# LIMITATION OF THE MVP DERIVATION BASED ON DECODER-SIDE MOTION VECTOR DERIVATION

**[0001]** This Application claims priority to U.S. Patent Application No. 16/384,044, filed April 15, 2019, which claims the benefit of U.S. Provisional Patent Application 62/659,046, filed April 17, 2018, the entire contents of which are hereby incorporated by reference.

## 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 compression techniques, such as those described in the standards defined by MPEG-2, MPEG-4, ITU-T H.263, ITU-T H.264/MPEG-4, Part 10, Advanced Video Coding (AVC), the recently finalized High Efficiency Video Coding (HEVC) standard, and extensions of such standards. The video devices may transmit, receive, encode, decode, and/or store digital video information more efficiently by implementing such video compression techniques.

**[0004]** Video compression techniques perform 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 (i.e., a video frame or a portion of a video frame) may be partitioned into video blocks, which may also be referred to as treeblocks, coding units (CUs) and/or coding nodes. Video blocks in an intra-coded (I) slice of a picture are encoded using spatial prediction with respect to reference samples in neighboring blocks in the same picture. Video blocks in an inter-coded (P or B) slice of a picture may use spatial prediction with respect to reference samples in neighboring blocks in the same picture or temporal prediction with respect to reference samples in other reference pictures. Pictures may be referred to as frames, and reference pictures may be referred to a reference frames.

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

## SUMMARY

**[0006]** This disclosure describes techniques related to decoder-side motion vector derivation (DMVD). The techniques of this disclosure may be used in conjunction with existing video codecs, such as the High Efficiency Video Coding (HEVC) standard or may be used as an efficient coding tool in any future video coding standards.

**[0007]** According to one example, a method of decoding video data includes determining a first block of video data is coded in an inter prediction mode; determining a first motion vector for the first block of video data; performing motion vector refinement on the first motion vector for the first block to determine a refined motion vector for the first block of video data; locating a first reference block in a first reference picture using the refined motion vector; generating a first predictive block for the first block of video data based on the first reference block; determining that a second block of video data is coded in a mode that utilizes a motion vector associated with the first block as a motion vector predictor; in response to determining that the second block of video data is coded in the mode that utilizes the motion vector associated with the first block as a motion vector predictor and in response to performing the motion vector refinement on the first motion vector for the first block, using a different motion vector than the first refined motion vector as the motion vector predictor associated with the first block; based on the different motion vector, determining a second motion vector for the second block; locating a second reference block in a second reference picture using

the second motion vector; and decoding a picture of video data based on the first reference block and the second reference block.

**[0008]** According to another example, a device for decoding video data includes a memory configured to store video data and one or more processors configured to determine a first block of the video data is coded in an inter prediction mode; determine a first motion vector for the first block of the video data; perform motion vector refinement on the first motion vector for the first block to determine a refined motion vector for the first block of the video data; locate a first reference block in a first reference picture using the refined motion vector; generate a first predictive block for the first block of the video data based on the first reference block; determine that a second block of the video data is coded in a mode that utilizes a motion vector associated with the first block as a motion vector predictor; in response to determining that the second block of the video data is coded in the mode that utilizes the motion vector associated with the first block as a motion vector predictor and in response to performing the motion vector refinement on the first motion vector for the first block, use a different motion vector than the first refined motion vector as the motion vector predictor associated with the first block; based on the different motion vector, determine a second motion vector for the second block; locate a second reference block in a second reference picture using the second motion vector; and decode a picture of the video data based on the first reference block and the second reference block.

**[0009]** According to another example, a computer-readable storage medium stores instructions that when executed by one or more processors causes the one or more processors to determine a first block of video data is coded in an inter prediction mode; determine a first motion vector for the first block of video data; perform motion vector refinement on the first motion vector for the first block to determine a refined motion vector for the first block of video data; locate a first reference block in a first reference picture using the refined motion vector; generate a first predictive block for the first block of video data based on the first reference block; determine that a second block of video data is coded in a mode that utilizes a motion vector associated with the first block as a motion vector predictor; in response to determining that the second block of video data is coded in the mode that utilizes the motion vector associated with the first block as a motion vector predictor and in response to performing the motion vector refinement on the first motion vector for the first block, use a different motion vector than the first refined motion vector as the motion vector predictor associated with the

first block; based on the different motion vector, determine a second motion vector for the second block; locate a second reference block in a second reference picture using the second motion vector; and decode a picture of video data based on the first reference block and the second reference block.

**[0010]** According to another example, an apparatus for decoding video data includes means for determining a first block of video data is coded in an inter prediction mode; means for determining a first motion vector for the first block of video data; means for performing motion vector refinement on the first motion vector for the first block to determine a refined motion vector for the first block of video data; means for locating a first reference block in a first reference picture using the refined motion vector; means for generating a first predictive block for the first block of video data based on the first reference block; means for determining that a second block of video data is coded in a mode that utilizes a motion vector associated with the first block as a motion vector predictor; means for using a different motion vector than the first refined motion vector as the motion vector predictor associated with the first block in response to determining that the second block of video data is coded in the mode that utilizes the motion vector associated with the first block as a motion vector predictor and in response to performing the motion vector refinement on the first motion vector for the first block; means for determining a second motion vector for the second block based on the different motion vector; means for locating a second reference block in a second reference picture using the second motion vector; and means for decoding a picture of video data based on the first reference block and the second reference block.

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

## BRIEF DESCRIPTION OF DRAWINGS

**[0012]** FIG. 1 is a block diagram illustrating an example video encoding and decoding system that may utilize the techniques of this disclosure for supporting decoder-side motion vector derivation.

**[0013]** FIG. 2A is a conceptual diagram showing an example of spatial neighboring motion vector candidates for merge mode.

**[0014]** FIG. 2B is a conceptual diagram showing an example of spatial neighboring motion vector candidates for an advanced motion vector prediction mode.

**[0015]** FIG. 3A is a conceptual diagram showing an example of a temporal motion vector predictor candidate.

**[0016]** FIG. 3B is a conceptual timing diagram showing an example of motion vector scaling.

**[0017]** FIG. 4 is a conceptual diagram showing an example of bilateral matching.

**[0018]** FIG. 5 is a conceptual diagram showing an example of template matching.

**[0019]** FIGS. 6A and 6B are flow diagrams showing example proposed modifications to frame-rate up conversion template matching mode.

**[0020]** FIG. 7 is a conceptual diagram showing an example of optical flow trajectory.

**[0021]** FIG. 8 is a conceptual diagram showing an example of bi-directional optical flow for an 8x4 block.

**[0022]** FIG. 9 is a conceptual diagram showing an example of proposed decoder-side motion vector derivation based on bilateral template matching.

**[0023]** FIGS. 10A and 10B are conceptual diagrams showing an example illustration of sub-blocks where overlapped block motion compensation may apply.

**[0024]** FIGS. 11A-11D are conceptual diagrams showing examples of overlapped block motion compensation weightings.

**[0025]** FIG. 12 is a block diagram illustrating an example of a video encoder that may implement techniques supporting decoder-side motion vector derivation.

**[0026]** FIG. 13 is a block diagram illustrating an example of a video decoder, which decodes an encoded video sequence and performs decoder-side motion vector derivation.

**[0027]** FIG. 14 is a flow diagram illustrating an example video decoding technique described in this disclosure.

## DETAILED DESCRIPTION

**[0028]** This disclosure describes techniques related to decoder-side motion vector derivation (DMVD). The techniques of this disclosure may be used in conjunction with existing video codecs, such as the High Efficiency Video Coding (HEVC) standard or may be used as an efficient coding tool in any future video coding standards.

**[0029]** Various techniques in this disclosure may be described with reference to a video coder, which is intended to be a generic term that can refer to either a video encoder or a video decoder. Unless explicitly stated otherwise, it should not be assumed that techniques described with respect to a video encoder or a video decoder cannot be

performed by the other of a video encoder or a video decoder. For example, in many instances, a video decoder performs the same, or sometimes a reciprocal, coding technique as a video encoder in order to decode encoded video data. In many instances, a video encoder also includes a video decoding loop, and thus the video encoder performs video decoding as part of encoding video data. Thus, unless stated otherwise, the techniques described in this disclosure with respect to a video decoder may also be performed by a video encoder, and vice versa.

**[0030]** This disclosure may also use terms such as current layer, current block, current picture, current slice, etc. In the context of this disclosure, the term current is intended to identify a layer, block, picture, slice, etc. that is currently being coded, as opposed to, for example, previously coded layers, blocks, pictures, and slices or yet to be coded blocks, pictures, and slices.

**[0031]** FIG. 1 is a block diagram illustrating an example video encoding and decoding system 10 that may utilize the techniques described in this disclosure. As shown in FIG. 1, system 10 includes a source device 12 that generates encoded video data to be decoded at a later time by a destination device 14. Source device 12 and destination device 14 may comprise any of a wide range of devices, including desktop computers, notebook (i.e., laptop) computers, tablet computers, set-top boxes, telephone handsets such as so-called "smart" phones, so-called "smart" pads, televisions, cameras, display devices, digital media players, video gaming consoles, video streaming device, or the like. In some cases, source device 12 and destination device 14 may be equipped for wireless communication.

**[0032]** Destination device 14 may receive the encoded video data to be decoded via a link 16. Link 16 may comprise any type of medium or device capable of moving the encoded video data from source device 12 to destination device 14. In one example, link 16 may comprise a communication medium to enable source device 12 to transmit encoded video data directly to destination device 14 in real-time. The encoded video data may be modulated according to a communication standard, such as a wireless communication protocol, and transmitted to destination device 14. The communication medium may comprise any wireless or wired communication medium, such as a radio frequency (RF) spectrum or one or more physical transmission lines. The communication medium may form part of a packet-based network, such as a local area network, a wide-area network, or a global network such as the Internet. The communication medium may include routers, switches, base stations, or any other

equipment that may be useful to facilitate communication from source device 12 to destination device 14.

**[0033]** In another example, encoded data may be output from output interface 22 to a storage device 26. Similarly, encoded data may be accessed from storage device 26 by input interface. Storage device 26 may include any of a variety of distributed or locally accessed data storage media such as a hard drive, Blu-ray discs, DVDs, CD-ROMs, flash memory, volatile or non-volatile memory, or any other suitable digital storage media for storing encoded video data. In a further example, storage device 26 may correspond to a file server or another intermediate storage device that may hold the encoded video generated by source device 12. Destination device 14 may access stored video data from storage device 26 via streaming or download. The file server may be any type of server capable of storing encoded video data and transmitting that encoded video data to the destination device 14. Example file servers include a web server (e.g., for a website), an FTP server, network attached storage (NAS) devices, or a local disk drive. Destination device 14 may access the encoded video data through any standard data connection, including an Internet connection. This may include a wireless channel (e.g., a Wi-Fi connection), a wired connection (e.g., DSL, cable modem, etc.), or a combination of both that is suitable for accessing encoded video data stored on a file server. The transmission of encoded video data from storage device 26 may be a streaming transmission, a download transmission, or a combination of both.

