is applied to at least one of TMVP, SbTMVP, merging candidates, or AMVP candidates.

**[0277]** In one example, the reference picture derivation technique is applied to SbTMVP as additional SbTMVP candidates.

**[0278]** In another example, the technique is only applied to certain types of slices, or pictures. In one example, the technique is only applied to one of more of RA, LDB, LDP slices and/or pictures. For example, video encoder 200 or video decoder 300 may determine that the current picture is a random-access picture, a low-delay B picture, or a low-delay P picture. Video encoder 200 or video decoder 300 may determine the current reference picture as set forth herein based on the determination that the current picture is the random-access picture, the low-delay B picture, or the low-delay P picture. For example, video encoder 200 or video decoder 300 may determine that the current block is a random-access slice, a low-delay B slice, or a low-delay P slice. Video encoder 200 or video decoder 300 may determine the current reference picture as set forth herein based on the determination that the current slice is the random-access slice, the low-delay B slice, or the low-delay P slice.

**[0279]** Adding a motion vector predictor with a different reference picture is now described. Video encoder 200 or video decoder 300 may add an MVP with a different reference picture as described herein. In one example, one or more motion vector predictors (referred to as new MVPs) may be derived from a motion vector predictor that already exists in the list (referred to as reference MVP). The list can be the merge candidate list in merge mode, and/or the list can be the motion vector predictor list for the AMVP mode. The new MVP has at least one different reference picture compared with the reference MVP. At least one of the motion vector (or motion vectors, in the case of bi-prediction) of the new MVP may be derived by scaling the corresponding motion vector of the reference MVP. The scaling factor may be determined according to the differences among the POC (picture order count) of the current picture, the reference picture of the reference MVP, and the reference picture of the new MVP. For example, if the x component of the motion vector in reference MVP is Vx, the POC distance between

current picture and reference picture of the reference MVP is d1, the POC distance between current picture and reference picture of the new MVP is d2, then the x component of the motion vector in the new MVP is Vx * d2 / d1. In one integer implementation, the scale factor distScaleFactor is calculated as Clip3( −4096, 4095, ( tb * tx + 32 ) >> 6 ), wherein Clip3(min, max, x) return a value min if x is less than min and max if x is larger than max. The motion vector value for x/y component mvLX is calculated as Clip3( −32768, 32767, Sign( distScaleFactor * mvLXA ) * ( ( Abs( distScaleFactor * mvLXA ) + 127 ) >> 8 ) ), wherein Sign() returns the sign of the input value and Abs(x) returns the absolute value of x. The motion vector of the new MVP may also be simply set equal to the motion vector of the reference MVP, therefore only the reference picture is different.

**[0280]** Video encoder 200 or video decoder 300 may derive MVP(s). In one example, one or more MVPs are derived from the top N MVPs that are already existing in the list. For each of the existing MVs, if the existing MV has one reference picture with an associated reference index not equal to a target reference index, a new MVP is derived by setting the reference index equal to the target reference index. The MV may be scaled accordingly. In one example, the target reference index is set to 0. In another example, the process is repeated by setting the target reference index to 0, 1, 2, et al. In an example specifically for a bi-prediction case, this technique is applied only to the MV with non-smaller BCW weights. In an example specifically for a bi-prediction case, this technique is applied only to the MV with non-larger BCW weights.

**[0281]** In one example, the target reference index of a new MVP is set equal to the that of the corresponding reference MVP plus a delta, where the delta value may be ±1, ±2, ±3, etc. When the derived MVP index of the new MVP could point to an existing reference picture in the reference picture list(s), this new MVP candidate may stay in an MVP candidate list; otherwise, the new MVP candidate may be ignored and not included. In addition, the new MVP may be ignored and not included if the same or similar MVPs are already existing in the reference MVP list or in the new MVP candidate list constructed so far. In the case of bi-prediction, two separate delta values may be used, respectively, for the two MVs, and three new MVPs may be added, where the first new MVP takes only the first delta value for its L0 MV and ignores the second delta value, the second new MVP takes only second delta value for its L1 MV and ignores the first delta value, and the third one takes both delta values. In another example for a simplified design, the two delta values could share the same value. In an example specifically for a bi-prediction case, this technique is applied only to the MV with non-smaller BCW

weights. In an example specifically for a bi-prediction case, this technique is applied only to the MV with non-larger BCW weights.

**[0282]** In another example, the new MVPs may be inserted into the list before the zero padding. In another example, the new MVPs may be inserted into the list before pair-wise candidates, history candidates, non-adjacent candidates, or TMVP candidates.

**[0283]** In yet another example, adaptive reordering merge candidates (ARMC) based on TM may be applied after the construction of the list.

**[0284]** In yet another example, ARMC is applied to the new MVPs, and the top M MVPs with lowest TM costs are inserted to the list.

**[0285]** In yet another example, ARMC is applied to the reference MVPs and the new MVPs, and only the best M candidates with lowest template matching costs stay in the final merge list.

**[0286]** In yet another example, the technique is only applicable to certain types of MVPs that are already existing in the list. For example, the reference MVPs can only be spatial MVPs. In another example, the reference MVPs can only be TMVPs. In another example, the reference MVPs can only be non-adjacent MVPs.

**[0287]** In another example, the above-mentioned examples may not be applied to GPM, MHP, or CIIP modes.

**[0288]** FIGS. 18A-18B are flowcharts illustrating example techniques for intersecting motion vector prediction according to one or more aspects of this disclosure. The techniques of FIG. 18A are described with respect to video decoder 300, but may be practiced by any device (e.g., video encoder 200) that is capable of practicing such techniques.

**[0289]** Referring now to FIG. 18A, video decoder 300 may determine, for each of a plurality of subblocks of a current block of a current picture of video data, a corresponding intersecting motion vector list (800). Each corresponding intersecting motion vector list may include at least one intersecting motion vector. The at least one intersecting motion vector may include a source motion vector associated with a source block located in a first reference picture and a destination motion vector associated with a destination block located in a second reference picture. The source block may have a predictor that is a subset of the destination block. The first reference picture may be already decoded and be temporally prior to the current picture in display order and the second reference picture being already decoded and temporally after the current picture

in display order. In some examples, the plurality of subblocks include a plurality of 4x4 subblocks.

**[0290]** Video decoder 300 may determine an intersecting motion vector for the current block based on the corresponding intersecting motion vector lists (802). For example, video decoder 300 may determine an intersecting motion vector from the intersecting motion vectors in the intersecting motion vector lists for the subblocks of the current block.

**[0291]** Video decoder 300 may decode the current block based on the intersecting motion vector (804). For example, video decoder 300 may use the intersecting motion vector to predict the current block and generate a prediction block. Video decoder 300 may combine the prediction block with a residual block to decode the current block.

**[0292]** In some examples, video decoder 300 may store each corresponding intersecting motion vector list. For example, video decoder 300 may separately store, for each subblock, the corresponding intersecting motion vector list.

**[0293]** In some examples, as part of storing each corresponding intersecting motion vector list, video decoder 300 may store, for a first corresponding intersecting motion vector list, only a first number of intersecting motion vectors appearing first in the first corresponding intersecting motion vector list, the first number of intersecting motion vectors being less than a number of intersecting motion vectors in the first corresponding intersecting motion vector list.

**[0294]** In some examples, as part of determining the corresponding intersecting motion vector list, video decoder 300 may determine which intersecting motion vectors are valid. The corresponding intersecting motion vector list may include intersecting motion vectors determined to be valid and not include intersecting motion vectors determined to not be valid (e.g., determined to be invalid).

**[0295]** In some examples, as part of determining the corresponding intersecting motion vector list, video decoder 300 may sort intersecting motion vectors of a corresponding subblock based on bilateral matching cost. In some examples, video decoder 300 may place an intersecting motion vector having a lower bilateral matching cost in a position closer to a front of the corresponding intersecting motion vector list than an intersecting motion vector having a higher bilateral matching cost. Some examples may include sorting intersecting motion vectors based on a bilateral matching cost for each intersecting motion vector that is calculated as a value representative of the degree of difference between samples of the source block and samples destination block defined by the

intersecting MV, the bilateral matching cost being a value representative of the degree of difference between samples of the blocks. The difference metric may use a sum of absolute difference (SAD), sum of squared differences (SSD), mean absolute difference (MAD), mean squared differences (MSD), or other such difference calculations.

**[0296]** In some examples, as part of determining the corresponding intersecting motion vector list, video decoder 300 may sort intersecting motion vectors of a corresponding subblock based on an occurrence count of each intersecting motion vector in the corresponding intersecting motion vector list. In some examples, video decoder 300 may place an intersecting motion vector having a higher occurrence count cost in a position closer to a front of the corresponding intersecting motion vector list than the intersecting motion vector having a lower occurrence count.

**[0297]** In some examples, as part of determining which intersecting motion vectors are valid, video decoder 300 may determine that a first intersecting motion vector is valid. As part of determining that the first intersecting motion vector is valid, video decoder 300 may determine that the first intersecting motion vector comprises both a first source motion vector and a first destination motion vector in an extended picture boundary. As part of determining that the first intersecting motion vector is valid, video decoder 300 may determine that a first absolute value of each of a first horizontal component of the first source motion vector and a first horizontal component of the first destination motion vector that is greater than a horizontal threshold. As part of determining that the first intersecting motion vector is valid, video decoder 300 may determine that a second absolute value of each of a first vertical component of the first source motion vector and a first vertical component of the first destination motion vector is greater than a vertical threshold. As part of determining that the first intersecting motion vector is valid, video decoder 300 may determine, for a pair of intersecting motion vectors comprising the first intersecting motion vector and a second intersecting motion vector, that a difference between the first horizontal component of the first source motion vector and a second horizontal component of a second source motion vector of the second intersecting motion vector is greater than a difference threshold. As part of determining that the first intersecting motion vector is valid, video decoder 300 may determine, for the pair of intersecting motion vectors, that a difference between the first vertical component of the first source motion vector and a second vertical component of the second source motion vector is greater than the difference threshold. As part of determining that the first intersecting motion vector is valid, video decoder 300 may determine, for the pair of

intersecting motion vectors, that a difference between the first horizontal component of the first destination motion vector and a second horizontal component of a second destination motion vector of the second intersecting motion vector is greater than the difference threshold. As part of determining that the first intersecting motion vector is valid, video decoder 300 may determine, for the pair of intersecting motion vectors, that a difference between the first vertical component of the first destination motion vector and a second vertical component of the second destination motion vector is greater than the difference threshold.

**[0298]** In some examples, as part of determining the intersecting motion vector, video decoder 300 may determine an intersecting motion vector list for the current block based on all of the corresponding intersecting motion vector lists. Video decoder 300 may determine a value of an index, the value of the index indicative of the intersecting motion vector in the intersecting motion vector list for the current block.

