

SCHEMES FOR ADJUSTING ADAPTIVE RESOLUTION FOR MOTION VECTOR DIFFERENCE

INCORPORATION BY REFERENCE

[0001] This application is based on and claims the benefit of priority to U.S. Non-Provisional Application No. 17/824,193 filed on May 25, 2022, entitled “Schemes for Adjusting Adaptive Resolution for Motion Vector Difference,” which is based on and claims the benefit of priority to U.S. Provisional Patent Application No. 63/302,518, filed on January 24, 2022, entitled “Further Improvement for Adaptive MVD Resolution.” These prior applications are herein incorporated by reference in their entireties.

TECHNICAL FIELD

[0002] This disclosure relates generally to video coding and particularly to methods and systems for providing schemes for setting allowable motion vector different values when implementing adaptive resolution for motion vector difference.

BACKGROUND

[0003] This background description provided herein is for the purpose of generally presenting the context of this disclosure. Work of the presently named inventors, to the extent the work is described in this background section, as well as aspects of the description that may not otherwise qualify as prior art at the time of filing of this application, are neither expressly nor impliedly admitted as prior art against the present disclosure.

[0004] Video coding and decoding can be performed using inter-picture prediction with motion compensation. Uncompressed digital video can include a series of pictures, with each picture having a spatial dimension of, for example, 1920 x 1080 luminance samples and associated full or subsampled chrominance samples. The series of pictures can have a fixed or variable picture rate (alternatively referred to as frame rate) of, for example, 60 pictures per second or 60 frames per second. Uncompressed video has specific bitrate requirements for streaming or data processing. For example, video with a pixel resolution of 1920 x 1080, a frame rate of 60 frames/second, and a chroma subsampling of 4:2:0 at 8 bit per pixel per color channel requires close to 1.5 Gbit/s bandwidth. An hour of such video requires more than 600 GBytes of storage space.

[0005] One purpose of video coding and decoding can be the reduction of redundancy in the uncompressed input video signal, through compression. Compression can help reduce the aforementioned bandwidth and/or storage space requirements, in some cases, by two

orders of magnitude or more. Both lossless compression and lossy compression, as well as a combination thereof can be employed. Lossless compression refers to techniques where an exact copy of the original signal can be reconstructed from the compressed original signal via a decoding process. Lossy compression refers to coding/decoding process where original video information is not fully retained during coding and not fully recoverable during decoding. When using lossy compression, the reconstructed signal may not be identical to the original signal, but the distortion between original and reconstructed signals is made small enough to render the reconstructed signal useful for the intended application albeit some information loss. In the case of video, lossy compression is widely employed in many applications. The amount of tolerable distortion depends on the application. For example, users of certain consumer video streaming applications may tolerate higher distortion than users of cinematic or television broadcasting applications. The compression ratio achievable by a particular coding algorithm can be selected or adjusted to reflect various distortion tolerance: higher tolerable distortion generally allows for coding algorithms that yield higher losses and higher compression ratios.

[0006] A video encoder and decoder can utilize techniques from several broad categories and steps, including, for example, motion compensation, Fourier transform, quantization, and entropy coding.

[0007] Video codec technologies can include techniques known as intra coding. In intra coding, sample values are represented without reference to samples or other data from previously reconstructed reference pictures. In some video codecs, a picture is spatially subdivided into blocks of samples. When all blocks of samples are coded in intra mode, that picture can be referred to as an intra picture. Intra pictures and their derivatives such as independent decoder refresh pictures, can be used to reset the decoder state and can, therefore, be used as the first picture in a coded video bitstream and a video session, or as a still image. The samples of a block after intra prediction can then be subject to a transform into frequency domain, and the transform coefficients so generated can be quantized before entropy coding. Intra prediction represents a technique that minimizes sample values in the pre-transform domain. In some cases, the smaller the DC value after a transform is, and the smaller the AC coefficients are, the fewer the bits that are required at a given quantization step size to represent the block after entropy coding.

[0008] Traditional intra coding such as that known from, for example, MPEG-2 generation coding technologies, does not use intra prediction. However, some newer video compression technologies include techniques that attempt coding/decoding of blocks based

on, for example, surrounding sample data and/or metadata that are obtained during the encoding and/or decoding of spatially neighboring, and that precede in decoding order the blocks of data being intra coded or decoded. Such techniques are henceforth called “intra prediction” techniques. Note that in at least some cases, intra prediction uses reference data only from the current picture under reconstruction and not from other reference pictures.

[0009] There can be many different forms of intra prediction. When more than one of such techniques are available in a given video coding technology, the technique in use can be referred to as an intra prediction mode. One or more intra prediction modes may be provided in a particular codec. In certain cases, modes can have submodes and/or may be associated with various parameters, and mode/submode information and intra coding parameters for blocks of video can be coded individually or collectively included in mode codewords. Which codeword to use for a given mode, submode, and/or parameter combination can have an impact in the coding efficiency gain through intra prediction, and so can the entropy coding technology used to translate the codewords into a bitstream.

[0010] A certain mode of intra prediction was introduced with H.264, refined in H.265, and further refined in newer coding technologies such as joint exploration model (JEM), versatile video coding (VVC), and benchmark set (BMS). Generally, for intra prediction, a predictor block can be formed using neighboring sample values that have become available. For example, available values of particular set of neighboring samples along certain direction and/or lines may be copied into the predictor block. A reference to the direction in use can be coded in the bitstream or may itself be predicted.