**[0034]** The techniques of this disclosure are not necessarily limited to wireless applications or settings. The techniques may be applied to video coding in support of any of a variety of multimedia applications, such as over-the-air television broadcasts, cable television transmissions, satellite television transmissions, streaming video transmissions, e.g., via the Internet, encoding of digital video for storage on a data storage medium, decoding of digital video stored on a data storage medium, or other applications. In some examples, system 10 may be configured to support one-way or two-way video transmission to support applications such as video streaming, video playback, video broadcasting, and/or video telephony.

**[0035]** In the example of FIG. 1, source device 12 includes a video source 18, video encoder 20 and an output interface 22. In some cases, output interface 22 may include a modulator/demodulator (modem) and/or a transmitter. In source device 12, video source 18 may include a source such as a video capture device, e.g., a video camera, a video archive containing previously captured video, a video feed interface to receive

video from a video content provider, and/or a computer graphics system for generating computer graphics data as the source video, or a combination of such sources. As one example, if video source 18 is a video camera, source device 12 and destination device 14 may form so-called camera phones or video phones. However, the techniques described in this disclosure may be applicable to video coding in general, and may be applied to wireless and/or wired applications.

**[0036]** The captured, pre-captured, or computer-generated video may be encoded by video encoder 20. The encoded video data may be transmitted directly to destination device 14 via output interface 22 of source device 12. The encoded video data may also (or alternatively) be stored onto storage device 26 for later access by destination device 14 or other devices, for decoding and/or playback.

**[0037]** Destination device 14 includes an input interface 28, a video decoder 30, and a display device 32. In some cases, input interface 28 may include a receiver and/or a modem. Input interface 28 of destination device 14 receives the encoded video data over link 16. The encoded video data communicated over link 16, or provided on storage device 26, may include a variety of syntax elements generated by video encoder 20 for use by a video decoder, such as video decoder 30, in decoding the video data. Such syntax elements may be included with the encoded video data transmitted on a communication medium, stored on a storage medium, or stored a file server.

**[0038]** Display device 32 may be integrated with, or external to, destination device 14. In some examples, destination device 14 may include an integrated display device and also be configured to interface with an external display device. In other examples, destination device 14 may be a display device. In general, display device 32 displays the decoded video data to a user, and may comprise 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.

**[0039]** Video encoder 20 and video decoder 30 may operate according to a video compression standard, such as HEVC standard, and may conform to the HEVC Test Model (HM). Video encoder 20 and video decoder 30 may additionally operate according to an HEVC extension, such as the range extension, the multiview extension (MV-HEVC), or the scalable extension (SHVC) which have 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).

**[0040]** Video encoder 20 and video decoder 30 may also operate according to other proprietary or industry standards, such as the ITU-T H.264 standard, alternatively referred to as ISO/IEC MPEG-4, Part 10, Advanced Video Coding (AVC), or extensions of such standards, such as the Scalable Video Coding (SVC) and Multi-view Video Coding (MVC) extensions. The techniques of this disclosure, however, are not limited to any particular coding standard. Other examples of video compression 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, and ISO/IEC MPEG-4 Visual.

**[0041]** ITU-T VCEG (Q6/16) and ISO/IEC MPEG (JTC 1/SC 29/WG 11) are now studying the potential need for standardization of future video coding technology with a compression capability that significantly exceeds that of the current HEVC standard (including its current extensions and near-term extensions for screen content coding and high-dynamic-range coding). The groups are working together on this exploration activity in a joint collaboration effort known as the Joint Video Exploration Team (JVET) to evaluate compression technology designs proposed by their experts in this area. The JVET first met during 19–21 October 2015. And the latest version of reference software, i.e., Joint Exploration Model 5 (JEM 5) could be downloaded from: https://jvet.hhi.fraunhofer.de/svn/svn_HMJEMSoftware/tags/HM-16.6-JEM-5.0/. Algorithm description of Joint Exploration Test Model 5 (JEM5) may be referred to as JVET-E1001.

**[0042]** Additionally or alternatively, video encoder 20 and video decoder 30 may operate according to other proprietary or industry standards, such as the Joint Exploration Test Model (JEM) or ITU-T H.266, also referred to as Versatile Video Coding (VVC). A recent draft of the VVC standard is described in Bross, et al. "Versatile Video Coding (Draft 4)," Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 13$^{th}$ Meeting: Marrakech, MA, 9-18 January 2019, JVET-M1001-v5 (hereinafter "VVC Draft 4"). The techniques of this disclosure, however, are not limited to any particular coding standard.

**[0043]** Techniques of this disclosure may utilize HEVC terminology for ease of explanation. It should not be assumed, however, that the techniques of this disclosure are limited to HEVC, and in fact, it is explicitly contemplated that the techniques of this disclosure may be implemented in successor standards to HEVC and its extensions. Video encoder 20 and video decoder 30 may encode and decode video data according to multiple standards.

**[0044]** Although not shown in FIG. 1, in some aspects, video encoder 20 and video decoder 30 may each be integrated with an audio encoder and decoder, and may include appropriate MUX-DEMUX units, or other hardware and software, to handle encoding of both audio and video in a common data stream or separate data streams. If applicable, in some examples, MUX-DEMUX units may conform to the ITU H.223 multiplexer protocol, or other protocols such as the user datagram protocol (UDP).

**[0045]** Video encoder 20 and video decoder 30 each may be implemented as any of a variety of suitable encoder circuitry or decoder circuitry, such as one or more microprocessors, digital signal processors (DSPs), application specific integrated circuits (ASICs), field programmable gate arrays (FPGAs), discrete logic, software, hardware, firmware or any combinations thereof. When the techniques are implemented partially in software, a device may store instructions for the software in a suitable, non-transitory computer-readable medium and execute the instructions in hardware using one or more processors to perform the techniques of this disclosure. Each of video encoder 20 and video decoder 30 may be included in one or more encoders or decoders, either of which may be integrated as part of a combined encoder/decoder (CODEC) in a respective device.

**[0046]** In HEVC and other video coding specifications, a video sequence typically includes a series of pictures. Pictures may also be referred to as "frames." In one example approach, a picture may include three sample arrays, denoted $S_L$, $S_{Cb}$, and $S_{Cr}$. In such an example approach, $S_L$ is a two-dimensional array (i.e., a block) of luma samples. $S_{Cb}$ is a two-dimensional array of Cb chrominance samples. $S_{Cr}$ is a two-dimensional array of Cr chrominance samples. Chrominance samples may also be referred to herein as "chroma" samples. In other instances, a picture may be monochrome and may only include an array of luma samples.

**[0047]** To generate an encoded representation of a picture, video encoder 20 may generate a set of coding tree units (CTUs). Each of the CTUs may comprise a coding tree block of luma samples, two corresponding coding tree blocks of chroma samples, and syntax structures used to code the samples of the coding tree blocks. In monochrome pictures or pictures having three separate color planes, a CTU may comprise a single coding tree block and syntax structures used to code the samples of the coding tree block. A coding tree block may be an NxN block of samples. A CTU may also be referred to as a "tree block" or a "largest coding unit" (LCU). The CTUs of HEVC may be broadly analogous to the macroblocks of other standards, such as

H.264/AVC. However, a CTU is not necessarily limited to a particular size and may include one or more coding units (CUs). A slice may include an integer number of CTUs ordered consecutively in a raster scan order.

**[0048]** To generate a coded CTU, video encoder 20 may recursively perform quad-tree partitioning on the coding tree blocks of a CTU to divide the coding tree blocks into coding blocks, hence the name "coding tree units." A coding block may be an NxN block of samples. A CU may comprise a coding block of luma samples and two corresponding coding blocks of chroma samples of a picture that has a luma sample array, a Cb sample array, and a Cr sample array, and syntax structures used to code the samples of the coding blocks. In monochrome pictures or pictures having three separate color planes, a CU may comprise a single coding block and syntax structures used to code the samples of the coding block.

**[0049]** Video encoder 20 may partition a coding block of a CU into one or more prediction blocks. A prediction block is a rectangular (i.e., square or non-square) block of samples on which the same prediction is applied. A prediction unit (PU) of a CU may comprise a prediction block of luma samples, two corresponding prediction blocks of chroma samples, and syntax structures used to predict the prediction blocks. In monochrome pictures or pictures having three separate color planes, a PU may comprise a single prediction block and syntax structures used to predict the prediction block. Video encoder 20 may generate predictive luma, Cb, and Cr blocks for luma, Cb, and Cr prediction blocks of each PU of the CU.

**[0050]** Video encoder 20 may use intra prediction or inter prediction to generate the predictive blocks for a PU. If video encoder 20 uses intra prediction to generate the predictive blocks of a PU, video encoder 20 may generate the predictive blocks of the PU based on decoded samples of the picture associated with the PU. If video encoder 20 uses inter prediction to generate the predictive blocks of a PU, video encoder 20 may generate the predictive blocks of the PU based on decoded samples of one or more pictures other than the picture associated with the PU.

**[0051]** After video encoder 20 generates predictive luma, Cb, and Cr blocks for one or more PUs of a CU, video encoder 20 may generate a luma residual block for the CU. Each sample in the CU's luma residual block indicates a difference between a luma sample in one of the CU's predictive luma blocks and a corresponding sample in the CU's original luma coding block. In addition, video encoder 20 may generate a Cb residual block for the CU. Each sample in the CU's Cb residual block may indicate a

difference between a Cb sample in one of the CU's predictive Cb blocks and a corresponding sample in the CU's original Cb coding block. Video encoder 20 may also generate a Cr residual block for the CU. Each sample in the CU's Cr residual block may indicate a difference between a Cr sample in one of the CU's predictive Cr blocks and a corresponding sample in the CU's original Cr coding block.

**[0052]** Furthermore, video encoder 20 may use quad-tree partitioning to decompose the luma, Cb, and Cr residual blocks of a CU into one or more luma, Cb, and Cr transform blocks. A transform block is a rectangular (e.g., square or non-square) block of samples on which the same transform is applied. A transform unit (TU) of a CU may comprise a transform block of luma samples, two corresponding transform blocks of chroma samples, and syntax structures used to transform the transform block samples. Thus, each TU of a CU may be associated with a luma transform block, a Cb transform block, and a Cr transform block. The luma transform block associated with the TU may be a sub-block of the CU's luma residual block. The Cb transform block may be a sub-block of the CU's Cb residual block. The Cr transform block may be a sub-block of the CU's Cr residual block. In monochrome pictures or pictures having three separate color planes, a TU may comprise a single transform block and syntax structures used to transform the samples of the transform block.

**[0053]** Video encoder 20 may apply one or more transforms to a luma transform block of a TU to generate a luma coefficient block for the TU. A coefficient block may be a two-dimensional array of transform coefficients. A transform coefficient may be a scalar quantity. Video encoder 20 may apply one or more transforms to a Cb transform block of a TU to generate a Cb coefficient block for the TU. Video encoder 20 may apply one or more transforms to a Cr transform block of a TU to generate a Cr coefficient block for the TU.

**[0054]** After generating a coefficient block (e.g., a luma coefficient block, a Cb coefficient block or a Cr coefficient block), video encoder 20 may quantize the coefficient block. 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. After video encoder 20 quantizes a coefficient block, video encoder 20 may entropy encode syntax elements indicating the quantized transform coefficients. For example, video encoder 20 may perform Context-Adaptive Binary Arithmetic Coding (CABAC) on the syntax elements indicating the quantized transform coefficients.