**[0299]** In some examples, a number of intersecting motion vectors in the intersecting motion vector list for the current block is less than a total number of intersecting motion vectors in the corresponding intersecting motion vector lists. For example, the number of intersecting motion vectors in the intersecting motion vector list for the current block is less than the total number of all the corresponding intersecting motion vectors in all the corresponding intersecting motion vector lists In some examples, as part of determining the intersecting motion vector list for the current block, video decoder 300 may sort intersecting motion vectors of each of the corresponding intersecting motion vector lists based on at least one of bilateral matching cost, occurrence count, or template matching cost.

**[0300]** In some examples, the current block has a top-left position of (X, Y), the source motion vector includes (srcItcMvHor, srcItcMvVer), the destination motion vector includes (dstItcMvHor, dstItcMvVer), the source block includes (srcRefBlk), and the destination block includes (dstRefBlk). In some examples, the source block has a top-left position of (srcX, srcY) and the destination block has a top-left position of (dstX, dstY), and wherein srcX is equal to X + srcItcMvHor, srcY is equal to Y + srcItcMvVer, dstX is equal to X + dstItcMvHor, dstY is equal to Y + dstItcMvVer. In some examples, 0. In some examples, a block size of the source block, a block size of the destination block, a position of the source block, and a position of the destination block are of a same precision as the source motion vector and the destination motion vector.

**[0301]** In some examples, as part of decoding the current block, video decoder 300 may decode the current block using subblock prediction, wherein each subblock is predicted using a respective intersecting motion vector. In some examples, as part of decoding the current block using subblock prediction, video decoder 300 may use an affine model.

**[0302]** Referring now to FIG. 18B, video encoder 200 may determine, for each of a plurality of subblocks of a current block of a current picture of video data, a corresponding intersecting motion vector list (810). Each corresponding intersecting motion vector list may include at least one intersecting motion vector. The at least one intersecting motion vector may include a source motion vector associated with a source block located in a first reference picture and a destination motion vector associated with a destination block located in a second reference picture. The source block may have a predictor that is a subset of the destination block. The first reference picture may be already encoded and be temporally prior to the current picture in display order and the second reference picture being already encoded and temporally after the current picture in display order. In some examples, the plurality of subblocks include a plurality of 4x4 subblocks.

**[0303]** Video encoder 200 may determine an intersecting motion vector for the current block based on the corresponding intersecting motion vector lists (812). For example, video encoder 200 may determine an intersecting motion vector from the intersecting motion vectors in the intersecting motion vector lists for the subblocks of the current block.

**[0304]** Video encoder 200 may encode the current block in accordance with the intersecting motion vector (814). For example, video encoder 200 may encode the current block such that video decoder 300 may use the intersecting motion vector to predict the current block and generate a prediction block. Video decoder 300 may then combine the prediction block with a residual block to decode the current block.

**[0305]** FIGS. 19A-19B are flowcharts illustrating example techniques for reference picture derivation according to one or more aspects of this disclosure. The techniques of FIG. 19A are described with respect to video decoder 300, but may be practiced by any device (e.g., video encoder 200) that is capable of practicing such techniques.

**[0306]** Referring to FIG. 19A, video decoder 300 may determine a current reference picture for a current block of a current picture of the video data based on a reference picture of a collocated block, a reference picture list of the current block, and a POC of a

collocated picture (900). The collocated block may be collocated with the current block and is in the collocated picture.

**[0307]** Video decoder 300 may decode the current block based on a motion vector associated with the current reference picture (902). For example, video decoder 300 may predict the current block using the current reference picture to generate a prediction block and may combine the prediction with a residual block to decode the current block.

**[0308]** In some examples, as part of determining the current reference picture, video decoder 300 may determine a collocated reference picture for the collocated block. In some examples, as part of determining the current reference picture, video decoder 300 may determine a reference picture from the reference picture list that is associated with a highest number of pictures that overlap pictures between the collocated reference picture and the collocated picture.

**[0309]** In some examples, as part of determining the current reference picture, video decoder 300 may determine first pictures that are between the collocated reference picture and the collocated picture in a POC. Video decoder 300 may determine, for each potential current reference picture, corresponding second pictures that are between the potential current reference picture and the current picture in the POC. Video decoder 300 may determine, for each potential current reference picture, a corresponding count of pictures that are both first pictures and corresponding second picture. Video decoder 300 determine the current reference picture based on the corresponding counts.

**[0310]** In some examples, as part of determining the current reference picture based on the corresponding counts, video decoder 300 may select, as the current reference picture, the potential current reference picture having a highest number of corresponding counts. In some examples, more than one potential current reference picture is associated with a highest number of corresponding counts. In some examples, as part of determining the current reference picture based on the corresponding counts, video decoder may select, from the more than one potential current reference picture associated with the highest number of counts, a reference picture having a smallest POC distance from the current picture.

**[0311]** In some examples, the motion vector is associated with a temporal motion vector predictor, a subblock temporal motion vector predictor, a spatial merge candidate, or an advanced motion vector prediction candidate. In some examples, video decoder 300 may determine that the current picture is a random-access picture, a low-delay B picture, or a low-delay P picture. In such examples, video decoder 300 may determine the current

reference picture based on the determination that the current picture is the random-access picture, the low-delay B picture, or the low-delay P picture. In some examples, video decoder 300 may determine that a current slice comprising the current block is a random-access slice, a low-delay B slice, or a low-delay P slice. In such examples, video decoder 300 may determine the current reference picture based on the determination that the current slice is the random-access slice, the low-delay B slice, or the low-delay P slice. Determining the current reference picture for the random-access picture (or slice), the low-delay B picture (or slice), or the low-delay P picture (or slice) according to the techniques described herein may facilitate improvements to coding performance.

**[0312]** Referring now to FIG. 19B, video encoder 200 may determine a current reference picture for a current block of a current picture of the video data based on a reference picture of a collocated block, a reference picture list of the current block, and a POC of a collocated picture (910). The collocated block may be collocated with the current block and is in the collocated picture.

**[0313]** Video encoder 200 may encode the current block in accordance with a motion vector associated with the current reference picture (912). For example, video encoder 200 may encode the current block such that video decoder 300 may predict the current block using the current reference picture to generate a prediction block and may combine the prediction with a residual block to decode the current block.

**[0314]** FIG. 20 is a block diagram illustrating an example video encoder 200 that may perform the techniques of this disclosure. FIG. 20 is provided for purposes of explanation and should not be considered limiting of the techniques as broadly exemplified and described in this disclosure. For purposes of explanation, this disclosure describes video encoder 200 according to the techniques of VVC and HEVC. However, the techniques of this disclosure may be performed by video encoding devices that are configured to other video coding standards and video coding formats, such as AV1 and successors to the AV1 video coding format.

**[0315]** In the example of FIG. 20, video encoder 200 includes video data memory 230, mode selection unit 202, residual generation unit 204, transform processing unit 206, quantization unit 208, inverse quantization unit 210, inverse transform processing unit 212, reconstruction unit 214, filter unit 216, decoded picture buffer (DPB) 218, and entropy encoding unit 220. Any or all of video data memory 230, mode selection unit 202, residual generation unit 204, transform processing unit 206, quantization unit 208, inverse quantization unit 210, inverse transform processing unit 212, reconstruction unit

214, filter unit 216, DPB 218, and entropy encoding unit 220 may be implemented in one or more processors or in processing circuitry. For instance, the units of video encoder 200 may be implemented as one or more circuits or logic elements as part of hardware circuitry, or as part of a processor, ASIC, or FPGA. Moreover, video encoder 200 may include additional or alternative processors or processing circuitry to perform these and other functions.

**[0316]** Video data memory 230 is an example of a memory system that may store video data to be encoded by the components of video encoder 200. Video encoder 200 may receive the video data stored in video data memory 230 from, for example, video source 104 (FIG. 1). DPB 218 is an example of a memory system that may act as a reference picture memory that stores reference video data for use in prediction of subsequent video data by video encoder 200. Video data memory 230 and DPB 218 may each be formed by any of a variety of one or more memory devices or memory units, such as dynamic random-access memory (DRAM), including synchronous DRAM (SDRAM), magnetoresistive RAM (MRAM), resistive RAM (RRAM), or other types of memory devices. Video data memory 230 and DPB 218 may be provided by the same memory device or separate memory devices. In various examples, video data memory 230 may be on-chip with other components of video encoder 200, as illustrated, or off-chip relative to those components.

**[0317]** In this disclosure, reference to video data memory 230 should not be interpreted as being limited to memory internal to video encoder 200, unless specifically described as such, or memory external to video encoder 200, unless specifically described as such. Rather, reference to video data memory 230 should be understood as reference memory that stores video data that video encoder 200 receives for encoding (e.g., video data for a current block that is to be encoded). Memory 106 of FIG. 1 may also provide temporary storage of outputs from the various units of video encoder 200.

**[0318]** The various units of FIG. 20 are illustrated to assist with understanding the operations performed by video encoder 200. The units may be implemented as fixed-function circuits, programmable circuits, or a combination thereof. Fixed-function circuits refer to circuits that provide particular functionality, and are preset on the operations that can be performed. Programmable circuits refer to circuits that can be programmed to perform various tasks, and provide flexible functionality in the operations that can be performed. For instance, programmable circuits may execute software or firmware that cause the programmable circuits to operate in the manner defined by

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

**[0319]** Video encoder 200 may include arithmetic logic units (ALUs), elementary function units (EFUs), digital circuits, analog circuits, and/or programmable cores, formed from programmable circuits. In examples where the operations of video encoder 200 are performed using software executed by the programmable circuits, memory 106 (FIG. 1) may store the instructions (e.g., object code) of the software that video encoder 200 receives and executes, or another memory within video encoder 200 (not shown) may store such instructions.

**[0320]** Video data memory 230 is configured to store received video data. Video encoder 200 may retrieve a picture of the video data from video data memory 230 and provide the video data to residual generation unit 204 and mode selection unit 202. Video data in video data memory 230 may be raw video data that is to be encoded.

**[0321]** Mode selection unit 202 includes a motion estimation unit 222, a motion compensation unit 224, and an intra-prediction unit 226. Mode selection unit 202 may include additional functional units to perform video prediction in accordance with other prediction modes. As examples, mode selection unit 202 may include a palette unit, an intra-block copy unit (which may be part of motion estimation unit 222 and/or motion compensation unit 224), an affine unit, a linear model (LM) unit, or the like.

**[0322]** Mode selection unit 202 generally coordinates multiple encoding passes to test combinations of encoding parameters and resulting rate-distortion values for such combinations. The encoding parameters may include partitioning of CTUs into CUs, prediction modes for the CUs, transform types for residual data of the CUs, quantization parameters for residual data of the CUs, and so on. Mode selection unit 202 may ultimately select the combination of encoding parameters having rate-distortion values that are better than the other tested combinations.

**[0323]** Video encoder 200 may partition a picture retrieved from video data memory 230 into a series of CTUs, and encapsulate one or more CTUs within a slice. Mode selection unit 202 may partition a CTU of the picture in accordance with a tree structure, such as the MTT structure, QTBT structure. superblock structure, or the quad-tree structure described above. As described above, video encoder 200 may form one or more CUs

from partitioning a CTU according to the tree structure. Such a CU may also be referred to generally as a "video block" or "block."

**[0324]** In general, mode selection unit 202 also controls the components thereof (e.g., motion estimation unit 222, motion compensation unit 224, and intra-prediction unit 226) to generate a prediction block for a current block (e.g., a current CU, or in HEVC, the overlapping portion of a PU and a TU). For inter-prediction of a current block, motion estimation unit 222 may perform a motion search to identify one or more closely matching reference blocks in one or more reference pictures (e.g., one or more previously coded pictures stored in DPB 218). In particular, motion estimation unit 222 may calculate a value representative of how similar a potential reference block is to the current block, e.g., according to sum of absolute difference (SAD), sum of squared differences (SSD), mean absolute difference (MAD), mean squared differences (MSD), or the like. Motion estimation unit 222 may generally perform these calculations using sample-by-sample differences between the current block and the reference block being considered. Motion estimation unit 222 may identify a reference block having a lowest value resulting from these calculations, indicating a reference block that most closely matches the current block.

**[0325]** Motion estimation unit 222 may form one or more motion vectors (MVs) that defines the positions of the reference blocks in the reference pictures relative to the position of the current block in a current picture. Motion estimation unit 222 may then provide the motion vectors to motion compensation unit 224. For example, for uni-directional inter-prediction, motion estimation unit 222 may provide a single motion vector, whereas for bi-directional inter-prediction, motion estimation unit 222 may provide two motion vectors. Motion compensation unit 224 may then generate a prediction block using the motion vectors. For example, motion compensation unit 224 may retrieve data of the reference block using the motion vector. As another example, if the motion vector has fractional sample precision, motion compensation unit 224 may interpolate values for the prediction block according to one or more interpolation filters. Moreover, for bi-directional inter-prediction, motion compensation unit 224 may retrieve data for two reference blocks identified by respective motion vectors and combine the retrieved data, e.g., through sample-by-sample averaging or weighted averaging.

**[0326]** When operating according to the AV1 video coding format, motion estimation unit 222 and motion compensation unit 224 may be configured to encode coding blocks of video data (e.g., both luma and chroma coding blocks) using translational motion

compensation, affine motion compensation, overlapped block motion compensation (OBMC), and/or compound inter-intra prediction.

**[0327]** As another example, for intra-prediction, or intra-prediction coding, intra-prediction unit 226 may generate the prediction block from samples neighboring the current block. For example, for directional modes, intra-prediction unit 226 may generally mathematically combine values of neighboring samples and populate these calculated values in the defined direction across the current block to produce the prediction block. As another example, for DC mode, intra-prediction unit 226 may calculate an average of the neighboring samples to the current block and generate the prediction block to include this resulting average for each sample of the prediction block.

**[0328]** When operating according to the AV1 video coding format, intra-prediction unit 226 may be configured to encode coding blocks of video data (e.g., both luma and chroma coding blocks) using directional intra prediction, non-directional intra prediction, recursive filter intra prediction, chroma-from-luma (CFL) prediction, intra block copy (IBC), and/or color palette mode. Mode selection unit 202 may include additional functional units to perform video prediction in accordance with other prediction modes.

**[0329]** Mode selection unit 202 provides the prediction block to residual generation unit 204. Residual generation unit 204 receives a raw, unencoded version of the current block from video data memory 230 and the prediction block from mode selection unit 202. Residual generation unit 204 calculates sample-by-sample differences between the current block and the prediction block. The resulting sample-by-sample differences define a residual block for the current block. In some examples, residual generation unit 204 may also determine differences between sample values in the residual block to generate a residual block using residual differential pulse code modulation (RDPCM). In some examples, residual generation unit 204 may be formed using one or more subtractor circuits that perform binary subtraction.

**[0330]** In examples where mode selection unit 202 partitions CUs into PUs, each PU may be associated with a luma prediction unit and corresponding chroma prediction units. Video encoder 200 and video decoder 300 may support PUs having various sizes. As indicated above, the size of a CU may refer to the size of the luma coding block of the CU and the size of a PU may refer to the size of a luma prediction unit of the PU. Assuming that the size of a particular CU is 2Nx2N, video encoder 200 may support PU sizes of 2Nx2N or NxN for intra prediction, and symmetric PU sizes of 2Nx2N, 2NxN, Nx2N, NxN, or similar for inter prediction. Video encoder 200 and video decoder 300

may also support asymmetric partitioning for PU sizes of 2NxnU, 2NxnD, nLx2N, and nRx2N for inter prediction.

**[0331]** In examples where mode selection unit 202 does not further partition a CU into PUs, each CU may be associated with a luma coding block and corresponding chroma coding blocks. As above, the size of a CU may refer to the size of the luma coding block of the CU. The video encoder 200 and video decoder 300 may support CU sizes of 2Nx2N, 2NxN, or Nx2N.

**[0332]** For other video coding techniques such as an intra-block copy mode coding, an affine-mode coding, and linear model (LM) mode coding, as some examples, mode selection unit 202, via respective units associated with the coding techniques, generates a prediction block for the current block being encoded. In some examples, such as palette mode coding, mode selection unit 202 may not generate a prediction block, and instead generate syntax elements that indicate the manner in which to reconstruct the block based on a selected palette. In such modes, mode selection unit 202 may provide these syntax elements to entropy encoding unit 220 to be encoded.

**[0333]** As described above, residual generation unit 204 receives the video data for the current block and the corresponding prediction block. Residual generation unit 204 then generates a residual block for the current block. To generate the residual block, residual generation unit 204 calculates sample-by-sample differences between the prediction block and the current block.

**[0334]** Transform processing unit 206 applies one or more transforms to the residual block to generate a block of transform coefficients (referred to herein as a "transform coefficient block"). Transform processing unit 206 may apply various transforms to a residual block to form the transform coefficient block. For example, transform processing unit 206 may apply a discrete cosine transform (DCT), a directional transform, a Karhunen-Loeve transform (KLT), or a conceptually similar transform to a residual block. In some examples, transform processing unit 206 may perform multiple transforms to a residual block, e.g., a primary transform and a secondary transform, such as a rotational transform. In some examples, transform processing unit 206 does not apply transforms to a residual block.

**[0335]** When operating according to AV1, transform processing unit 206 may apply one or more transforms to the residual block to generate a block of transform coefficients (referred to herein as a "transform coefficient block"). Transform processing unit 206 may apply various transforms to a residual block to form the transform coefficient block.

For example, transform processing unit 206 may apply a horizontal/vertical transform combination that may include a discrete cosine transform (DCT), an asymmetric discrete sine transform (ADST), a flipped ADST (e.g., an ADST in reverse order), and an identity transform (IDTX). When using an identity transform, the transform is skipped in one of the vertical or horizontal directions. In some examples, transform processing may be skipped.

**[0336]** Quantization unit 208 may quantize the transform coefficients in a transform coefficient block, to produce a quantized transform coefficient block. Quantization unit 208 may quantize transform coefficients of a transform coefficient block according to a quantization parameter (QP) value associated with the current block. Video encoder 200 (e.g., via mode selection unit 202) may adjust the degree of quantization applied to the transform coefficient blocks associated with the current block by adjusting the QP value associated with the CU. Quantization may introduce loss of information, and thus, quantized transform coefficients may have lower precision than the original transform coefficients produced by transform processing unit 206.

**[0337]** Inverse quantization unit 210 and inverse transform processing unit 212 may apply inverse quantization and inverse transforms to a quantized transform coefficient block, respectively, to reconstruct a residual block from the transform coefficient block. Reconstruction unit 214 may produce a reconstructed block corresponding to the current block (albeit potentially with some degree of distortion) based on the reconstructed residual block and a prediction block generated by mode selection unit 202. For example, reconstruction unit 214 may add samples of the reconstructed residual block to corresponding samples from the prediction block generated by mode selection unit 202 to produce the reconstructed block.

**[0338]** Filter unit 216 may perform one or more filter operations on reconstructed blocks. For example, filter unit 216 may perform deblocking operations to reduce blockiness artifacts along edges of CUs. Operations of filter unit 216 may be skipped, in some examples.

**[0339]** When operating according to AV1, filter unit 216 may perform one or more filter operations on reconstructed blocks. For example, filter unit 216 may perform deblocking operations to reduce blockiness artifacts along edges of CUs. In other examples, filter unit 216 may apply a constrained directional enhancement filter (CDEF), which may be applied after deblocking, and may include the application of non-separable, non-linear, low-pass directional filters based on estimated edge directions. Filter unit 216 may also

include a loop restoration filter, which is applied after CDEF, and may include a separable symmetric normalized Wiener filter or a dual self-guided filter.

**[0340]** Video encoder 200 stores reconstructed blocks in DPB 218. For instance, in examples where operations of filter unit 216 are not performed, reconstruction unit 214 may store reconstructed blocks to DPB 218. In examples where operations of filter unit 216 are performed, filter unit 216 may store the filtered reconstructed blocks to DPB 218. Motion estimation unit 222 and motion compensation unit 224 may retrieve a reference picture from DPB 218, formed from the reconstructed (and potentially filtered) blocks, to inter-predict blocks of subsequently encoded pictures. In addition, intra-prediction unit 226 may use reconstructed blocks in DPB 218 of a current picture to intra-predict other blocks in the current picture.

**[0341]** In general, entropy encoding unit 220 may entropy encode syntax elements received from other functional components of video encoder 200. For example, entropy encoding unit 220 may entropy encode quantized transform coefficient blocks from quantization unit 208. As another example, entropy encoding unit 220 may entropy encode prediction syntax elements (e.g., motion information for inter-prediction or intra-mode information for intra-prediction) from mode selection unit 202. Entropy encoding unit 220 may perform one or more entropy encoding operations on the syntax elements, which are another example of video data, to generate entropy-encoded data. For example, entropy encoding unit 220 may perform a context-adaptive variable length coding (CAVLC) operation, a CABAC operation, a variable-to-variable (V2V) length coding operation, a syntax-based context-adaptive binary arithmetic coding (SBAC) operation, a Probability Interval Partitioning Entropy (PIPE) coding operation, an Exponential-Golomb encoding operation, or another type of entropy encoding operation on the data. In some examples, entropy encoding unit 220 may operate in bypass mode where syntax elements are not entropy encoded.

**[0342]** Video encoder 200 may output a bitstream that includes the entropy encoded syntax elements needed to reconstruct blocks of a slice or picture. In particular, entropy encoding unit 220 may output the bitstream.

**[0343]** In accordance with AV1, entropy encoding unit 220 may be configured as a symbol-to-symbol adaptive multi-symbol arithmetic coder. A syntax element in AV1 includes an alphabet of N elements, and a context (e.g., probability model) includes a set of N probabilities. Entropy encoding unit 220 may store the probabilities as n-bit (e.g., 15-bit) cumulative distribution functions (CDFs). Entropy encoding unit 220 may

perform recursive scaling, with an update factor based on the alphabet size, to update the contexts.

**[0344]** The operations described above are described with respect to a block. Such description should be understood as being operations for a luma coding block and/or chroma coding blocks. As described above, in some examples, the luma coding block and chroma coding blocks are luma and chroma components of a CU. In some examples, the luma coding block and the chroma coding blocks are luma and chroma components of a PU.