**[0055]** Video encoder 20 may output a bitstream that includes a sequence of bits that forms a representation of coded pictures and associated data. The bitstream may comprise a sequence of Network Abstraction Layer (NAL) units. A NAL unit is a syntax structure containing an indication of the type of data in the NAL unit and bytes containing that data in the form of a raw byte sequence payload (RBSP) interspersed as necessary with emulation prevention bits. Each of the NAL units includes a NAL unit header and encapsulates a RBSP. The NAL unit header may include a syntax element that indicates a NAL unit type code. The NAL unit type code specified by the NAL unit header of a NAL unit indicates the type of the NAL unit. A RBSP may be a syntax structure containing an integer number of bytes that is encapsulated within a NAL unit. In some instances, an RBSP includes zero bits.

**[0056]** Different types of NAL units may encapsulate different types of RBSPs. For example, a first type of NAL unit may encapsulate an RBSP for a PPS, a second type of NAL unit may encapsulate an RBSP for a coded slice, a third type of NAL unit may encapsulate an RBSP for SEI messages, and so on. NAL units that encapsulate RBSPs for video coding data (as opposed to RBSPs for parameter sets and SEI messages) may be referred to as VCL NAL units.

**[0057]** Video decoder 30 may receive a bitstream generated by video encoder 20. In addition, video decoder 30 may parse the bitstream to obtain syntax elements from the bitstream. Video decoder 30 may reconstruct the pictures of the video data based at least in part on the syntax elements obtained from the bitstream. The process to reconstruct the video data may be generally reciprocal to the process performed by video encoder 20. In addition, video decoder 30 may inverse quantize coefficient blocks associated with TUs of a current CU. Video decoder 30 may perform inverse transforms on the coefficient blocks to reconstruct transform blocks associated with the TUs of the current CU. Video decoder 30 may reconstruct the coding blocks of the current CU by adding the samples of the predictive blocks for PUs of the current CU to corresponding samples of the transform blocks of the TUs of the current CU. By reconstructing the coding blocks for each CU of a picture, video decoder 30 may reconstruct the picture.

**[0058]** In HEVC, the largest coding unit in a slice is called a coding tree block (CTB) or coding tree unit (CTU). A CTB contains a quad-tree, the nodes of which are coding units. The size of a CTB can range from 16x16 to 64x64 in the HEVC main profile (although technically 8x8 CTB sizes can be supported). A CU may be as large as a

CTB or as small as 8x8 or a size in between the two. Each CU is typically coded using one coding mode. When a CU is inter coded, the inter coded CU may be further partitioned into 2 or 4 PUs or have just one PU when further partitioning does not apply. When two PUs are present in one CU, the two PUs can be half size rectangles or two rectangles with sizes that are ¼ or ¾ the size of the CU. When the CU is inter coded, one set of motion information is present for each PU. In addition, each PU is coded with a unique inter-prediction mode to derive the set of motion information.

**[0059]** In order to reduce the bit rate needed to transmit motion information (e.g., motion vectors, reference indexes, and/or motion vector precision), video coding standards typically use different types of motion vector prediction. In the HEVC standard, for example, there are two inter prediction modes, named merge mode (with skip mode being considered a special case of merge mode) and advanced motion vector prediction (AMVP) mode respectively for a PU.

**[0060]** In either AMVP or merge mode, video decoder 30 maintains a motion vector (MV) candidate list for multiple motion vector predictors. Video decoder 30 generates the motion vector(s), as well as reference indices in the merge mode, of the current PU by taking one candidate from the MV candidate list. Both video encoder 20 and video decoder 30 generate the same candidate lists. This disclosure will describe motion vector prediction from the perspective of video decoder 30, but it should be understood that video encoder 20 generally implements the same techniques.

**[0061]** In the base HEVC standard, the MV candidate list contains up to five candidates for the merge mode and only two candidates for the AMVP mode, although other standards may use different numbers of candidates. A merge candidate may contain a set of motion information, e.g., motion vectors corresponding to one or both reference picture lists (list 0 and list 1) and the reference indices. If a merge candidate is identified by a merge index, then video decoder 30 uses the motion vectors and reference picture indices of the identified merge candidate for the prediction of the current block. However, under AMVP mode for each potential prediction direction from either list 0 or list 1, a reference index needs to be explicitly signaled, together with an 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.

**[0062]** As can be seen above, a merge candidate corresponds to a full set of motion information while an AMVP candidate contains just one motion vector for a specific

prediction direction and for a reference index. The candidates for both modes may be derived similarly from the same spatial and temporal neighboring blocks.

**[0063]** FIG. 2A is a conceptual diagram showing an example of spatial neighboring motion vector candidates for merge mode. Video decoder 30 may generate a candidate list by adding the motion information of spatial neighboring candidates to the candidate list. Spatial MV candidates are derived from the neighboring blocks shown in FIGS. 2A and 2B, for a specific PU ($PU_0$), although the methods generating the candidates from the blocks differ for merge and AMVP modes. In merge mode, up to four spatial MV candidates can be derived for block 200 (PU0) with the orders shown in FIG. 2A. The order is the following: left (0, A1), above (1, B1), above right (2, B0), below left (3, A0), and above left (4, B2), as shown in FIG. 2A.

**[0064]** FIG. 2B is a conceptual diagram showing an example of spatial neighboring motion vector candidates for an advanced motion vector prediction mode In AVMP mode, the neighboring blocks of block 202 (PU0) are divided into two groups: a left group including block 0 and 1, and an above group including blocks 2, 3, and 4, as shown in FIG. 2B. For each group, the potential candidate in a neighboring block referring to the same reference picture as that indicated by the signaled reference index has the highest priority to be chosen to form a final candidate of the group. It is possible that all neighboring blocks do not contain a motion vector pointing to the same reference picture. Therefore, if such a candidate cannot be found, the first available candidate will be scaled to form the final candidate, thus the temporal distance differences can be compensated.

**[0065]** Video encoder 20 and video decoder 30 may perform temporal motion vector prediction (TMVP) as in the HEVC standard. Video decoder 30 may add a TMVP candidate, if enabled and available, 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, in HEVC, the target reference index for the TMVP candidate in the merge mode is always set to 0.

**[0066]** FIG. 3A is a conceptual diagram showing an example of a temporal motion vector predictor candidate for block 204 (PU0). The primary block location for TMVP candidate derivation is the bottom right block outside of the collocated PU as shown in FIG. 3A as a block "T", to compensate the bias to the above and left blocks used to generate spatial neighboring candidates. However, if that block is located outside of the

current CTB row or motion information is not available, the block is substituted with a center block of the PU.

**[0067]** Video decoder 30 may derive a motion vector for the TMVP candidate from the co-located PU of the co-located picture, indicated in the slice level. The motion vector for the co-located PU is called collocated MV. A block in a reference picture may, for example, be considered to be co-located to a block in a current picture if the block in the reference picture and the current block each include at least one pixel corresponding to a same relative position in the reference picture and the current picture.

**[0068]** FIG. 3B is a conceptual timing diagram showing an example of motion vector scaling process 206. Similar to temporal direct mode in AVC, to derive the TMVP candidate motion vector, video decoder 30 may scale the co-located MV to compensate for the temporal distance differences, as shown in FIG. 3B. With motion vector scaling, it is generally assumed that the value of motion vectors is proportional to the distance of pictures in the presentation time. A motion vector associates two pictures, the reference picture, and the picture containing the motion vector (namely the containing picture). When a motion vector is utilized to predict the other motion vector, the distance of the containing picture and the reference picture is calculated based on the Picture Order Count (POC) values.

**[0069]** When a motion vector is being predicted, its reference picture and the reference picture of the motion vector predictor may be different. Therefore, a new distance (based on POC) is calculated. And the motion vector is scaled based on these two POC distances. In HEVC, motion vector scaling applies to both TMVP and AMVP for spatial and temporal neighboring candidates.

**[0070]** With respect to artificial motion vector candidate generation, if a motion vector candidate list is not complete, then video decoder 30 may generate artificial motion vector candidates insert that artificial motion vector candidates at the end of the list until the list is full or until options for artificial candidates are exhausted.

**[0071]** In merge mode, there are two types of artificial MV candidates: combined candidate derived only for B-slices and zero candidates used only if the first type does not provide enough artificial candidates.

**[0072]** For each pair of candidates that are already in the candidate list and have necessary motion information, video decoder 30 may derive bi-directional combined motion vector candidates 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.

**[0073]** With respect to the 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 due to candidate duplication in the list. To help reduce this inefficiency, video decoder 30 may apply a pruning process. As part of the pruning process, video decoder 30 compares one candidate against the others in the current candidate list to avoid inserting identical candidate in certain extent. To reduce the complexity, only a limited number of pruning processes may be applied instead of comparing each potential candidate with all the other existing candidates.

**[0074]** The JEM reference software includes several inter coding tools that utilize DMVD to derive or refine the motion vector for a current block. One such DMVD tool is pattern matched motion vector derivation (PMMVD) mode, which is a special merge mode based on Frame-Rate Up Conversion (FRUC) techniques. When implementing the JEM reference software, in PMMVD mode, video decoder 30 may derive motion information for a block rather than receive explicit signaling.

**[0075]** Video decoder 30 may receive A FRUC flag for a CU when a merge flag for the CU is true. When the FRUC flag is false, then video decoder 30 may receive a merge index and use the regular merge mode. When the FRUC flag is true, video decoder 30 may receive an additional FRUC mode flag to indicate which method (e.g., bilateral matching or template matching) is to be used to derive motion information for the block. The syntax table to code flags for FRUC is as follows:

| | |
|---|---|
| fruc_flag | u(1) |
| if(fruc_flag){ | |
| if(slice_type != P_slice){ | |
| fruc_mode | u(1) |
| } | |
| } | |

**[0076]** During the motion derivation process, video decoder 30 may first derive an initial motion vector for the whole CU based on bilateral matching or template matching. First, the merge list of the CU, or called PMMVD seeds, is checked and the candidate which leads to the minimum matching cost is selected as the starting point. Then a local search based on bilateral matching or template matching around the starting point is performed and the MV results in the minimum matching cost is taken as

the MV for the whole CU. Subsequently, the motion information is further refined at sub-block level with the derived CU motion vectors as the starting points.

**[0077]** FIG. 4 is a conceptual diagram showing an example of bilateral matching. As shown in the FIG. 4, bilateral matching is used to derive motion information of the current block (Cur) by finding the best match between two reference blocks ($R_0$ and $R_1$) along the motion trajectory of the current block in two different reference pictures (Ref0 and Ref1). The motion trajectory may include the path that a pixel in a block follows through space and time when considering an image sequence (e.g., reference frames and the current frame) as a 3-dimensional continuous spatio-temporal field. Under the assumption of continuous motion trajectory, the motion vectors MV0 and MV1 pointing to the two reference blocks ($R_0$ and $R_1$) are proportional to the temporal distances between the current picture (Cur) and the two reference pictures (Ref0 and Ref1). Derived MVs are derived using bilateral matching and point to reference blocks $R'_0$ and $R'_1$ respectively.