**[0345]** In some examples, operations performed with respect to a luma coding block need not be repeated for the chroma coding blocks. As one example, operations to identify a motion vector (MV) and reference picture for a luma coding block need not be repeated for identifying a MV and reference picture for the chroma blocks. Rather, the MV for the luma coding block may be scaled to determine the MV for the chroma blocks, and the reference picture may be the same. As another example, the intra-prediction process may be the same for the luma coding block and the chroma coding blocks.

**[0346]** Video encoder 200 represents an example of a device configured to encode video data including one or more memories for storing the video data; and one or more processors operatively coupled to the one or more memories, the one or more processors configured to: determine, for each of a plurality of subblocks of a current block of a current picture of the video data, a corresponding intersecting motion vector list, each corresponding intersecting motion vector list comprising at least one intersecting motion vector, the at least one intersecting motion vector comprising a source motion vector associated with a source block located in a first reference picture and a destination motion vector associated with a destination block located in a second reference picture, and wherein the source block has a predictor that is a subset of the destination block, the first reference picture being already decoded and temporally prior to the current picture and the second reference picture being already decoded and temporally after the current picture; determine an intersecting motion vector for the current block based on the corresponding intersecting motion vector lists; and encode the current block in accordance with the intersecting motion vector.

**[0347]** Video encoder 200 also represents an example of a device configured to encode video data including one or more memories for storing the video data; and one or more processors operatively coupled to the one or more memories, the one or more processors configured to: determine a current reference picture for a current block of a current picture

of the video data based on a reference picture of a collocated block, a reference picture list of the current block, and a picture order count (POC) of a collocated picture, wherein the collocated block is collocated with the current block and is in the collocated picture; and encode the current block in accordance with a motion vector associated with the current reference picture.

**[0348]** Video encoder 200 also represents an example of a device configured to encode video data including one or more memories configured to store video data, and one or more processing units implemented in circuitry and configured to determine at least one intersecting motion vector for a current block of a current picture of the video data, the intersecting motion vector comprising a source motion vector associated with a first reference picture and a destination motion vector associated with a second reference picture, the first reference picture being temporally prior to the current picture and the second reference picture being temporally after the current picture; and encode the current block based on the at least one intersecting motion vector.

**[0349]** Video encoder 200 also represents an example of a device configured to encode video data including a memory configured to store video data, and one or more processing units implemented in circuitry and configured to determine a temporal motion vector predictor for a current block of a current picture of the video data based on a collocated block of a collocated picture offset by a motion vector of a spatial neighbor block, neighboring the current block in the current picture; and encode the current block based on the temporal motion vector predictor.

**[0350]** Video encoder 200 also represents an example of a device configured to encode video data including a memory configured to store video data, and one or more processing units implemented in circuitry and configured to determine a reference picture for a current block of the video data; and encode the current block of the video data based on a motion vector associated with the reference picture.

**[0351]** Video encoder 200 also represents an example of a device configured to encode video data including a memory configured to store video data, and one or more processing units implemented in circuitry and configured to determine one or more new motion vector predictors from at least one existing motion vector predictor in a motion vector predictor list for a current block of the video data; and encode the video data based on the one or more new motion vector predictors.

**[0352]** FIG. 21 is a block diagram illustrating an example video decoder 300 that may perform the techniques of this disclosure. FIG. 21 is provided for purposes of explanation

and is not limiting on the techniques as broadly exemplified and described in this disclosure. For purposes of explanation, this disclosure describes video decoder 300 according to the techniques of VVC and HEVC. However, the techniques of this disclosure may be performed by video coding devices that are configured to other video coding standards.

**[0353]** In the example of FIG. 21, video decoder 300 includes coded picture buffer (CPB) memory 320, entropy decoding unit 302, prediction processing unit 304, inverse quantization unit 306, inverse transform processing unit 308, reconstruction unit 310, filter unit 312, and DPB 314. Any or all of CPB memory 320, entropy decoding unit 302, prediction processing unit 304, inverse quantization unit 306, inverse transform processing unit 308, reconstruction unit 310, filter unit 312, and DPB 314 may be implemented in one or more processors or in processing circuitry. For instance, the units of video decoder 300 may be implemented as one or more circuits or logic elements as part of hardware circuitry, or as part of a processor, ASIC, or FPGA. Moreover, video decoder 300 may include additional or alternative processors or processing circuitry to perform these and other functions.

**[0354]** Prediction processing unit 304 includes motion compensation unit 316 and intra-prediction unit 318. Prediction processing unit 304 may include additional units to perform prediction in accordance with other prediction modes. As examples, prediction processing unit 304 may include a palette unit, an intra-block copy unit (which may form part of motion compensation unit 316), an affine unit, a linear model (LM) unit, or the like. In other examples, video decoder 300 may include more, fewer, or different functional components.

**[0355]** When operating according to AV1, motion compensation unit 316 may be configured to decode coding blocks of video data (e.g., both luma and chroma coding blocks) using translational motion compensation, affine motion compensation, OBMC, and/or compound inter-intra prediction, as described above. Intra-prediction unit 318 may be configured to decode coding blocks of video data (e.g., both luma and chroma coding blocks) using directional intra prediction, non-directional intra prediction, recursive filter intra prediction, CFL, IBC, and/or color palette mode, as described above.

**[0356]** CPB memory 320 is an example of a memory system that may store video data, such as an encoded video bitstream, to be decoded by the components of video decoder 300. The video data stored in CPB memory 320 may be obtained, for example, from computer-readable medium 110 (FIG. 1). CPB memory 320 may include a CPB that

stores encoded video data (e.g., syntax elements) from an encoded video bitstream. Also, CPB memory 320 may store video data other than syntax elements of a coded picture, such as temporary data representing outputs from the various units of video decoder 300. DPB 314 is an example of a memory system that generally stores decoded pictures, which video decoder 300 may output and/or use as reference video data when decoding subsequent data or pictures of the encoded video bitstream. CPB memory 320 and DPB 314 may each be formed by any of a variety of memory devices or memory units, such as DRAM, including SDRAM, MRAM, RRAM, or other types of memory devices. CPB memory 320 and DPB 314 may be provided by the same memory device or separate memory devices. In various examples, CPB memory 320 may be on-chip with other components of video decoder 300, or off-chip relative to those components.

**[0357]** Additionally or alternatively, in some examples, video decoder 300 may retrieve coded video data from memory 120 (FIG. 1). That is, memory 120 may store data as discussed above with CPB memory 320. Likewise, memory 120 may store instructions to be executed by video decoder 300, when some or all of the functionality of video decoder 300 is implemented in software to be executed by processing circuitry of video decoder 300.

**[0358]** The various units shown in FIG. 21 are illustrated to assist with understanding the operations performed by video decoder 300. The units may be implemented as fixed-function circuits, programmable circuits, or a combination thereof. Similar to FIG. 20, fixed-function circuits refer to circuits that provide particular functionality, and are preset on the operations that can be performed. Programmable circuits refer to circuits that can be programmed to perform various tasks, and provide flexible functionality in the operations that can be performed. For instance, programmable circuits may execute software or firmware that cause the programmable circuits to operate in the manner defined by instructions of the software or firmware. Fixed-function circuits may execute software instructions (e.g., to receive parameters or output parameters), but the types of operations that the fixed-function circuits perform are generally immutable. In some examples, one or more of the units may be distinct circuit blocks (fixed-function or programmable), and in some examples, one or more of the units may be integrated circuits.

**[0359]** Video decoder 300 may include ALUs, EFUs, digital circuits, analog circuits, and/or programmable cores formed from programmable circuits. In examples where the operations of video decoder 300 are performed by software executing on the

programmable circuits, on-chip or off-chip memory may store instructions (e.g., object code) of the software that video decoder 300 receives and executes.

**[0360]** Entropy decoding unit 302 may receive encoded video data from the CPB and entropy decode the video data to reproduce syntax elements. Prediction processing unit 304, inverse quantization unit 306, inverse transform processing unit 308, reconstruction unit 310, and filter unit 312 may generate decoded video data based on the syntax elements extracted from the bitstream.

**[0361]** In general, video decoder 300 reconstructs a picture on a block-by-block basis. Video decoder 300 may perform a reconstruction operation on each block individually (where the block currently being reconstructed, i.e., decoded, may be referred to as a "current block").

**[0362]** Entropy decoding unit 302 may entropy decode syntax elements defining quantized transform coefficients of a quantized transform coefficient block, as well as transform information, such as a quantization parameter (QP) and/or transform mode indication(s). Inverse quantization unit 306 may use the QP associated with the quantized transform coefficient block to determine a degree of quantization and, likewise, a degree of inverse quantization for inverse quantization unit 306 to apply. Inverse quantization unit 306 may, for example, perform a bitwise left-shift operation to inverse quantize the quantized transform coefficients. Inverse quantization unit 306 may thereby form a transform coefficient block including transform coefficients.

**[0363]** After inverse quantization unit 306 forms the transform coefficient block, inverse transform processing unit 308 may apply one or more inverse transforms to the transform coefficient block to generate a residual block associated with the current block. For example, inverse transform processing unit 308 may apply an inverse DCT, an inverse integer transform, an inverse Karhunen-Loeve transform (KLT), an inverse rotational transform, an inverse directional transform, or another inverse transform to the transform coefficient block.

**[0364]** Furthermore, prediction processing unit 304 generates a prediction block according to prediction information syntax elements that were entropy decoded by entropy decoding unit 302. For example, if the prediction information syntax elements indicate that the current block is inter-predicted, motion compensation unit 316 may generate the prediction block. In this case, the prediction information syntax elements may indicate a reference picture in DPB 314 from which to retrieve a reference block, as well as a motion vector identifying a location of the reference block in the reference

picture relative to the location of the current block in the current picture. Motion compensation unit 316 may generally perform the inter-prediction process in a manner that is substantially similar to that described with respect to motion compensation unit 224 (FIG. 20).

**[0365]** As another example, if the prediction information syntax elements indicate that the current block is intra-predicted, intra-prediction unit 318 may generate the prediction block according to an intra-prediction mode indicated by the prediction information syntax elements. Again, intra-prediction unit 318 may generally perform the intra-prediction process in a manner that is substantially similar to that described with respect to intra-prediction unit 226 (FIG. 20). Intra-prediction unit 318 may retrieve data of neighboring samples to the current block from DPB 314.

**[0366]** Reconstruction unit 310 may reconstruct the current block using the prediction block and the residual block. For example, reconstruction unit 310 may add samples of the residual block to corresponding samples of the prediction block to reconstruct the current block.

**[0367]** Filter unit 312 may perform one or more filter operations on reconstructed blocks. For example, filter unit 312 may perform deblocking operations to reduce blockiness artifacts along edges of the reconstructed blocks. Operations of filter unit 312 are not necessarily performed in all examples.