**[0078]** As shown in FIG. 4, video decoder 30 uses bilateral matching to derive motion information of the current block 208 in picture 210 by finding the best match between two reference blocks along the motion trajectory of the current block in two different reference pictures (e.g., reference pictures 212 (Ref0) and 214 (Ref1)). In FIG. 4, video decoder 30 finds reference block 216 ($R_0$) and reference block 218 ($R_1$) as best matches along motion vector MV0. Likewise, video decoder 30 finds reference block 220 ($R'_0$) and reference block 222 ($R'_1$) as best matches along motion vector MV1. Under the assumption of continuous motion trajectory, the motion vectors MV0 and MV1 pointing to the two reference blocks shall be proportional to the temporal distances between the current picture and the two reference pictures. As a special case, when the current picture is temporally between the two reference pictures and the temporal distance from the current picture to the two reference pictures is the same, the bilateral matching becomes mirror based bi-directional MV.

**[0079]** FIG. 5 is a conceptual diagram showing an example of template matching. As shown in FIG. 5, template matching is used to derive motion information of the current block (Cur) by finding the best match between a template (top and/or left neighboring blocks of the current block) in the current picture and a block (same size to the template) in a reference picture (Ref0 and Ref1). A template may include neighboring pixels of a block that is used to compare a block of interest (Cur) with candidate references ($R_0$ with MV0 and $R_1$ with MV1) or derived references ($R'_0$ with MV and

$R'_1$ with MV) by searching neighboring blocks of $R_0$ and $R_1$. The most similar reference is then used as the prediction.

**[0080]** As shown in FIG. 5, template matching is used to derive motion information of the current block 224 in picture 226 by finding the best match between a template (top neighboring blocks 228 and/or left neighboring blocks 230 of current block 224) in current picture 226 and a block (same size to the template) in a reference picture (e.g., reference picture 232 (Ref0) or 434 (Ref1)). In FIG. 5, reference blocks that are possibilities as the best match are shows as reference block 236 ($R_1$), reference block 238 ($R'_1$), reference block 240 ($R_0$), and reference block 242 ($R'_0$).

**[0081]** At video encoder 20, the decision on whether using FRUC merge mode for a CU is based on RD cost selection as done for normal merge candidate. That is the two matching modes (bilateral matching and template matching) are both checked for a CU by using RD cost selection. The one leading to the minimal cost is further compared to other CU modes. If a FRUC matching mode is the most efficient one, FRUC flag is set to true for the CU and the related matching mode is used.

**[0082]** In the 5th JVET meeting, "Enhanced Template Matching in FRUC Mode," JVET-E0035, available at http://phenix.it-sudparis.eu/jvet/, was proposed to further improve FRUC Template matching. A flowchart of an exemplary FRUC template matching mode is shown in FIG. 6A. In the first step, a template $T_0$ (and its corresponding motion information MV0) is found to match current template Tc of current block from list0 reference pictures. In the second step, template $T_1$ (and its corresponding motion information MV1) is found from list1 reference pictures. The obtained motion information MV0 and MV1 are used to perform bi-prediction to generate predictor of the current block.

**[0083]** FIGS. 6A and 6B are flow diagrams showing example proposed modifications to frame-rate up conversion template matching mode. FRUC template matching mode may be enhanced by introducing bi-directional template matching and adaptive selection between uni-prediction and bi-prediction. Exemplary modifications relative to FIG. 6A are underlined in FIG. 6B.

**[0084]** Bi-directional template matching may be implemented based on uni-directional template matching. As shown in FIG. 6A, a matched template $T_0$ is first found in the first step of template matching from List0 reference pictures (240). Note that List0 here is only taken as an example. In fact, whether List0 or List1 used in the first step is adaptive to initial distortion cost between current template and initial template in

corresponding reference picture. The initial template can be determined with initial motion information of the current block which is available before performing the first template matching. The reference picture list corresponding to minimal initial template distortion cost will be used in the first step of template matching. For example, if initial template distortion cost corresponding to list0 is no larger than cost corresponding to List1, List0 is used in the first step of template matching and List1 is used in the second step), then, the current template $T_C$ of current block is updated as follows:

$$T'_C = 2*T_C - T_0$$

**[0085]** The updated current template $T'_C$, instead of the current template $T_C$, is used to find another matched template $T_1$ from List1 reference pictures in the second template matching (242). As a result, the matched template $T_1$ is found by jointly using List0 and List1 reference pictures (244). This matching process is called bi-directional template matching.

**[0086]** The selection between uni-prediction and bi-prediction for motion compensation prediction (MCP) may be based on template matching distortion. As shown in FIG. 6B, during template matching, distortion between template $T_0$ and Tc (the current template) can be calculated as cost0 (250), the current template may be updated (252), and distortion between template $T_1$ and $T'_C$ (the updated current template) can be calculated as cost1 (254). If cost0 is less than 0.5*cost1 (256), uni-prediction based on MV0 may be applied to FRUC template matching mode (258); otherwise, bi-prediction based on MV0 and MV1 is applied (260). Note that cost0 is compared to 0.5*cost1 since cost1 indicates a difference between template $T_1$ and $T'_C$ (the updated current template), which is 2 times of difference between Tc (the current template) and its prediction of $0.5*(T_0+T_1)$. It is noted that MCP may be applied to PU-level motion refinement. Sub-PU level motion refinement may be kept unchanged.

**[0087]** FIG. 7 shows an example of optical flow trajectory for BIO. In the example of FIG. 7, B-picture 270 is a bi-directional inter-predicted picture that is being predicted using reference picture 272 ($Ref_0$) and reference picture 274 ($Ref_1$). BIO utilizes pixel-wise motion refinement which is performed on top of block-wise motion compensation in the case of bi-prediction. As BIO compensates the fine motion inside the block, enabling BIO potentially results in enlarging the block size for motion compensation. Sample-level motion refinement does not require exhaustive search or signaling by using an explicit equation to give the fine motion vector for each sample.

**[0088]** $I^{(k)}$ represents a luminance value from reference $k$ ($k$=0, 1) after motion compensation is performed for a bi-predicted block. $\partial I^{(k)}/\partial x$ and $\partial I^{(k)}/\partial y$ are the horizontal and vertical components of the $I^{(k)}$ gradient, respectively. Assuming the optical flow is valid, the motion vector field $(v_x, v_y)$ is given by the following equation:

$$\partial I^{(k)}/\partial t + v_x \partial I^{(k)}/\partial x + v_y \partial I^{(k)}/\partial y = 0. \quad (1)$$

**[0089]** Combining the optical flow equation with Hermite interpolation for motion trajectory of each sample one gets a unique polynomial of third order which matches both function values $I^{(k)}$ and derivatives $\partial I^{(k)}/\partial x$, $\partial I^{(k)}/\partial y$ at the ends. The value of this polynomial at $t$=0 is BIO prediction:

$$pred_{BIO} = 1/2 \cdot \left(I^{(0)} + I^{(1)} + v_x/2 \cdot \left(\tau_1 \partial I^{(1)}/\partial x - \tau_0 \partial I^{(0)}/\partial x\right) + v_y/2 \cdot \left(\tau_1 \partial I^{(1)}/\partial y - \tau_0 \partial I^{(0)}/\partial y\right)\right). \quad (2)$$

**[0090]** In equation (2), $\tau_0$ and $\tau_1$ correspond to the distance to reference frames as shown is FIG. 7. Distances $\tau_0$ and $\tau_1$ are calculated based on POC values for Ref0 and Ref1: $\tau_0$=POC(current)–POC(Ref0), $\tau_1$= POC(Ref1)– POC(current). If both predictions come from the same time direction (both from the past or both from the future) then signs are different $\tau_0 \cdot \tau_1 < 0$. In this case BIO can be applied only if prediction is not from the same time moment ($\tau_0 \neq \tau_1$), both referenced regions have non-zero motion ($MVx_0, MVy_0, MVx_1, MVy_1 \neq 0$) and block motion vectors are proportional to the time distance ($MVx_0/MVx_1 = MVy_0/MVy_1 = -\tau_0/\tau_1$).

**[0091]** The motion vector field $(v_x, v_y)$ is determined by minimizing the difference $\Delta$ between values at points A and B, which corresponds to the intersection of motion trajectory and reference frame planes in FIG. 7. This intersection is shown as point 276 in FIG. 7. One model uses only the first linear term of local Taylor expansion for $\Delta$:

$$\Delta = \left(I^{(0)} - I^{(1)}_0 + v_x\left(\tau_1 \partial I^{(1)}/\partial x + \tau_0 \partial I^{(0)}/\partial x\right) + v_y\left(\tau_1 \partial I^{(1)}/\partial y + \tau_0 \partial I^{(0)}/\partial y\right)\right) \quad (3)$$

**[0092]** All values in equation (1) depend on sample location $(i', j')$, which was omitted so far. Assuming the motion is consistent in a local surrounding, the $\Delta$ inside (2M+1)×(2M+1) square window $\Omega$ centered in currently predicted point $(i, j)$ may be minimized:

$$(v_x, v_y) = \underset{v_x, v_y}{\operatorname{argmin}} \sum_{[i',j]\in\Omega} \Delta^2[i', j'] \tag{4}$$

**[0093]** For this optimization problem, a simplified solution making first minimization in vertical and then in horizontal directions may be used, which results in:

$$v_x = (s_1 + r) > m ? clip3\left(-thBIO, thBIO, -\frac{s_3}{(s_1+r)}\right) : 0 \tag{5}$$

$$v_y = (s_5 + r) > m ? clip3\left(-thBIO, thBIO, -\frac{s_6 - v_x s_2/2}{(s_5+r)}\right) : 0 \tag{6}$$

where,

$$\begin{aligned} s_1 &= \sum_{[i',j]\in\Omega} \left(\tau_1 \partial I^{(1)}/\partial x + \tau_0 \partial I^{(0)}/\partial x\right)^2; s_3 = \sum_{[i',j]\in\Omega} \left(I^{(1)} - I^{(0)}\right)\left(\tau_1 \partial I^{(1)}/\partial x + \tau_0 \partial I^{(0)}/\partial x\right); \\ s_2 &= \sum_{[i',j]\in\Omega} \left(\tau_1 \partial I^{(1)}/\partial x + \tau_0 \partial I^{(0)}/\partial x\right)\left(\tau_1 \partial I^{(1)}/\partial y + \tau_0 \partial I^{(0)}/\partial y\right); \\ s_5 &= \sum_{[i',j]\in\Omega} \left(\tau_1 \partial I^{(1)}/\partial y + \tau_0 \partial I^{(0)}/\partial y\right)^2; s_6 = \sum_{[i',j]\in\Omega} \left(I^{(1)} - I^{(0)}\right)\left(\tau_1 \partial I^{(1)}/\partial y + \tau_0 \partial I^{(0)}/\partial y\right) \end{aligned} \tag{7}$$

**[0094]** In order to avoid division by zero or very small value, regularization parameters *r* and *m* are introduced in equations (2), (3).

$$r = 500 \cdot 4^{d-8} \tag{8}$$

$$m = 700 \cdot 4^{d-8} \tag{9}$$

Here $d$ is the internal bit-depth of the input video.

**[0095]** In some cases, the MV refinement of BIO might be unreliable due to noise or irregular motion. Therefore, in BIO, the magnitude of MV regiment is clipped to the certain threshold thBIO. The threshold value is determined based on whether all the reference pictures of the current picture are all from one direction. If all the reference pictures of the current pictures of the current picture are from one direction, the value of the threshold is set to $12 \times 2^{14-d}$, otherwise, it is set to $12 \times 2^{13-d}$.