**[0368]** Video decoder 300 may store the reconstructed blocks in DPB 314. For instance, in examples where operations of filter unit 312 are not performed, reconstruction unit 310 may store reconstructed blocks to DPB 314. In examples where operations of filter unit 312 are performed, filter unit 312 may store the filtered reconstructed blocks to DPB 314. As discussed above, DPB 314 may provide reference information, such as samples of a current picture for intra-prediction and previously decoded pictures for subsequent motion compensation, to prediction processing unit 304. Moreover, video decoder 300 may output decoded pictures (e.g., decoded video) from DPB 314 for subsequent presentation on a display device, such as display device 118 of FIG. 1.

**[0369]** In this manner, video decoder 300 represents an example of a video decoding device including one or more memories for storing the video data; and one or more processors operatively coupled to the one or more memories, the one or more processors configured to: determine, for each of a plurality of subblocks of a current block of a current picture of the video data, a corresponding intersecting motion vector list, each corresponding intersecting motion vector list comprising at least one intersecting motion

vector, the at least one intersecting motion vector comprising a source motion vector associated with a source block located in a first reference picture and a destination motion vector associated with a destination block located in a second reference picture, and wherein the source block has a predictor that is a subset of the destination block, the first reference picture being already decoded and temporally prior to the current picture and the second reference picture being already decoded and temporally after the current picture; determine an intersecting motion vector for the current block based on the corresponding intersecting motion vector lists; and decode the current block based on the intersecting motion vector.

**[0370]** Video decoder 300 also represents an example of a device configured to decode video data including one or more memories for storing the video data; and one or more processors operatively coupled to the one or more memories, the one or more processors configured to: determine a current reference picture for a current block of a current picture of the video data based on a reference picture of a collocated block, a reference picture list of the current block, and a picture order count (POC) of a collocated picture, wherein the collocated block is collocated with the current block and is in the collocated picture; and decode the current block based on a motion vector associated with the current reference picture.

**[0371]** Video decoder 300 also represents an example of a video decoding device including one or more memories configured to store video data, and one or more processing units implemented in circuitry and configured to determine at least one intersecting motion vector for a current block of a current picture of the video data, the intersecting motion vector comprising a source motion vector associated with a first reference picture and a destination motion vector associated with a second reference picture, the first reference picture being temporally prior to the current picture and the second reference picture being temporally after the current picture; and decode the current block based on the at least one intersecting motion vector.

**[0372]** Video decoder 300 also represents an example of a device configured to decode video data including one or more memories configured to store video data, and one or more processing units implemented in circuitry and configured to determine a temporal motion vector predictor for a current block of a current picture of the video data based on a collocated block of a collocated picture offset by a motion vector of a spatial neighbor block, neighboring the current block in the current picture; and decode the current block based on the temporal motion vector predictor.

**[0373]** Video decoder 300 also represents an example of a device configured to decode video data including one or more memories configured to store video data, and one or more processing units implemented in circuitry and configured to determine a reference picture for a current block of the video data; and decode the current block of the video data based on a motion vector associated with the reference picture.

**[0374]** Video decoder 300 also represents an example of a device configured to decode video data including one or more memories configured to store video data, and one or more processing units implemented in circuitry and configured to determine one or more new motion vector predictors from at least one existing motion vector predictor in a motion vector predictor list for a current block of the video data; and decode the video data based on the one or more new motion vector predictors.

**[0375]** FIG. 22 is a flowchart illustrating an example method for encoding a current block in accordance with the techniques of this disclosure. The current block may be or include a current CU. Although described with respect to video encoder 200 (FIGS. 1 and 20), it should be understood that other devices may be configured to perform a method similar to that of FIG. 22.

**[0376]** In this example, video encoder 200 initially predicts the current block (1000). For example, video encoder 200 may form a prediction block for the current block. Video encoder 200 may then calculate a residual block for the current block (1002). To calculate the residual block, video encoder 200 may calculate a difference between the original, unencoded block and the prediction block for the current block. Video encoder 200 may then transform the residual block and quantize transform coefficients of the residual block (1004). Next, video encoder 200 may scan the quantized transform coefficients of the residual block (1006). During the scan, or following the scan, video encoder 200 may entropy encode the transform coefficients (1008). For example, video encoder 200 may encode the transform coefficients using CAVLC or CABAC. Video encoder 200 may then output the entropy encoded data of the block (1010).

**[0377]** FIG. 23 is a flowchart illustrating an example method for decoding a current block of video data in accordance with the techniques of this disclosure. The current block may be or include a current CU. Although described with respect to video decoder 300 (FIGS. 1 and 21), it should be understood that other devices may be configured to perform a method similar to that of FIG. 23.

**[0378]** Video decoder 300 may receive entropy encoded data for the current block, such as entropy encoded prediction information and entropy encoded data for transform

coefficients of a residual block corresponding to the current block (1100). Video decoder 300 may entropy decode the entropy encoded data to determine prediction information for the current block and to reproduce transform coefficients of the residual block (1102). Video decoder 300 may predict the current block (1104), e.g., using an intra- or inter-prediction mode as indicated by the prediction information for the current block, to calculate a prediction block for the current block. Video decoder 300 may then inverse scan the reproduced transform coefficients (1106), to create a block of quantized transform coefficients. Video decoder 300 may then inverse quantize the transform coefficients and apply an inverse transform to the transform coefficients to produce a residual block (1108). Video decoder 300 may ultimately decode the current block by combining the prediction block and the residual block (1110).

**[0379]** The following numbered clauses illustrate one or more aspects of the devices and techniques described in this disclosure.

**[0380]** Aspect 1A. A method of coding video data, the method comprising: determining at least one intersecting motion vector for a current block of a current picture of the video data, the intersecting motion vector comprising a source motion vector associated with a first reference picture and a destination motion vector associated with a second reference picture, the first reference picture being temporally prior to the current picture and the second reference picture being temporally after the current picture; and coding the current block based on the at least one intersecting motion vector.

**[0381]** Aspect 2A. The method of aspect 1A, wherein the at least one intersecting motion vector comprises a plurality of intersecting motion vectors.

**[0382]** Aspect 3A. The method of aspect 2A, wherein coding the current block comprises determining that a first intersecting motion vector of the plurality of intersecting motion vectors is valid.

**[0383]** Aspect 4A. The method of aspect 3A, wherein coding the current block further comprises storing, based on determining that the first intersecting motion vector is valid, the first intersecting motion vector in memory.

**[0384]** Aspect 5A. The method of any of aspects 2A-4A, wherein coding the current block further comprises sorting the plurality of intersecting motion vectors.

**[0385]** Aspect 6A. The method of aspect 5A, wherein sorting the plurality of intersecting motion vectors is based at least in part on bilateral matching cost.

**[0386]** Aspect 7A. The method of aspect 5A or aspect 6A, wherein sorting the plurality of intersecting motion vectors is based at least in part on occurrence counts.

**[0387]** Aspect 8A. The method of any of aspects 1A-7A, further comprising determining that the current block is an inter prediction block, and wherein determining the at least one intersecting motion vector for the current block is based on the current block being the inter prediction block.

**[0388]** Aspect 9A. The method of any of aspects 2A-8A, wherein the coding the current block comprises coding the current block using subblock prediction, wherein each subblock is predicted using a respective one of the plurality of intersecting motion vectors.

**[0389]** Aspect 10A. The method of aspect 9A, wherein coding the current block using subblock prediction comprises using an affine model.

**[0390]** Aspect 11A. A method of coding video data, the method comprising: determining a temporal motion vector predictor for a current block of a current picture of the video data based on a collocated block of a collocated picture offset by a motion vector of a spatial neighbor block, neighboring the current block in the current picture; and coding the current block based on the temporal motion vector predictor.

**[0391]** Aspect 12A. The method of aspect 11A, wherein determining the temporal motion vector predictor comprises determining the offset.

**[0392]** Aspect 13A. A method of coding video data, the method comprising: determining a reference picture for a current block of the video data; and coding the current block of the video data based on a motion vector associated with the reference picture.

**[0393]** Aspect 14A. The method of aspect 13A, wherein determining the reference picture comprises determining a reference picture for a collocated block of the video data collocated with the current block.

**[0394]** Aspect 15A. The method of any of aspects 13A-14A, wherein the motion vector comprises a TMVP, SbTMVP, spatial merge candidate, or AMVP candidate.

**[0395]** Aspect 16A. A method of coding video data, the method comprising: determining one or more new motion vector predictors from at least one existing motion vector predictor in a motion vector predictor list for a current block of the video data; and coding the video data based on the one or more new motion vector predictors.

**[0396]** Aspect 17A. The method of any of aspects 1A–16A, wherein coding comprises decoding.

**[0397]** Aspect 18A. The method of any of aspects 1A–17A, wherein coding comprises encoding.

**[0398]** Aspect 19A. A device for coding video data, the device comprising one or more means for performing the method of any of aspects 1A–18A.

**[0399]** Aspect 20A. The device of aspect 19A, wherein the one or more means comprise one or more processors implemented in circuitry.

**[0400]** Aspect 21A. The device of any of aspects 19A-20A, further comprising one or more memories to store the video data.

**[0401]** Aspect 22A. The device of any of aspects 19A-21A, further comprising a display configured to display decoded video data.

**[0402]** Aspect 23A. The device of any of aspects 19A-22A, wherein the device comprises one or more of a camera, a computer, a mobile device, a broadcast receiver device, or a set-top box.

**[0403]** Aspect 24A. The device of any of aspects 19A-23A, wherein the device comprises a video decoder.

**[0404]** Aspect 25A. The device of any of aspects 19A-24A, wherein the device comprises a video encoder.

**[0405]** Aspect 26A. A computer-readable storage medium having stored thereon instructions that, when executed, cause one or more processors to perform the method of any of aspects 1A–18A.

**[0406]** Aspect 1B. A method of decoding video data, the method comprising: determining, for each of a plurality of subblocks of a current block of a current picture of the video data, a corresponding intersecting motion vector list, each corresponding intersecting motion vector list comprising at least one intersecting motion vector, the at least one intersecting motion vector comprising a source motion vector associated with a source block located in a first reference picture and a destination motion vector associated with a destination block located in a second reference picture, and wherein the source block has a predictor that is a subset of the destination block, the first reference picture being already decoded and temporally prior to the current picture and the second reference picture being already decoded and temporally after the current picture; determining an intersecting motion vector for the current block based on the corresponding intersecting motion vector lists; and decoding the current block based on the intersecting motion vector.

**[0407]** Aspect 2B. The method of aspect 1B, further comprising storing each corresponding intersecting motion vector list.

**[0408]** Aspect 3B. The method of aspect 2B, wherein storing each corresponding intersecting motion vector list comprises storing, for a first corresponding intersecting motion vector list, only a first number of intersecting motion vectors appearing first in a first corresponding intersecting motion vector list, the first number of intersecting motion vectors being less than a number of intersecting motion vectors in the first corresponding intersecting motion vector list.

**[0409]** Aspect 4B. The method of any of aspects 1B-3B, wherein determining the corresponding intersecting motion vector list comprises determining which intersecting motion vectors are valid, wherein the corresponding intersecting motion vector list comprises intersecting motion vectors determined to be valid and does not comprise intersecting motion vectors determined to not be valid.

**[0410]** Aspect 5B. The method of aspect 4B, wherein determining the corresponding intersecting motion vector list further comprises sorting valid intersecting motion vectors of a corresponding subblock based on bilateral matching cost so as to place an intersecting motion vector having a lower bilateral matching cost in a position closer to a front of the corresponding intersecting motion vector list than an intersecting motion vector having a higher bilateral matching cost.

**[0411]** Aspect 6B. The method of any of aspects 4B-5B, wherein determining the corresponding intersecting motion vector list further comprises sorting valid intersecting motion vectors of a corresponding subblock based on an occurrences count of each intersecting motion vector in the corresponding intersecting motion vector list so as to place an intersecting motion vector having a higher occurrence count cost in a position closer to a front of the corresponding intersecting motion vector list than the intersecting motion vector having a lower occurrence count.

**[0412]** Aspect 7B. The method of any of aspects 4B-6B, wherein determining which intersecting motion vectors are valid comprises determining that a first intersecting motion vector is valid, wherein determining that the first intersecting motion vector is valid comprises at least one of: determining that the first intersecting motion vector comprises both a first source motion vector and a first destination motion vector in an extended picture boundary; determining that a first absolute value of each of a first horizontal component of the first source motion vector and a first horizontal component of the first destination motion vector is greater than a horizontal threshold; determining that a second absolute value of each of a first vertical component of the first source motion vector and a first vertical component of the first destination motion vector is greater than

a vertical threshold; determining, for a pair of intersecting motion vectors comprising the first intersecting motion vector and a second intersecting motion vector, that a difference between the first horizontal component of the first source motion vector and a second horizontal component of a second source motion vector of the second intersecting motion vector is greater than a difference threshold; determining, for the pair of intersecting motion vectors, that a difference between the first vertical component of the first source motion vector and a second vertical component of the second source motion vector is greater than the difference threshold; determining, for the pair of intersecting motion vectors, that a difference between the first horizontal component of the first destination motion vector and a second horizontal component of a second destination motion vector of the second intersecting motion vector is greater than the difference threshold; or determining, for the pair of intersecting motion vectors, that a difference between the first vertical component of the first destination motion vector and a second vertical component of the second destination motion vector is greater than the difference threshold;

**[0413]** Aspect 8B. The method of any of aspects 1B-7B, wherein determining the intersecting motion vector comprises: determining an intersecting motion vector list for the current block based on all of the corresponding intersecting motion vector lists; and determining a value of an index, the value of the index indicative of the intersecting motion vector in the intersecting motion vector list for the current block.

**[0414]** Aspect 9B. The method of aspect 8B, wherein a number of intersecting motion vectors in the intersecting motion vector list for the current block is less than a total number of intersecting motion vectors in the corresponding intersecting motion vector lists.

**[0415]** Aspect 10B. The method of any of aspects 8B-9B, wherein determining the intersecting motion vector list for the current block comprises sorting intersecting motion vectors of each of the corresponding intersecting motion vector lists based on at least one of bilateral matching cost, occurrence count, or template matching cost.

**[0416]** Aspect 11B. The method of any of aspects 1B-10B, wherein the current block has a top-left position of (X, Y), the source motion vector comprises (srcItcMvHor, srcItcMvVer), the destination motion vector comprises (dstItcMvHor, dstItcMvVer), the source block comprises (srcRefBlk), and the destination block comprises (dstRefBlk), wherein the source block has a top-left position of (srcX, srcY) and the destination block has a top-left position of (dstX, dstY), and wherein srcX is equal to X + srcItcMvHor,

srcY is equal to Y + srcItcMvVer, dstX is equal to X + dstItcMvHor, dstY is equal to Y + dstItcMvVer.

**[0417]** Aspect 12B. The method of any of aspects 1B-11B, wherein a block size of the source block, a block size of the destination block, a position of the source block, and a position of the destination block are of a same precision as the source motion vector and the destination motion vector.

**[0418]** Aspect 13B. The method of any of aspects 1B-12B, wherein the decoding the current block comprises decoding the current block using subblock prediction, and wherein each subblock is predicted using a respective intersecting motion vector.

**[0419]** Aspect 14B. The method of aspect 13B, wherein decoding the current block using subblock prediction comprises using an affine model.

**[0420]** Aspect 15B. The method of any of aspects 1B-14B, wherein the plurality of subblocks comprises a plurality of 4x4 subblocks.

**[0421]** Aspect 16B. A device for decoding video data, the device comprising: one or more memories for storing the video data; and one or more processors operatively coupled to the one or more memories, the one or more processors configured to: determine, for each of a plurality of subblocks of a current block of a current picture of the video data, a corresponding intersecting motion vector list, each corresponding intersecting motion vector list comprising at least one intersecting motion vector, the at least one intersecting motion vector comprising a source motion vector associated with a source block located in a first reference picture and a destination motion vector associated with a destination block located in a second reference picture, and wherein the source block has a predictor that is a subset of a destination block, the first reference picture being already decoded and temporally prior to the current picture and the second reference picture being already decoded and temporally after the current picture; determine an intersecting motion vector for the current block based on the corresponding intersecting motion vector lists; and decode the current block based on the intersecting motion vector.

**[0422]** Aspect 17B. The device of aspect 16B, wherein the one or more processors are further configured to store each corresponding intersecting motion vector list in the one or more memories.

**[0423]** Aspect 18B. The device of aspect 17B, wherein as part of storing each corresponding intersecting motion vector list, the one or more processors are configured to store, for a first corresponding intersecting motion vector list, only a first number of intersecting motion vectors appearing first in a first corresponding intersecting motion

vector list, the first number of intersecting motion vectors being less than a number of intersecting motion vectors in the first corresponding intersecting motion vector list.

**[0424]** Aspect 19B. The device of any of aspects 16B-18B, wherein as part of determining the corresponding intersecting motion vector list, the one or more processors are configured to determine which intersecting motion vectors are valid, wherein the corresponding intersecting motion vector list comprises intersecting motion vectors determined to be valid and does not comprise intersecting motion vectors determined to not be valid.

**[0425]** Aspect 20B. The device of aspect 19B, wherein as part of determining the corresponding intersecting motion vector list, the one or more processors are configured to sort intersecting motion vectors of a corresponding subblock based on bilateral matching cost so as to place an intersecting motion vector having a lower bilateral matching cost in a position closer to a front of the corresponding intersecting motion vector list than an intersecting motion vector having a higher bilateral matching cost.

**[0426]** Aspect 21B. The device of any of aspects 19B-20B, wherein as part of determining the corresponding intersecting motion vector list, the one or more processors are configured to sort intersecting motion vectors of a corresponding subblock based on an occurrences count of each intersecting motion vector in the corresponding intersecting motion vector list so as to place an intersecting motion vector having a higher occurrence count cost in a position closer to a front of the corresponding intersecting motion vector list than the intersecting motion vector having a lower occurrence count.

**[0427]** Aspect 22B. The device of any of aspects 19B-21B, wherein as part of determining which intersecting motion vectors are valid, the one or more processors are configured to determine that a first intersecting motion vector is valid, and as part of determining that the first intersecting motion vector is valid, the one or more processors are configured to at least one of: determine that the first intersecting motion vector comprises both a first source motion vector and a first destination motion vector in an extended picture boundary; determine that a first absolute value of each of a first horizontal component of the first source motion vector and a first horizontal component of the first destination motion vector is greater than a horizontal threshold; determine that a second absolute value of each of a first vertical component of the first source motion vector and a first vertical component of the first destination motion vector is greater than a vertical threshold; determine, for a pair of intersecting motion vectors comprising the first intersecting motion vector and a second intersecting motion vector, that a difference

between the first horizontal component of the first source motion vector and a second horizontal component of a second source motion vector of the second intersecting motion vector is greater than a difference threshold; determine, for the pair of intersecting motion vectors, that a difference between the first vertical component of the first source motion vector and a second vertical component of the second source motion vector is greater than the difference threshold; determine, for the pair of intersecting motion vectors, that a difference between the first horizontal component of the first destination motion vector and a second horizontal component of a second destination motion vector of the second intersecting motion vector is greater than the difference threshold; or determine, for the pair of intersecting motion vectors, that a difference between the first vertical component of the first destination motion vector and a second vertical component of the second destination motion vector is greater than the difference threshold.

**[0428]** Aspect 23B. The device of any of aspects 16B-22B, wherein as part of determining the intersecting motion vector, the one or more processors are configured to: determine an intersecting motion vector list for the current block based on all of the corresponding intersecting motion vector lists; and determine a value of an index, the value of the index indicative of the intersecting motion vector in the intersecting motion vector list for the current block.

**[0429]** Aspect 24B. The device of aspect 23B, wherein a number of intersecting motion vectors in the intersecting motion vector list for the current block is less than a total number of intersecting motion vectors in the corresponding intersecting motion vector lists.

**[0430]** Aspect 25B. The device of any of aspects 23B-24B, wherein as part of determining the intersecting motion vector list for the current block, the one or more processors are configured to sort intersecting motion vectors of each of the corresponding intersecting motion vector lists based on at least one of bilateral matching cost, occurrence count, or template matching cost.

**[0431]** Aspect 26B. The device of any of aspects 16B-25B, wherein the current block has a top-left position of (X, Y), the source motion vector comprises (srcItcMvHor, srcItcMvVer), the destination motion vector comprises (dstItcMvHor, dstItcMvVer), the source block comprises (srcRefBlk), and the destination block comprises (dstRefBlk), wherein the source block has a top-left position of (srcX, srcY) and the destination block has a top-left position of (dstX, dstY), and wherein srcX is equal to X + srcItcMv,Hor

srcY is equal to Y + srcItcMvVer, dstX is equal to X + dstItcMvHor, dstY is equal to Y + dstItcMvVer.

**[0432]** Aspect 27B. The device of any of aspects 16B-26B, wherein a block size of the source block, a block size of the destination block, a position of the source block, and a position of the destination block are of a same precision as the source motion vector and the destination motion vector.

**[0433]** Aspect 28B. The device of any of aspects 16B-27B, wherein as part of decoding the current block, the one or more processors are configured to decode the current block using subblock prediction, and wherein each subblock is predicted using a respective intersecting motion vector.

**[0434]** Aspect 29B. The device of aspect 28B, wherein as part of decoding the current block using subblock prediction, the one or more processors are configured to use an affine model.

**[0435]** Aspect 30B. The device of any of aspects 16B-29B, further comprising a display configured to display decoded video data.

**[0436]** Aspect 31B. The device of any of aspects 16B-30B, wherein the device comprises one or more of a camera, a computer, a mobile device, a broadcast receiver device, or a set-top box.

**[0437]** Aspect 32B. The device of any of aspects 16B-31B, wherein the plurality of subblocks comprises a plurality of 4x4 subblocks.