**[0096]** Gradients for BIO can be calculated at the same time with motion compensation interpolation using operations consistent with HEVC motion compensation process (2D separable FIR). The input for this 2D separable FIR is the same reference frame sample as for motion compensation process and fractional position (*fracX*, *fracY*) according to the fractional part of block motion vector. For horizontal gradient $\partial I/\partial x$, the signal is first interpolated vertically using the *BIOfilterS* corresponding to the fractional position *fracY* with de-scaling shift *d*-8, and then gradient filter *BIOfilterG* is applied in a

horizontal direction corresponding to the fractional position *fracX* with a de-scaling shift by 18-*d*. For vertical gradient $\partial I/\partial y$, the gradient filter is first applied vertically using the *BIOfilterG* corresponding to the fractional position *fracY* with de-scaling shift *d*-8, and then signal displacement is performed using *BIOfilterS* in a horizontal direction corresponding to the fractional position *fracX* with de-scaling shift by 18-*d*. The length of interpolation filter for gradients calculation BIOfilterG and signal displacement BIOfilterF may be shorter (6-tap) in order to maintain reasonable complexity. Table 1 shows the filters that can be used for gradients calculation for different fractional positions of block motion vector in BIO. Table 2 shows the interpolation filters that can be used for prediction signal generation in BIO.

**[0097]** FIG. 8 shows an example of the gradient calculation for an 8x4 block (shown as current block 280 in FIG. 8). For the 8x4 block, a video coder fetches the motion compensated predictors (also referred to as MC predictors) and calculates the HOR/VER gradients of the pixels within current block 280 as well as the outer two lines of pixels because solving vx and vy for each pixel uses the HOR/VER gradient values and motion compensated predictors of the pixels within the window Ω centered in each pixel, as shown in equation (4). In JEM, for example, the size of this window is set to 5x5, meaning a video coder fetches the motion compensated predictors and calculates the gradients for the outer two lines of pixels. Window 282 represents the 5x5 window centered at pixel A, and window 284 represents the 5x5 window centered at pixel B.

**Table 1: Filters for gradients calculation in BIO**

| Fractional pel position | Interpolation filter for gradient(BIOfilterG) |
|---|---|
| 0 | { 8, -39, -3, 46, -17, 5} |
| 1/16 | { 8, -32, -13, 50, -18, 5} |
| 1/8 | { 7, -27, -20, 54, -19, 5} |
| 3/16 | { 6, -21, -29, 57, -18, 5} |
| ¼ | { 4, -17, -36, 60, -15, 4} |
| 5/16 | { 3, -9, -44, 61, -15, 4} |
| 3/8 | { 1, -4, -48, 61, -13, 3} |
| 7/16 | { 0, 1, -54, 60, -9, 2} |
| 1/2 | { 1, 4, -57, 57, -4, 1} |

**Table 2: Interpolation filters for prediction signal generation in BIO**

| Fractional pel position | Interpolation filter for prediction signal(BIOfilterS) |
|---|---|
| 0 | { 0, 0, 64, 0, 0, 0} |
| 1/16 | { 1, -3, 64, 4, -2, 0} |
| 1/8 | { 1, -6, 62, 9, -3, 1} |
| 3/16 | { 2, -8, 60, 14, -5, 1} |
| 1/4 | { 2, -9, 57, 19, -7, 2} |
| 5/16 | { 3, -10, 53, 24, -8, 2} |
| 3/8 | { 3, -11, 50, 29, -9, 2} |
| 7/16 | { 3, -11, 44, 35, -10, 3} |
| 1/2 | { 1, -7, 38, 38, -7, 1} |

**[0098]** In JEM, BIO is applied to all bi-directional predicted blocks when the two predictions are from different reference pictures. When LIC is enabled for a CU, BIO is disabled.

**[0099]** FIG. 9 is a conceptual diagram showing an example of proposed decoder-side motion vector derivation based on bilateral template matching. JEM also includes a coding tool referred to as template matching. Video decoder 30 generates a bilateral template as the weighted combination of the two prediction blocks, from the initial MV0 of list0 and MV1 of list1 respectively, as shown in FIG. 9.

**[0100]** The template matching operation consists of calculating cost measures between the generated template and the sample region (around the initial prediction block) in the reference picture. For each of the two reference pictures, the MV that yields the minimum template cost is considered as the updated MV of that list to replace the original one. Finally, the two new MVs, i.e., MV0' and MV1' as shown in Fig. 8, are used for regular bi-prediction. As it is commonly used in block-matching motion estimation, the sum of absolute differences (SAD) is utilized as cost measure.

**[0101]** The proposed DMVD techniques are applied for merge mode of bi-prediction with one from the reference picture in the past and the other from reference picture in the future, without the transmission of additional syntax element.

**[0102]** In JEM4.0, when LIC, affine, sub-CU merge candidate or FRUC is selected for one CU, the DMVD is not applied

**[0103]** FIGS. 10A and 10B are conceptual diagrams showing an example illustration of sub-blocks where overlapped block motion compensation (OBMC) may apply, such as the OBMC included in JEM. FIG. 10A shows inter CU 300, which includes 4x4 sub-blocks. For current sub-block 302, MVs of left neighboring sub-block 304 and above neighboring sub-block 306 are used in performing OBMC for current sub-block 302.

FIG. 10B shows inter CU 310, which includes 4x4 sub-blocks. For current sub-block 312, MVs of left neighboring sub-block 314, above neighboring sub-block 316, below neighboring sub-block 318, and right neighboring sub-block 320 are used in performing OBMC for current sub-block 312.

**[0104]** OBMC has been used for early generations of video standards, e.g., as in H.263. In JEM, the OBMC is performed for all Motion Compensated (MC) block boundaries except the right and bottom boundaries of a CU. Moreover, it is applied for both luma and chroma components. In JEM, a MC block is corresponding to a coding block. When a CU is coded with sub-CU mode (includes sub-CU merge, Affine and FRUC mode, as described in Kaiming He, Jian Sun, and Xiaoou Tang, "Guided image filtering," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 35, no. 6, pp. 1397-1409, 2013), each sub-block of the CU is a MC block. To process CU boundaries in a uniform fashion, OBMC is performed at sub-block level for all MC block boundaries, where sub-block size is set equal to 4x4, as illustrated in FIGS. 10A and 10B.

**[0105]** When OBMC applies to the current sub-block, besides current motion vectors, motion vectors of four connected neighboring sub-blocks, if available and are not identical to the current motion vector, are also used to derive prediction block for the current sub-block. These multiple prediction blocks based on multiple motion vectors are combined to generate the final prediction signal of the current sub-block.

**[0106]** FIGS. 11A-11D illustrate a process for determining a predictive block for current sub-block 312. In the example of FIG. 11A, the OBMC prediction of current sub-block 312 equals a weighted average of the predictive sub-block determined using the MV of above-neighboring block 324 and the predictive sub-block determined for the current sub-block using the MV of the current sub-block. In the example of FIG. 11B, the OBMC prediction of current sub-block 312 equals a weighted average of the predictive sub-block determined using the MV of left-neighboring block 326 and the predictive sub-block determined for the current sub-block using the MV of the current sub-block. In the example of FIG. 11C, the OBMC prediction of current sub-block 312 equals a weighted average of the predictive sub-block determined using the MV of below-neighboring block 328 and the predictive sub-block determined for the current sub-block using the MV of the current sub-block. In the example of FIG. 11D, the OBMC prediction of current sub-block 312 equals a weighted average of the predictive sub-block determined using the MV of right-neighboring block 330 and the predictive sub-block determined for the current sub-block using the MV of the current sub-block.

**[0107]** In JEM, for a CU with size less than or equal to 256 luma samples, a CU level flag is signaled to indicate whether OBMC is applied or not for the current CU. For the CUs with size larger than 256 luma samples or not coded with AMVP mode, OBMC is applied by default. At encoder, when OBMC is applied for a CU, its impact is taken into account during motion estimation stage. The prediction signal by using motion information of the top neighboring block and the left neighboring block is used to compensate the top and left boundaries of the original signal of the current CU, and then the normal motion estimation process is applied.

**[0108]** The techniques of this disclosure may address some or all of the issues introduced above. All of the DMVD-related techniques described herein (e.g., FRUC Bilateral Matching, FRUC Template Matching, Bilateral Template matching and so on) potentially provide significant bit-rate reductions. However, the MVs refined by the DMVD methods can be used as the MVP for the following block and it is not friendly for hardware implementation (e.g., the hardware implementation may be undesirably complex). For example, FRUC Bilateral Matching or Template Matching require a decoder to perform decoder-side motion search to decide the motion vector, which may cause serious dependency issues due to the reconstruction of motion vectors depending on pixel-domain processing at the decoder side. It is common practice that asynchronous architecture is exploited in practical hardware design for video codecs so that the symbol and pixel throughput can be maximized. Furthermore, the pre-fetch of reference samples is also important, as the cache miss when fetching reference samples for interpolation greatly reduces the pixel throughput. It is potentially desirable to know the exact value of motion vectors during parsing stage for efficient hardware design. Unfortunately, decoder-side motion refinement/search prohibits the use of the aforementioned techniques, because the exact value of motion vectors cannot be determined upfront (e.g., in the parsing stage).

**[0109]** This disclosure describes techniques related to limitations on MVPs. The techniques of this disclosure include techniques related to disallowing the MVs which are derived or refined by the decoder-side MV refinement approaches to be used as the MVPs for the following coding blocks. The derived or refined MVs are used only for deriving the predictors for current block. Alternative MVs may be stored for a current block and used instead as MVPs for the following blocks. The alternative MVs may, for example, be the signaled MV before the DMVD refinement or the first available spatial MV according to a pre-defined order of derivation, or a MV determined any way

without decoder-side MV search. The decoder-side MV refinement approaches include FRUC Template Matching, FRUC Bilateral Matching, and Bilateral Template Matching as used in JEM.

**[0110]** According to one example, if a block is coded as Merge mode and the merge index is received to indicate which neighboring MV is used. Assuming the signaled Merge MV is refined by the Bilateral Template Matching, then the refined MV is not stored as the MVPs for the following blocks. Instead, the MV before the refinement is stored as the MVPs for the following blocks.

**[0111]** According to another example, if a block is coded as FRUC Template Matching Merge mode as in JEM. The final MVs derived by the Template Matching is not stored as the MVPs for the following blocks. Instead, a pre-defined MV is stored as the MVPs for the following blocks. The pre-defined MV could be the first available initial MV seeding, first available spatial MV according to a pre-defined order of derivation, or any way to determine a MV without decoder-side MV search.

**[0112]** According to another example, if a block is coded as inter mode and the selected MVP is derived by Template Matching as JEM. The (MVD+refined MVP) is used to fetch the predictors for current block. But (MVD+refined MVP) is not stored as the MVPs for the following blocks. Instead, (MVD+ originally signaled MVP) is stored as the MVPs for the following blocks.

**[0113]** In other examples, instead of the using initial seeding or original signaled MVP before refinement for MVP derivation, non-adjacent motion vectors located at non-immediate positions can also be used to derived or extrapolate the MVP value of the current block. For example, a linear model which fits (x-1, y) and (x-N, y) can be used to estimate the x-direction MVP value of the current block. Similarly, a linear mode which fits (x, y-1) and (x, y-N) can be used to estimate the y-direction MVP value of the current block. This is to enhance the quality of the MVP when motion search is unavailable at decoder side.

**[0114]** This disclosure also describes other potential limitations on MVPs. In some examples, the limitation is only applied to the blocks which use DMVD methods utilizing reconstructed pixels such as the Template Matching methods in JEM. In some examples, the MVs derived by DMVD methods are not used as MVPs for the following blocks within the same picture. In other words, the MVs derived by the DMVD methods can be used as MVPs for the blocks in the following pictures. In some examples, the

MVs derived by DMVD methods are not used as MVPs for the following blocks but can be used as the MVs to perform OBMC in the following blocks.

**[0115]** FIG. 12 is a block diagram illustrating an example video encoder 20 that may implement the techniques described in this disclosure. Video encoder 20 may perform intra- and inter-coding of video blocks within video slices. Intra-coding relies on spatial prediction to reduce or remove spatial redundancy in video within a given video frame or picture. Inter-coding relies on temporal prediction to reduce or remove temporal redundancy in video within adjacent frames or pictures of a video sequence. Intra-mode (I mode) may refer to any of several spatial based compression modes. Inter-modes, such as uni-directional prediction (P mode) or bi-prediction (B mode), may refer to any of several temporal-based compression modes.

**[0116]** In the example of FIG. 12, video encoder 20 includes a video data memory 33, partitioning unit 35, prediction processing unit 41, summer 50, transform processing unit 52, quantization unit 54, entropy encoding unit 56. Prediction processing unit 41 includes motion estimation unit (MEU) 42, motion compensation unit (MCU) 44, and intra prediction unit 46. For video block reconstruction, video encoder 20 also includes inverse quantization unit 58, inverse transform processing unit 60, summer 62, filter unit 64, and decoded picture buffer (DPB) 66.

**[0117]** As shown in FIG. 12, video encoder 20 receives video data and stores the received video data in video data memory 33. Video data memory 33 may store video data to be encoded by the components of video encoder 20. The video data stored in video data memory 33 may be obtained, for example, from video source 18. DPB 66 may be a reference picture memory that stores reference video data for use in encoding video data by video encoder 20, e.g., in intra- or inter-coding modes. Video data memory 33 and DPB 66 may be formed by any of a variety of memory devices, such as dynamic random access memory (DRAM), including synchronous DRAM (SDRAM), magnetoresistive RAM (MRAM), resistive RAM (RRAM), or other types of memory devices. Video data memory 33 and DPB 66 may be provided by the same memory device or separate memory devices. In various examples, video data memory 33 may be on-chip with other components of video encoder 20, or off-chip relative to those components.

**[0118]** Partitioning unit 35 retrieves the video data from video data memory 33 and partitions the video data into video blocks. This partitioning may also include partitioning into slices, tiles, or other larger units, as wells as video block partitioning,

e.g., according to a quadtree structure of LCUs and CUs. Video encoder 20 generally illustrates the components that encode video blocks within a video slice to be encoded. The slice may be divided into multiple video blocks (and possibly into sets of video blocks referred to as tiles). Prediction processing unit 41 may select one of a plurality of possible coding modes, such as one of a plurality of intra coding modes or one of a plurality of inter coding modes, for the current video block based on error results (e.g., coding rate and the level of distortion). Prediction processing unit 41 may provide the resulting intra- or inter-coded block to summer 50 to generate residual block data and to summer 62 to reconstruct the encoded block for use as a reference picture.

**[0119]** Intra prediction unit 46 within prediction processing unit 41 may perform intra-predictive coding of the current video block relative to one or more neighboring blocks in the same frame or slice as the current block to be coded to provide spatial compression. Motion estimation unit 42 and motion compensation unit 44 within prediction processing unit 41 perform inter-predictive coding of the current video block relative to one or more predictive blocks in one or more reference pictures to provide temporal compression. Motion estimation unit 42 and motion compensation unit 44 may be configured to perform DMVD in accordance with the techniques described in this disclosure.

**[0120]** Motion estimation unit 42 may be configured to determine the inter-prediction mode for a video slice according to a predetermined pattern for a video sequence. The predetermined pattern may designate video slices in the sequence as P slices or B slices. Motion estimation unit 42 and motion compensation unit 44 may be highly integrated, but are illustrated separately for conceptual purposes. Motion estimation, performed by motion estimation unit 42, is the process of generating motion vectors, which estimate motion for video blocks. A motion vector, for example, may indicate the displacement of a PU of a video block within a current video frame or picture relative to a predictive block within a reference picture.

**[0121]** A predictive block is a block that is found to closely match the PU of the video block to be coded in terms of pixel difference, which may be determined by sum of absolute difference (SAD), sum of square difference (SSD), or other difference metrics. In some examples, video encoder 20 may calculate values for sub-integer pixel positions of reference pictures stored in DPB 66. For example, video encoder 20 may interpolate values of one-quarter pixel positions, one-eighth pixel positions, or other fractional pixel positions of the reference picture. Therefore, motion estimation unit 42 may

perform a motion search relative to the full pixel positions and fractional pixel positions and output a motion vector with fractional pixel precision.

**[0122]** Motion estimation unit 42 calculates a motion vector for a PU of a video block in an inter-coded slice by comparing the position of the PU to the position of a predictive block of a reference picture. The reference picture may be selected from a first reference picture list (List 0) or a second reference picture list (List 1), each of which identify one or more reference pictures stored in DPB 66. Motion estimation unit 42 sends the calculated motion vector to entropy encoding unit 56 and motion compensation unit 44.

**[0123]** Motion compensation, performed by motion compensation unit 44, may involve fetching or generating the predictive block based on the motion vector determined by motion estimation, possibly performing interpolations to sub-pixel precision. Upon receiving the motion vector for the PU of the current video block, motion compensation unit 44 may locate the predictive block to which the motion vector points in one of the reference picture lists. Video encoder 20 forms a residual video block by subtracting pixel values of the predictive block from the pixel values of the current video block being coded, forming pixel difference values. The pixel difference values form residual data for the block, and may include both luma and chroma difference components. Summer 50 represents the component or components that perform this subtraction operation. Motion compensation unit 44 may also generate syntax elements associated with the video blocks and the video slice for use by video decoder 30 in decoding the video blocks of the video slice.

**[0124]** After prediction processing unit 41 generates the predictive block for the current video block, either via intra prediction or inter prediction, video encoder 20 forms a residual video block by subtracting the predictive block from the current video block. The residual video data in the residual block may be included in one or more TUs and applied to transform processing unit 52. Transform processing unit 52 transforms the residual video data into residual transform coefficients using a transform, such as a discrete cosine transform (DCT) or a conceptually similar transform. Transform processing unit 52 may convert the residual video data from a pixel domain to a transform domain, such as a frequency domain.

**[0125]** Transform processing unit 52 may send the resulting transform coefficients to quantization unit 54. Quantization unit 54 quantizes the transform coefficients to further reduce bit rate. The quantization process may reduce the bit depth associated

with some or all of the coefficients. The degree of quantization may be modified by adjusting a quantization parameter. In some examples, quantization unit 54 may then perform a scan of the matrix including the quantized transform coefficients. In another example, entropy encoding unit 56 may perform the scan.

**[0126]** Following quantization, entropy encoding unit 56 entropy encodes the quantized transform coefficients. For example, entropy encoding unit 56 may perform context adaptive variable length coding (CAVLC), context adaptive binary arithmetic coding (CABAC), syntax-based context-adaptive binary arithmetic coding (SBAC), probability interval partitioning entropy (PIPE) coding or another entropy encoding methodology or technique. Following the entropy encoding by entropy encoding unit 56, the encoded bitstream may be transmitted to video decoder 30, or archived for later transmission or retrieval by video decoder 30. Entropy encoding unit 56 may also entropy encode the motion vectors and the other syntax elements for the current video slice being coded.

**[0127]** Inverse quantization unit 58 and inverse transform processing unit 60 apply inverse quantization and inverse transformation, respectively, to reconstruct the residual block in the pixel domain for later use as a reference block of a reference picture. Motion compensation unit 44 may calculate a reference block by adding the residual block to a predictive block of one of the reference pictures within one of the reference picture lists. Motion compensation unit 44 may also apply one or more interpolation filters to the reconstructed residual block to calculate sub-integer pixel values for use in motion estimation. Summer 62 adds the reconstructed residual block to the motion compensated prediction block produced by motion compensation unit 44 to produce a reconstructed block.

**[0128]** Filter unit 64 filters the reconstructed block (e.g. the output of summer 62) and stores the filtered reconstructed block in DPB 66 for uses as a reference block. The reference block may be used by motion estimation unit 42 and motion compensation unit 44 as a reference block to inter-predict a block in a subsequent video frame or picture. Filter unit 64 is intended to represent any or any combination of a deblock filter, a sample adaptive offset (SAO) filter, and adaptive loop filter (ALF), or any other type of loop filters. A deblock filter may, for example, apply deblocking filtering to filter block boundaries to remove blockiness artifacts from reconstructed video. An SAO filter may apply offsets to reconstructed pixel values in order to improve overall coding quality. Additional loop filters (in loop or post loop) may also be used.

**[0129]** FIG. 13 is a block diagram illustrating an example video decoder 30 that may implement the techniques described in this disclosure. Video decoder 30 of FIG. 13 may, for example, be configured to receive the signaling described above with respect to video encoder 20 of FIG. 12. In the example of FIG. 13, video decoder 30 includes video data memory 78, entropy decoding unit 80, prediction processing unit 81, inverse quantization unit 86, inverse transform processing unit 88, summer 90, filter unit 92, and DPB 94. Prediction processing unit 81 includes motion compensation unit 82 and intra prediction unit 84. Video decoder 30 may, in some examples, perform a decoding pass generally reciprocal to the encoding pass described with respect to video encoder 20 from FIG. 12.

**[0130]** During the decoding process, video decoder 30 receives an encoded video bitstream that represents video blocks of an encoded video slice and associated syntax elements from video encoder 20. Video decoder 20 stores the received encoded video bitstream in video data memory 78. Video data memory 78 may store video data, such as an encoded video bitstream, to be decoded by the components of video decoder 30. The video data stored in video data memory 78 may be obtained, for example, via link 16, from storage device 26, or from a local video source, such as a camera, or by accessing physical data storage media. Video data memory 78 may form a coded picture buffer (CPB) that stores encoded video data from an encoded video bitstream. DPB 94 may be a reference picture memory that stores reference video data for use in decoding video data by video decoder 30, e.g., in intra- or inter-coding modes. Video data memory 78 and DPB 94 may be formed by any of a variety of memory devices, such as DRAM, SDRAM, MRAM, RRAM, or other types of memory devices. Video data memory 78 and DPB 94 may be provided by the same memory device or separate memory devices. In various examples, video data memory 78 may be on-chip with other components of video decoder 30, or off-chip relative to those components.

**[0131]** Entropy decoding unit 80 of video decoder 30 entropy decodes the video data stored in video data memory 78 to generate quantized coefficients, motion vectors, and other syntax elements. Entropy decoding unit 80 forwards the motion vectors and other syntax elements to prediction processing unit 81. Video decoder 30 may receive the syntax elements at the video slice level and/or the video block level.