**[0438]** Aspect 33B. A method for encoding video data, the method comprising:

determining, for each of a plurality of subblocks of a current block of a current picture of the video data, a corresponding intersecting motion vector list, each corresponding intersecting motion vector list comprising at least one intersecting motion vector, the at least one intersecting motion vector comprising a source motion vector associated with a source block located in a first reference picture and a destination motion vector associated with a destination block located in a second reference picture, and wherein the source block has a predictor that is a subset of the destination block, the first reference picture being already decoded and temporally prior to the current picture and the second reference picture being already decoded and temporally after the current picture; determining an intersecting motion vector for the current block based on the corresponding intersecting motion vector lists; and encoding the current block in accordance with the intersecting motion vector.

**[0439]** Aspect 34B. A device for encoding video data, the device comprising: one or more memories for storing the video data; and one or more processors operatively coupled to the one or more memories, the one or more processors configured to: determine, for each of a plurality of subblocks of a current block of a current picture of video data, a corresponding intersecting motion vector list, each corresponding intersecting motion vector list comprising at least one intersecting motion vector, the at least one intersecting motion vector comprising a source motion vector associated with a source block located in a first reference picture and a destination motion vector associated with a destination block located in a second reference picture, and wherein the source block has a predictor that is a subset of the destination block, the first reference picture being already decoded and temporally prior to the current picture and the second reference picture being already decoded and temporally after the current picture; determine an intersecting motion vector for the current block based on the corresponding intersecting motion vector lists; and encode the current block in accordance with the intersecting motion vector.

**[0440]** Aspect 35B. A non-transitory, computer-readable storage medium having stored thereon instructions that, when executed, cause one or more processors to: determine, for each of a plurality of subblocks of a current block of a current picture of video data, a corresponding intersecting motion vector list, each corresponding intersecting motion vector list comprising at least one intersecting motion vector, the at least one intersecting motion vector comprising a source motion vector associated with a source block located in a first reference picture and a destination motion vector associated with a destination block located in a second reference picture, and wherein the source block has a predictor that is a subset of the destination block, the first reference picture being already decoded and temporally prior to the current picture and the second reference picture being already decoded and temporally after the current picture; determine an intersecting motion vector for the current block based on the corresponding intersecting motion vector lists; and decode the current block based on the intersecting motion vector.

**[0441]** Aspect 36B. A device for decoding video data, the device comprising: means for determining, for each of a plurality of subblocks of a current block of a current picture of the video data, a corresponding intersecting motion vector list, each corresponding intersecting motion vector list comprising at least one intersecting motion vector, the at least one intersecting motion vector comprising a source motion vector associated with a source block located in a first reference picture and a destination motion vector associated with a destination block located in a second reference picture, and wherein the source

block has a predictor that is a subset of the destination block, the first reference picture being already decoded and temporally prior to the current picture and the second reference picture being already decoded and temporally after the current picture; means for determining an intersecting motion vector for the current block based on the corresponding intersecting motion vector lists; and means for decoding the current block based on the intersecting motion vector.

**[0442]** Aspect 1C. A method of decoding video data, the method comprising: determining a current reference picture for a current block of a current picture of the video data based on a reference picture of a collocated block, a reference picture list of the current block, and a picture order count (POC) of a collocated picture, wherein the collocated block is collocated with the current block and is in the collocated picture; and decoding the current block based on a motion vector associated with the current reference picture.

**[0443]** Aspect 2C. The method of aspect 1C, wherein determining the current reference picture comprises determining a collocated reference picture for the collocated block.

**[0444]** Aspect 3C. The method of aspect 2C, wherein determining the current reference picture comprises determining a reference picture from the reference picture list that is associated with a highest number of pictures that overlap pictures between the collocated reference picture and the collocated picture.

**[0445]** Aspect 4C. The method of aspect 3C, wherein determining the current reference picture comprises: determining first pictures that are between the collocated reference picture and the collocated picture in a POC; determining, for each potential current reference picture, corresponding second pictures that are between the potential current reference picture and the current picture in the POC; determining, for each potential current reference picture, a corresponding count of pictures that are both first pictures and corresponding second picture; and determining the current reference picture based on the corresponding counts.

**[0446]** Aspect 5C. The method of aspect 4C, wherein determining the current reference picture based on the corresponding counts comprises selecting, as the current reference picture, the potential current reference picture having a highest number of corresponding counts.

**[0447]** Aspect 6C. The method of any of aspects 3C-5C, further comprising determining that no potential current reference picture in the current reference picture list

has a same POC distance from the current picture as a POC distance from the collocated picture to the collocated reference picture, wherein determining the reference picture from the reference picture list that is associated with a highest number of pictures that overlap pictures between the collocated reference picture and the collocated picture is based on the determining that no potential current reference picture in the current reference picture list has a same POC distance from the current picture as a POC distance from the collocated picture to the collocated reference picture.

**[0448]** Aspect 7C. The method of any of aspects 1C-6C, wherein the motion vector is associated with a temporal motion vector predictor, a subblock temporal motion vector predictor, a spatial merge candidate, or an advanced motion vector prediction candidate.

**[0449]** Aspect 8C. The method of any of aspects 1C-7C, further comprising determining that the current picture is a random-access picture, a low-delay B picture, or a low-delay P picture, wherein determining the current reference picture is based on the determination that the current picture is the random-access picture, the low-delay B picture, or the low-delay P picture.

**[0450]** Aspect 9C. The method of any of aspects 1C-7C, further comprising determining that a current slice comprising the current block is a random-access slice, a low-delay B slice, or a low-delay P slice, wherein determining the current reference picture is based on the determination that the current slice is the random-access slice, the low-delay B slice, or the low-delay P slice.

**[0451]** Aspect 10C. A device for decoding video data, the device comprising: one or more memories for storing the video data; and one or more processors operatively coupled to the one or more memories, the one or more processors configured to: determine a current reference picture for a current block of a current picture of the video data based on a reference picture of a collocated block, a reference picture list of the current block, and a picture order count (POC) of a collocated picture, wherein the collocated block is collocated with the current block and is in the collocated picture; and decode the current block based on a motion vector associated with the current reference picture.

**[0452]** Aspect 11C. The device of aspect 10C, wherein as part of determining the current reference picture, the one or more processors are configured to determine a collocated reference picture for the collocated block.

**[0453]** Aspect 12C. The device of aspect 11C, wherein as part of determining the current reference picture, the one or more processors are configured to determine a reference picture from the reference picture list that is associated with a highest number

of pictures that overlap pictures between the collocated reference picture and the collocated picture.

**[0454]** Aspect 13C. The device of aspect 12C, wherein as part of determining the current reference picture, the one or more processors are configured to: determine first pictures that are between the collocated reference picture and the collocated picture in a POC; determine, for each potential current reference picture, corresponding second pictures that are between the potential current reference picture and the current picture in the POC; determine, for each potential current reference picture, a corresponding count of pictures that are both first pictures and corresponding second picture; and determine the current reference picture based on the corresponding counts.

**[0455]** Aspect 14C. The device of aspect 13C, wherein as part of determining the current reference picture based on the corresponding counts, the one or more processors are configured to select, as the current reference picture, the potential current reference picture having a highest number of corresponding counts.

**[0456]** Aspect 15C. The device of any of aspect 12C-14C, wherein the further the one or more processors are configured to determine that no potential current reference picture in the current reference picture list has a same POC distance from the current picture as a POC distance from the collocated picture to the collocated reference picture, and wherein determining the reference picture from the reference picture list that is associated with a highest number of pictures that overlap pictures between the collocated reference picture and the collocated picture is based on the determination that no potential current reference picture in the current reference picture list has a same POC distance from the current picture as a POC distance from the collocated picture to the collocated reference picture.

**[0457]** Aspect 16C. The device of any of aspects 10C-15C, wherein the motion vector is associated with a temporal motion vector predictor, a subblock temporal motion vector predictor, a spatial merge candidate, or an advanced motion vector prediction candidate.

**[0458]** Aspect 17C. The device of any of aspects 10C-16C, wherein the one or more processors are configured to determine that the current picture is a random-access picture, a low-delay B picture, or a low-delay P picture, wherein the one or more processors are configured to determine the current reference picture is based on the determination that the current picture is the random-access picture, the low-delay B picture, or the low-delay P picture.

**[0459]** Aspect 18C. The device of any of aspects 10C-16C, wherein the one or more processors are configured to determine that a current slice comprising the current block

is a random-access slice, a low-delay B slice, or a low-delay P slice, wherein the one or more processors are configured to determine the current reference picture is based on the determination that the current slice is the random-access slice, the low-delay B slice, or the low-delay P slice.

**[0460]** Aspect 19C. The device of any of aspects 10C-18C, further comprising a display configured to display decoded video data.

**[0461]** Aspect 20C. A method for encoding video data, the method comprising: determining a current reference picture for a current block of a current picture of the video data based on a reference picture of a collocated block, a reference picture list of the current block, and a picture order count (POC) of a collocated picture, wherein the collocated block is collocated with the current block and is in the collocated picture; and encoding the current block in accordance with a motion vector associated with the current reference picture.

**[0462]** Aspect 21C. A device for decoding video data, the device comprising: means for determining a current reference picture for a current block of a current picture of the video data based on a reference picture of a collocated block, a reference picture list of the current block, and a picture order count (POC) of a collocated picture, wherein the collocated block is collocated with the current block and is in the collocated picture; and means for decoding the current block based on a motion vector associated with the current reference picture.

**[0463]** 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.

**[0464]** 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.

**[0465]** By way of example, and not limitation, such computer-readable storage media may include one or more of 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.

**[0466]** Instructions may be executed by one or more processors, such as one or more DSPs, general purpose microprocessors, ASICs, FPGAs, or other equivalent integrated or discrete logic circuitry. Accordingly, the terms "processor" and "processing circuitry," as used herein may refer to any of the foregoing structures or any other structure suitable for implementation of the techniques described herein. In addition, in some aspects, the functionality described herein may be provided within dedicated hardware and/or software modules configured for encoding and decoding, or incorporated in a combined codec. Also, the techniques could be fully implemented in one or more circuits or logic elements.

**[0467]** 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.

**[0468]** 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, for each of a plurality of subblocks of a current block of a current picture of the video data, a corresponding intersecting motion vector list, each corresponding intersecting motion vector list comprising at least one intersecting motion vector, the at least one intersecting motion vector comprising a source motion vector associated with a source block located in a first reference picture and a destination motion vector associated with a destination block located in a second reference picture, and wherein the source block has a predictor that is a subset of the destination block, the first reference picture being already decoded and temporally prior to the current picture and the second reference picture being already decoded and temporally after the current picture;

determining an intersecting motion vector for the current block based on the corresponding intersecting motion vector lists; and

decoding the current block based on the intersecting motion vector.

2. The method of claim 1, further comprising storing each corresponding intersecting motion vector list.

3. The method of claim 2, wherein storing each corresponding intersecting motion vector list comprises storing, for a first corresponding intersecting motion vector list, only a first number of intersecting motion vectors appearing first in the first corresponding intersecting motion vector list, the first number of intersecting motion vectors being less than a number of intersecting motion vectors in the first corresponding intersecting motion vector list.

4. The method of claim 1, wherein determining the corresponding intersecting motion vector list comprises determining which intersecting motion vectors are valid, wherein the corresponding intersecting motion vector list comprises intersecting motion vectors determined to be valid and does not comprise intersecting motion vectors determined to not be valid.