**[0132]** When the video slice is coded as an intra-coded (I) slice, intra prediction unit 84 of prediction processing unit 81 may generate prediction data for a video block of the

current video slice based on a signaled intra prediction mode and data from previously decoded blocks of the current frame or picture. When the video frame is coded as an inter-coded slice (e.g., B slice or P slice), motion compensation unit 82 of prediction processing unit 81 produces predictive blocks for a video block of the current video slice based on the motion vectors and other syntax elements received from entropy decoding unit 80. The predictive blocks may be produced from one of the reference pictures within one of the reference picture lists. Video decoder 30 may construct the reference frame lists, List 0 and List 1, using default construction techniques based on reference pictures stored in DPB 94.

**[0133]** Motion compensation unit 82, in conjunction with other parts of video decoder 30, may be configured to perform DMVD in accordance with the techniques described in this disclosure. Motion compensation unit 82 determines prediction information for a video block of the current video slice by parsing the motion vectors and other syntax elements, and uses the prediction information to produce the predictive blocks for the current video block being decoded. For example, motion compensation unit 82 uses some of the received syntax elements to determine a prediction mode (e.g., intra- or inter-prediction) used to code the video blocks of the video slice, an inter-prediction slice type (e.g., B slice or P slice), construction information for one or more of the reference picture lists for the slice, motion vectors for each inter-encoded video block of the slice, inter-prediction status for each inter-coded video block of the slice, and other information to decode the video blocks in the current video slice.

**[0134]** Motion compensation unit 82 may also perform interpolation based on interpolation filters. Motion compensation unit 82 may use interpolation filters as used by video encoder 20 during encoding of the video blocks to calculate interpolated values for sub-integer pixels of reference blocks. In this case, motion compensation unit 82 may determine the interpolation filters used by video encoder 20 from the received syntax elements and use the interpolation filters to produce predictive blocks.

**[0135]** Inverse quantization unit 86 inverse quantizes, i.e., de-quantizes, the quantized transform coefficients provided in the bitstream and decoded by entropy decoding unit 80. The inverse quantization process may include use of a quantization parameter calculated by video encoder 20 for each video block in the video slice to determine a degree of quantization and, likewise, a degree of inverse quantization that should be applied. Inverse transform processing unit 88 applies an inverse transform, e.g., an inverse DCT, an inverse integer transform, or a conceptually similar inverse transform

process, to the transform coefficients in order to produce residual blocks in the pixel domain.

**[0136]** After prediction processing unit generates the predictive block for the current video block using, for example, intra or inter prediction, video decoder 30 forms a reconstructed video block by summing the residual blocks from inverse transform processing unit 88 with the corresponding predictive blocks generated by motion compensation unit 82. Summer 90 represents the component or components that perform this summation operation.

**[0137]** Filter unit 92 represents any or any combination of a deblocking filter, an SAO filter, and ALF, or any other type of loop filter (either in the coding loop or after the coding loop). The decoded video blocks in a given frame or picture are then stored in DPB 94, which stores reference pictures used for subsequent motion compensation. DPB 94 may be part of or separate from additional memory that stores decoded video for later presentation on a display device, such as display device 32 of FIG. 1.

**[0138]** Video decoder 30 represents an example of a video decoder configured to determine a block of video data is coded in an inter prediction mode; determine motion information for the block of video data according to any of the various techniques or combination of techniques described in this disclosure; locating a reference block in a reference picture using the motion information; and generate a predictive block for the block of video data based on the reference block.

**[0139]** The motion information may include some or all of a motion vector, a motion vector precision, and a reference picture index. Video decoder 30 may be configured to determine that a DMVD mode is enabled for the block of video data and process the reference block in accordance with the DMVD mode. The DMVD mode may be any one of FRUC Template Matching, FRUC Bilateral Matching, and Bilateral Template Matching. Video decoder 30 may also be configured to add residual data to the predictive block to generate a reconstructed block of video data; process the reconstructed block of video data to generate a decoded block of video data; and output the decoded block of video data. To output the decoded block of video data, video decoder may store a picture including the decoded block of video data in a decoded picture buffer for use as reference picture in decoding subsequent pictures of video data and/or output the picture including the decoded block of video data to a display device.

**[0140]** Video decoder 30 also represents an example of a video decoder configured to determine a first block of video data is coded in an inter prediction mode; determine that

a decoder-side motion vector derivation (DMVD) mode is enabled for the first block of video data; determine a motion vector for the first block of video data; refine the motion vector in accordance with the DMVD mode to determine a refined motion vector; use the refined motion vector, generating a predictive block for the first block; and use the motion vector for the first block of video data as an MVP for the second block. The motion vector may, for example, have a first precision, and the refined motion vector has a second precision that is different than the first precision, and wherein using the motion vector for the first block of video data as the MVP comprises using the motion vector with the first precision.

**[0141]** Video decoder 30 also represents an example of a video decoder configured to determine a first block of video data is coded in an inter prediction mode; determine that a decoder-side motion vector derivation (DMVD) mode is enabled for the first block of video data; determine a motion vector for the first block of video data; refine the motion vector in accordance with the DMVD mode to determine a refined motion vector; use the refined motion vector, generating a predictive block for the first block; and use a predefined motion vector as a motion vector predictor (MVP) for the second block.

**[0142]** Video decoder 30 also represents an example of a video decoder configured to determine a first block of video data is coded in an inter prediction mode; determine that a DMVD mode is enabled for the first block of video data; determine a motion vector for the first block of video data; refine the motion vector in accordance with the DMVD mode to determine a refined motion vector; use the refined motion vector, generating a predictive block for the first block; and use a linear model to determine a motion vector predictor (MVP) for the second block based on the motion vector for the first block.

**[0143]** FIG. 14 is a flow diagram illustrating an example video decoding technique described in this disclosure. The techniques of FIG. 14 will be described with reference to a generic video decoder, such as but not limited to video decoder 30. In some instances, the techniques of FIG. 14 may be performed by a video encoder such as video encoder 20, in which case the generic video decoder corresponds to the decoding loop or other decoding functionality of the video encoder.

**[0144]** In the example of FIG. 14, the video decoder determine a first block of video data is coded in an inter prediction mode (350). The video decoder determines a first motion vector for the first block of video data (352). The video decoder performs motion vector refinement on the first motion vector for the first block to determine a refined motion vector for the first block of video data (354). To perform the motion

vector refinement on the first motion vector for the first, the video decoder may determine that a DMVD mode is enabled for the first block of video data and perform the motion vector refinement on the first motion vector for the first block in accordance with the DMVD mode. The DMVD mode may, for example, be one of FRUC template matching, FRUC bilateral matching, or bilateral template matching.

**[0145]** The video decoder locates a first reference block in a first reference picture using the refined motion vector (356). The video decoder generates a first predictive block for the first block of video data based on the first reference block (358).

**[0146]** In response to determining that a second block of video data is coded in a mode that utilizes a motion vector associated with the first block as a motion vector predictor and in response to performing the motion vector refinement on the first motion vector for the first block, the video decoder uses a different motion vector as the motion vector predictor associated with the first block (360). For example, if the second block is coded in a merge mode, then the motion vectors of spatial or temporal neighboring blocks may be used to generate a candidate list for the second block. If one of those neighboring block of the second block is coded using a refined motion vector, however, then instead of adding the refined motion vector to the candidate list, a different motion vector may be added to the candidate list instead.

**[0147]** The mode that utilizes the motion vector associated with the first block as a motion vector predictor may, for example, be a merge mode, where motion vectors of previously coded blocks are used to generate a candidate list of motion information for a current block or an AMVP mode, where the motion vectors of previously coded blocks are used as motion vector predictors for the motion vector of a current block. The different motion vector may, for example, be the first motion vector for the first block or may be a default motion vector.

**[0148]** In some examples, the video decoder may determine the first motion vector for the first block of video data by determining a first motion vector prediction and a first motion vector difference and perform motion vector refinement on the first motion vector for the first block by refining the first motion vector predictor to determine a first refined motion vector predictor. The video decoder may set the refined motion vector for the first block of video data equal to the first refined motion vector predictor plus the first motion vector difference and set the motion vector predictor equal to the first motion vector predictor plus the first motion vector difference.

**[0149]** Based on the first motion vector, the video decoder determines a second motion vector for the second block (362). The video decoder may, for example, determine that the second block is coded in a merge mode or AMVP mode and generate a candidate list for the second block. To generate the candidate list for the second block, the video decoder may add the first motion vector or a default motion vector to the candidate list. Thus, the video decoder may determine a motion vector candidate list for the second block of video data that includes a candidate motion vector associated with the first block that is not the first refined motion vector. The video decoder locates a second reference block in a second reference picture using the second motion vector (364).

**[0150]** The video decoder may add first residual data to the first predictive block to generate a first reconstructed block of video data and process the first reconstructed block of video data to generate a first decoded block of video data. Using the second motion vector, the video decoder may locate a second reference block in a second reference picture and generate a second predictive block for the second block of video data based on the second reference block. The video decoder may add second residual data to the second predictive block to generate a second reconstructed block of video data and process the second reconstructed block of video data to generate a second decoded block of video data. To process the first and second reconstructed blocks, the video decoder may, for example, performing filtering or other types of processing.

**[0151]** The video decoder may output a picture that includes the first decoded block of video data and the second decoded block of video data. To output the picture, the video decoder may store a copy of the picture in a decoded picture buffer for use as reference picture in encoding or decoding subsequent pictures of video data. The video decoder may additionally output the picture to a display device.

**[0152]** 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, 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.

**[0153]** By way of example, and not limitation, such computer-readable storage media can comprise RAM, ROM, EEPROM, CD-ROM or other optical disk storage, magnetic disk storage, or other magnetic storage devices, flash memory, or any other medium that can be used to store desired program code in the form of instructions or data structures and that can be accessed by a computer. Also, any connection is properly termed a computer-readable medium. For example, if instructions are transmitted from a website, server, or other remote source using a coaxial cable, fiber optic cable, twisted pair, digital subscriber line (DSL), or wireless technologies such as infrared, radio, and microwave, then the coaxial cable, fiber optic cable, twisted pair, DSL, or wireless technologies such as infrared, radio, and microwave are included in the definition of medium. It should be understood, however, that computer-readable storage media and data storage media do not include connections, carrier waves, signals, or other transient media, but are instead directed to non-transient, 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.

**[0154]** Instructions may be executed by one or more processors, such as one or more digital signal processors (DSPs), general purpose microprocessors, application specific integrated circuits (ASICs), field programmable logic arrays (FPGAs), or other equivalent integrated or discrete logic circuitry. Accordingly, the term "processor," as used herein may refer to any of the foregoing structure or any other structure suitable for implementation of the techniques described herein. In addition, in some aspects, the functionality described herein may be provided within dedicated hardware and/or software modules configured for encoding and decoding, or incorporated in a combined codec. Also, the techniques could be fully implemented in one or more circuits or logic elements.

**[0155]** 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.

**[0156]** Various examples have been described. These and other examples are within the scope of the following claims.

## CLAIMS

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

determining a first block of video data is coded in an inter prediction mode;

determining a first motion vector for the first block of video data;

performing motion vector refinement on the first motion vector for the first block to determine a refined motion vector for the first block of video data;

locating a first reference block in a first reference picture using the refined motion vector;

generating a first predictive block for the first block of video data based on the first reference block;

determining that a second block of video data is coded in a mode that utilizes a motion vector associated with the first block as a motion vector predictor;

in response to determining that the second block of video data is coded in the mode that utilizes the motion vector associated with the first block as a motion vector predictor and in response to performing the motion vector refinement on the first motion vector for the first block, using a different motion vector than the first refined motion vector as the motion vector predictor associated with the first block;

based on the different motion vector, determining a second motion vector for the second block;

locating a second reference block in a second reference picture using the second motion vector; and

decoding a picture of video data based on the first reference block and the second reference block.

2. The method of claim 1, wherein performing the motion vector refinement on the first motion vector for the first block comprises:

determining that a decoder-side motion vector derivation (DMVD) mode is enabled for the first block of video data; and

performing the motion vector refinement on the first motion vector for the first block in accordance with the DMVD mode.

3. The method of claim 2, wherein the DMVD mode comprises one of frame-rate up conversion (FRUC) template matching, FRUC bilateral matching, and bilateral template matching.

4. The method of claim 1, wherein the different motion vector comprises the first motion vector for the first block.

5. The method of claim 1, wherein the different motion vector comprises a default motion vector.

6. The method of claim 1, further comprising:

determining that the second block is coded in a merge mode;

generating a candidate list for the second block, wherein generating the candidate list for the second block comprises, in response to determining that the first block is coded using bilateral template matching, adding the different motion vector to the candidate list.

7. The method of claim 1, wherein:

determining the first motion vector for the first block of video data comprises determining a first motion vector prediction and a first motion vector difference;

performing motion vector refinement on the first motion vector for the first block comprises refining the first motion vector predictor to determine a first refined motion vector predictor;

setting the refined motion vector for the first block of video data equal to the first refined motion vector predictor plus the first motion vector difference; and

setting the motion vector predictor equal to the first motion vector predictor plus the first motion vector difference.

8. The method of claim 1, further comprising:

adding first residual data to the first predictive block to generate a first reconstructed block of video data;

processing the first reconstructed block of video data to generate a first decoded block of video data;

using the second motion vector, locating a second reference block in a second reference picture;

generating a second predictive block for the second block of video data based on the second reference block;

adding second residual data to the second predictive block to generate a second reconstructed block of video data;

processing the second reconstructed block of video data to generate a second decoded block of video data; and

outputting a picture comprises the first decoded block of video data and the second decoded block of video data.

9. The method of claim 8, wherein outputting the picture comprises one or both of:

storing a copy of the picture in a decoded picture buffer for use as reference picture in decoding subsequent pictures of video data; and

outputting the picture to a display device.

10. The method of claim 1, wherein the method of decoding is performed as part of a reconstruction loop of a video encoding process.

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

a memory configured to store video data;

one or more processors configured to:

determine a first block of the video data is coded in an inter prediction mode;

determine a first motion vector for the first block of the video data;

perform motion vector refinement on the first motion vector for the first block to determine a refined motion vector for the first block of the video data;

locate a first reference block in a first reference picture using the refined motion vector;

generate a first predictive block for the first block of the video data based on the first reference block;

determine that a second block of the video data is coded in a mode that utilizes a motion vector associated with the first block as a motion vector predictor;

in response to determining that the second block of the video data is coded in the mode that utilizes the motion vector associated with the first block as a motion vector

predictor and in response to performing the motion vector refinement on the first motion vector for the first block, use a different motion vector than the first refined motion vector as the motion vector predictor associated with the first block;

based on the different motion vector, determine a second motion vector for the second block; and

locate a second reference block in a second reference picture using the second motion vector; and

decode a picture of the video data based on the first reference block and the second reference block.

12. The device of claim 11, wherein to perform the motion vector refinement on the first motion vector for the first block, the one or more processors are further configured to:

determine that a decoder-side motion vector derivation (DMVD) mode is enabled for the first block of the video data; and

perform the motion vector refinement on the first motion vector for the first block in accordance with the DMVD mode.

13. The device of claim 12, wherein the DMVD mode comprises one of frame-rate up conversion (FRUC) template matching, FRUC bilateral matching, and bilateral template matching.

14. The device of claim 11, wherein the different motion vector comprises the first motion vector for the first block.

15. The device of claim 11, wherein the different motion vector comprises a default motion vector.

16. The device of claim 11, wherein the one or more processors are further configured to:

determine that the second block is coded in a merge mode;

generate a candidate list for the second block, wherein generating the candidate list for the second block comprises, in response to determining that the first block is coded using bilateral template matching, adding the different motion vector to the candidate list.

17. The device of claim 11, wherein the one or more processors are further configured to:

determine the first motion vector for the first block of the video data comprises determining a first motion vector prediction and a first motion vector difference;

perform motion vector refinement on the first motion vector for the first block comprises refining the first motion vector predictor to determine a first refined motion vector predictor;

set the refined motion vector for the first block of the video data equal to the first refined motion vector predictor plus the first motion vector difference; and

set the motion vector predictor equal to the first motion vector predictor plus the first motion vector difference.

18. The device of claim 11, wherein the one or more processors are further configured to:

add first residual data to the first predictive block to generate a first reconstructed block of video data;

process the first reconstructed block of video data to generate a first decoded block of video data;

use the second motion vector, locating a second reference block in a second reference picture;

generate a second predictive block for the second block of the video data based on the second reference block;

add second residual data to the second predictive block to generate a second reconstructed block of video data;

process the second reconstructed block of video data to generate a second decoded block of video data; and

output a picture comprises the first decoded block of video data and the second decoded block of video data.

19. The device of claim 18, wherein to output the picture, the one or more processors are further configured to one or both of:

store a copy of the picture in a decoded picture buffer for use as reference picture in decoding subsequent pictures of video data; and

output the picture to a display device.

20. The device of claim 11, wherein the one or more processors are further configured to perform decoding as part of a reconstruction loop of a video encoding process.

21. The device of claim 11, wherein the device comprises a wireless communication device, further comprising a receiver configured to receive encoded video data.

22. The device of claim 21, wherein the wireless communication device comprises a telephone handset and wherein the receiver is configured to demodulate, according to a wireless communication standard, a signal comprising the encoded video data.

23. The device of claim 11, wherein the device comprises a wireless communication device, further comprising a transmitter configured to transmit encoded video data.

24. The device of claim 23, wherein the wireless communication device comprises a telephone handset and wherein the transmitter is configured to modulate, according to a wireless communication standard, a signal comprising the encoded video data.

25. A computer-readable storage medium storing instructions that when executed by one or more processors causes the one or more processors to:

determine a first block of video data is coded in an inter prediction mode;

determine a first motion vector for the first block of video data;

perform motion vector refinement on the first motion vector for the first block to determine a refined motion vector for the first block of video data;

locate a first reference block in a first reference picture using the refined motion vector;

generate a first predictive block for the first block of video data based on the first reference block;

determine that a second block of video data is coded in a mode that utilizes a motion vector associated with the first block as a motion vector predictor;

in response to determining that the second block of video data is coded in the mode that utilizes the motion vector associated with the first block as a motion vector predictor and in response to performing the motion vector refinement on the first motion vector for the first block, use a different motion vector than the first refined motion vector as the motion vector predictor associated with the first block;

based on the different motion vector, determine a second motion vector for the second block;

locate a second reference block in a second reference picture using the second motion vector; and

decode a picture of video data based on the first reference block and the second reference block.

26. The computer-readable storage medium of claim 25, wherein performing the motion vector refinement on the first motion vector for the first block comprises:

determining that a decoder-side motion vector derivation (DMVD) mode is enabled for the first block of video data; and

performing the motion vector refinement on the first motion vector for the first block in accordance with the DMVD mode.

27. The computer-readable storage medium of claim 26, wherein the DMVD mode comprises one of frame-rate up conversion (FRUC) template matching, FRUC bilateral matching, and bilateral template matching.

28. The computer-readable storage medium of claim 25, wherein the different motion vector comprises the first motion vector for the first block.

29. The computer-readable storage medium of claim 25, wherein the different motion vector comprises a default motion vector.

30. The computer-readable storage medium of claim 25, further comprising:

determining that the second block is coded in a merge mode;

generating a candidate list for the second block, wherein generating the candidate list for the second block comprises, in response to determining that the first block is coded using bilateral template matching, adding the different motion vector to the candidate list.

31. The computer-readable storage medium of claim 25, wherein:

determining the first motion vector for the first block of video data comprises determining a first motion vector prediction and a first motion vector difference;

performing motion vector refinement on the first motion vector for the first block comprises refining the first motion vector predictor to determine a first refined motion vector predictor;

setting the refined motion vector for the first block of video data equal to the first refined motion vector predictor plus the first motion vector difference; and

setting the motion vector predictor equal to the first motion vector predictor plus the first motion vector difference.

32. The computer-readable storage medium of claim 25, further comprising:

adding first residual data to the first predictive block to generate a first reconstructed block of video data;

processing the first reconstructed block of video data to generate a first decoded block of video data;

using the second motion vector, locating a second reference block in a second reference picture;

generating a second predictive block for the second block of video data based on the second reference block;

adding second residual data to the second predictive block to generate a second reconstructed block of video data;

processing the second reconstructed block of video data to generate a second decoded block of video data; and

outputting a picture comprises the first decoded block of video data and the second decoded block of video data.

33. The computer-readable storage medium of claim 32, wherein outputting the picture comprises one or both of:

storing a copy of the picture in a decoded picture buffer for use as reference picture in decoding subsequent pictures of video data; and

outputting the picture to a display device.

34. The computer-readable storage medium of claim 25, wherein the method of decoding is performed as part of a reconstruction loop of a video encoding process.

35. An apparatus for decoding video data, the apparatus comprising:

means for determining a first block of video data is coded in an inter prediction mode;

means for determining a first motion vector for the first block of video data;

means for performing motion vector refinement on the first motion vector for the first block to determine a refined motion vector for the first block of video data;

means for locating a first reference block in a first reference picture using the refined motion vector;

means for generating a first predictive block for the first block of video data based on the first reference block;

means for determining that a second block of video data is coded in a mode that utilizes a motion vector associated with the first block as a motion vector predictor;

means for using a different motion vector than the first refined motion vector as the motion vector predictor associated with the first block in response to determining that the second block of video data is coded in the mode that utilizes the motion vector associated with the first block as a motion vector predictor and in response to performing the motion vector refinement on the first motion vector for the first block;

means for determining a second motion vector for the second block based on the different motion vector;

means for locating a second reference block in a second reference picture using the second motion vector; and

means for decoding a picture of video data based on the first reference block and the second reference block.

36. The apparatus of claim 35, wherein the means for performing the motion vector refinement on the first motion vector for the first block comprises:

means for determining that a decoder-side motion vector derivation (DMVD) mode is enabled for the first block of video data; and

means for performing the motion vector refinement on the first motion vector for the first block in accordance with the DMVD mode.

37. The apparatus of claim 36, wherein the DMVD mode comprises one of frame-rate up conversion (FRUC) template matching, FRUC bilateral matching, and bilateral template matching.

38. The apparatus of claim 35, wherein the different motion vector comprises the first motion vector for the first block.

39. The apparatus of claim 35, wherein the different motion vector comprises a default motion vector.