5. The method of claim 4, wherein determining the corresponding intersecting motion vector list further comprises sorting valid intersecting motion vectors of a corresponding subblock based on bilateral matching cost so as to place an intersecting motion vector having a lower bilateral matching cost in a position closer to a front of the corresponding intersecting motion vector list than an intersecting motion vector having a higher bilateral matching cost.

6. The method of claim 4, wherein determining the corresponding intersecting motion vector list further comprises sorting valid intersecting motion vectors of a corresponding subblock based on an occurrences count of each intersecting motion vector in the corresponding intersecting motion vector list so as to place an intersecting motion vector having a higher occurrence count in a position closer to a front of the corresponding intersecting motion vector list than the intersecting motion vector having a lower occurrence count.

7. The method of claim 4, wherein determining which intersecting motion vectors are valid comprises determining that a first intersecting motion vector is valid, wherein determining that the first intersecting motion vector is valid comprises at least one of:

determining that the first intersecting motion vector comprises both a first source motion vector and a first destination motion vector in an extended picture boundary;

determining that a first absolute value of each of a first horizontal component of the first source motion vector and a first horizontal component of the first destination motion vector is greater than a horizontal threshold;

determining that a second absolute value of each of a first vertical component of the first source motion vector and a first vertical component of the first destination motion vector is greater than a vertical threshold;

determining, for a pair of intersecting motion vectors comprising the first intersecting motion vector and a second intersecting motion vector, that a difference between the first horizontal component of the first source motion vector and a second horizontal component of a second source motion vector of the second intersecting motion vector is greater than a difference threshold;

determining, for the pair of intersecting motion vectors, that a difference between the first vertical component of the first source motion vector and a second

vertical component of the second source motion vector is greater than the difference threshold;

determining, for the pair of intersecting motion vectors, that a difference between the first horizontal component of the first destination motion vector and a second horizontal component of a second destination motion vector of the second intersecting motion vector is greater than the difference threshold; or

determining, for the pair of intersecting motion vectors, that a difference between the first vertical component of the first destination motion vector and a second vertical component of the second destination motion vector is greater than the difference threshold.

8. The method of claim 1, wherein determining the intersecting motion vector comprises:

determining an intersecting motion vector list for the current block based on all of the corresponding intersecting motion vector lists; and

determining a value of an index, the value of the index indicative of the intersecting motion vector in the intersecting motion vector list for the current block.

9. The method of claim 8, wherein a number of intersecting motion vectors in the intersecting motion vector list for the current block is less than a total number of intersecting motion vectors in the corresponding intersecting motion vector lists.

10. The method of claim 8, wherein determining the intersecting motion vector list for the current block comprises sorting intersecting motion vectors of each of the corresponding intersecting motion vector lists based on at least one of bilateral matching cost, occurrence count, or template matching cost.

11. The method of claim 1, wherein the current block has a top-left position of (X, Y), the source motion vector comprises (srcItcMvHor, srcItcMvVer), the destination motion vector comprises (dstItcMvHor, dstItcMvVer), the source block comprises (srcRefBlk), and the destination block comprises (dstRefBlk), wherein the source block has a top-left position of (srcX, srcY) and the destination block has a top-left position of (dstX, dstY), and wherein srcX is equal to X + srcItcMvHor, srcY is equal to Y + srcItcMvVer, dstX is equal to X + dstItcMvHor, and dstY is equal to Y + dstItcMvVer.

12. The method of claim 1, wherein a block size of the source block, a block size of the destination block, a position of the source block, and a position of the destination block are of a same precision as the source motion vector and the destination motion vector.

13. The method of claim 1, wherein the decoding the current block comprises decoding the current block using subblock prediction, and wherein each subblock is predicted using a respective intersecting motion vector.

14. The method of claim 13, wherein decoding the current block using subblock prediction comprises using an affine model.

15. The method of claim 1, wherein the plurality of subblocks comprises a plurality of 4x4 subblocks.

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

one or more memories for storing the video data; and

one or more processors operatively coupled to the one or more memories, the one or more processors configured to:

determine, for each of a plurality of subblocks of a current block of a current picture of the video data, a corresponding intersecting motion vector list, each corresponding intersecting motion vector list comprising at least one intersecting motion vector, the at least one intersecting motion vector comprising a source motion vector associated with a source block located in a first reference picture and a destination motion vector associated with a destination block located in a second reference picture, and wherein the source block has a predictor that is a subset of the destination block, the first reference picture being already decoded and temporally prior to the current picture and the second reference picture being already decoded and temporally after the current picture;

determine an intersecting motion vector for the current block based on the corresponding intersecting motion vector lists; and

decode the current block based on the intersecting motion vector.

17. The device of claim 16, wherein the one or more processors are further configured to store each corresponding intersecting motion vector list in the one or more memories.

18. The device of claim 17, wherein as part of storing each corresponding intersecting motion vector list, the one or more processors are configured to store a first number of intersecting motion vectors appearing first in a first corresponding intersecting motion vector list, the first number of intersecting motion vectors being less than the number of intersecting motion vectors in the first corresponding intersecting motion vector list.

19. The device of claim 16, wherein as part of determining the corresponding intersecting motion vector list, the one or more processors are configured to determine which intersecting motion vectors are valid, wherein the corresponding intersecting motion vector list comprises intersecting motion vectors determined to be valid and does not comprise intersecting motion vectors determined to not be valid.

20. The device of claim 19, wherein as part of determining the corresponding intersecting motion vector list, the one or more processors are configured to sort valid intersecting motion vectors of a corresponding subblock based on bilateral matching cost so as to place an intersecting motion vector having a lower bilateral matching cost in a position closer to a front of the corresponding intersecting motion vector list than an intersecting motion vector having a higher bilateral matching cost.

21. The device of claim 19, wherein as part of determining the corresponding intersecting motion vector list, the one or more processors are configured to sort valid intersecting motion vectors of a corresponding subblock based on an occurrences count of each intersecting motion vector in the corresponding intersecting motion vector list so as to place an intersecting motion vector having a higher occurrence count cost in a position closer to a front of the corresponding intersecting motion vector list than the intersecting motion vector having a lower occurrence count.

22. The device of claim 19, wherein as part of determining which intersecting motion vectors are valid, the one or more processors are configured to determine that a first intersecting motion vector is valid, and as part of determining that the first intersecting motion vector is valid, the one or more processors are configured to at least one of:

determine that the first intersecting motion vector comprises both a first source motion vector and a first destination motion vector in an extended picture boundary;

determine that a first absolute value of each of a first horizontal component of the first source motion vector and a first horizontal component of the first destination motion vector is greater than a horizontal threshold;

determine that a second absolute value of each of a first vertical component of the first source motion vector and a first vertical component of the first destination motion vector is greater than a vertical threshold;

determine, for a pair of intersecting motion vectors comprising the first intersecting motion vector and a second intersecting motion vector, that a difference between the first horizontal component of the first source motion vector and a second horizontal component of a second source motion vector of the second intersecting motion vector is greater than a difference threshold;

determine, for the pair of intersecting motion vectors, that a difference between the first vertical component of the first source motion vector and a second vertical component of the second source motion vector is greater than the difference threshold;

determine, for the pair of intersecting motion vectors, that a difference between the first horizontal component of the first destination motion vector and a second horizontal component of a second destination motion vector of the second intersecting motion vector is greater than the difference threshold; or

determine, for the pair of intersecting motion vectors, that a difference between the first vertical component of the first destination motion vector and a second vertical component of the second destination motion vector is greater than the difference threshold.

23. The device of claim 16, wherein as part of determining the intersecting motion vector, the one or more processors are configured to:

determine an intersecting motion vector list for the current block based on all of the corresponding intersecting motion vector lists; and

determine a value of an index, the value of the index indicative of the intersecting motion vector in the intersecting motion vector list for the current block.

24. The device of claim 23, wherein a number of intersecting motion vectors in the intersecting motion vector list for the current block is less than a total number of intersecting motion vectors in the corresponding intersecting motion vector lists.

25. The device of claim 23, wherein as part of determining the intersecting motion vector list for the current block, the one or more processors are configured to sort intersecting motion vectors of each of the corresponding intersecting motion vector lists based on at least one of bilateral matching cost, occurrence count, or template matching cost.

26. The device of claim 16, wherein the current block has a top-left position of (X, Y), the source motion vector comprises (srcItcMvHor, srcItcMvVer), the destination motion vector comprises (dstItcMvHor, dstItcMvVer), the source block comprises (srcRefBlk), and the destination block comprises (dstRefBlk), wherein the source block has a top-left position of (srcX, srcY) and the destination block has a top-left position of (dstX, dstY), and wherein srcX is equal to X + srcItcMvHor, srcY is equal to Y + srcItcMvVer, dstX is equal to X + dstItcMvHor, dstY is equal to Y + dstItcMvVer.

27. The device of claim 16, wherein a block size of the source block, a block size of the destination block, a position of the source block, and a position of the destination block are of a same precision as the source motion vector and the destination motion vector.

28. The device of claim 16, wherein as part of decoding the current block, the one or more processors are configured to decode the current block using subblock prediction, and wherein each subblock is predicted using a respective intersecting motion vector.

29. The device of claim 28, wherein as part of decoding the current block using subblock prediction, the one or more processors are configured to use an affine model.

30. The device of claim 16, further comprising a display configured to display decoded video data.

31. The device of claim 16, wherein the device comprises one or more of a camera, a computer, a mobile device, a broadcast receiver device, or a set-top box.

32. The device of claim 16, wherein the plurality of subblocks comprises a plurality of 4x4 subblocks.

33. A method for encoding video data, the method comprising:

determining, for each of a plurality of subblocks of a current block of a current picture of the video data, a corresponding intersecting motion vector list, each corresponding intersecting motion vector list comprising at least one intersecting motion vector, the at least one intersecting motion vector comprising a source motion vector associated with a source block located in a first reference picture and a destination motion vector associated with a destination block located in a second reference picture, and wherein the source block has a predictor that is a subset of the destination block, the first reference picture being already decoded and temporally prior to the current picture and the second reference picture being already decoded and temporally after the current picture;

determining an intersecting motion vector for the current block based on the corresponding intersecting motion vector lists; and

encoding the current block in accordance with the intersecting motion vector.

34. A device for encoding video data, the device comprising:

one or more memories for storing the video data; and

one or more processors operatively coupled to the one or more memories, the one or more processors configured to:

determine, for each of a plurality of subblocks of a current block of a current picture of video data, a corresponding intersecting motion vector list, each corresponding intersecting motion vector list comprising at least one intersecting motion vector, the at least one intersecting motion vector comprising a source motion vector associated with a source block located in a first reference picture and a destination motion vector associated with a destination block located in a second reference picture, and wherein the source block has a predictor that is a subset of the destination block, the first reference picture being already decoded and temporally prior to the current picture and the second reference picture being already decoded and temporally after the current picture;

determine an intersecting motion vector for the current block based on the corresponding intersecting motion vector lists; and

encode the current block in accordance with on the intersecting motion vector.