[0011] Referring to FIG. 1A, depicted in the lower right is a subset of nine predictor directions specified in H.265’s 33 possible intra predictor directions (corresponding to the 33 angular modes of the 35 intra modes specified in H.265). The point where the arrows converge (101) represents the sample being predicted. The arrows represent the direction from which neighboring samples are used to predict the sample at 101. For example, arrow (102) indicates that sample (101) is predicted from a neighboring sample or samples to the upper right, at a 45-degree angle from the horizontal direction. Similarly, arrow (103) indicates that sample (101) is predicted from a neighboring sample or samples to the lower left of sample (101), in a 22.5-degree angle from the horizontal direction.

[0012] Still referring to FIG. 1A, on the top left there is depicted a square block (104) of 4 x 4 samples (indicated by a dashed, boldface line). The square block (104) includes 16 samples, each labelled with an “S”, its position in the Y dimension (e.g., row index) and its position in the X dimension (e.g., column index). For example, sample S21 is the second

sample in the Y dimension (from the top) and the first (from the left) sample in the X dimension. Similarly, sample S44 is the fourth sample in block (104) in both the Y and X dimensions. As the block is 4 x 4 samples in size, S44 is at the bottom right. Further shown are example reference samples that follow a similar numbering scheme. A reference sample is labelled with an R, its Y position (e.g., row index) and X position (column index) relative to block (104). In both H.264 and H.265, prediction samples adjacently neighboring the block under reconstruction are used.

[0013] Intra picture prediction of block 104 may begin by copying reference sample values from the neighboring samples according to a signaled prediction direction. For example, assuming that the coded video bitstream includes signaling that, for this block 104, indicates a prediction direction of arrow (102)—that is, samples are predicted from a prediction sample or samples to the upper right, at a 45-degree angle from the horizontal direction. In such a case, samples S41, S32, S23, and S14 are predicted from the same reference sample R05. Sample S44 is then predicted from reference sample R08.

[0014] In certain cases, the values of multiple reference samples may be combined, for example through interpolation, in order to calculate a reference sample; especially when the directions are not evenly divisible by 45 degrees.

[0015] The number of possible directions has increased as video coding technology has continued to develop. In H.264 (year 2003), for example, nine different direction are available for intra prediction. That increased to 33 in H.265 (year 2013), and JEM/VVC/BMS, at the time of this disclosure, can support up to 65 directions. Experimental studies have been conducted to help identify the most suitable intra prediction directions, and certain techniques in the entropy coding may be used to encode those most suitable directions in a small number of bits, accepting a certain bit penalty for directions. Further, the directions themselves can sometimes be predicted from neighboring directions used in the intra prediction of the neighboring blocks that have been decoded.

[0016] FIG. 1B shows a schematic (180) that depicts 65 intra prediction directions according to JEM to illustrate the increasing number of prediction directions in various encoding technologies developed over time.

[0017] The manner for mapping of bits representing intra prediction directions to the prediction directions in the coded video bitstream may vary from video coding technology to video coding technology; and can range, for example, from simple direct mappings of prediction direction to intra prediction mode, to codewords, to complex adaptive schemes involving most probable modes, and similar techniques. In all cases, however, there can be

certain directions for intro prediction that are statistically less likely to occur in video content than certain other directions. As the goal of video compression is the reduction of redundancy, those less likely directions will, in a well-designed video coding technology, may be represented by a larger number of bits than more likely directions.

[0018] Inter picture prediction, or inter prediction may be based on motion compensation. In motion compensation, sample data from a previously reconstructed picture or part thereof (reference picture), after being spatially shifted in a direction indicated by a motion vector (MV henceforth), may be used for a prediction of a newly reconstructed picture or picture part (e.g., a block). In some cases, the reference picture can be the same as the picture currently under reconstruction. MVs may have two dimensions X and Y, or three dimensions, with the third dimension being an indication of the reference picture in use (akin to a time dimension).

[0019] In some video compression techniques, a current MV applicable to a certain area of sample data can be predicted from other MVs, for example from those other MVs that are related to other areas of the sample data that are spatially adjacent to the area under reconstruction and precede the current MV in decoding order. Doing so can substantially reduce the overall amount of data required for coding the MVs by relying on removing redundancy in correlated MVs, thereby increasing compression efficiency. MV prediction can work effectively, for example, because when coding an input video signal derived from a camera (known as natural video) there is a statistical likelihood that areas larger than the area to which a single MV is applicable move in a similar direction in the video sequence and, therefore, can in some cases be predicted using a similar motion vector derived from MVs of neighboring area. That results in the actual MV for a given area to be similar or identical to the MV predicted from the surrounding MVs. Such an MV in turn may be represented, after entropy coding, in a smaller number of bits than what would be used if the MV is coded directly rather than predicted from the neighboring MV(s). In some cases, MV prediction can be an example of lossless compression of a signal (namely: the MVs) derived from the original signal (namely: the sample stream). In other cases, MV prediction itself can be lossy, for example because of rounding errors when calculating a predictor from several surrounding MVs.

[0020] Various MV prediction mechanisms are described in H.265/HEVC (ITU-T Rec. H.265, "High Efficiency Video Coding", December 2016). Out of the many MV prediction mechanisms that H.265 specifies, described below is a technique henceforth referred to as "spatial merge".

[0021] Specifically, referring to FIG. 2, a current block (201) comprises samples that have been found by the encoder during the motion search process to be predictable from a previous block of the same size that has been spatially shifted. Instead of coding that MV directly, the MV can be derived from metadata associated with one or more reference pictures, for example from the most recent (in decoding order) reference picture, using the MV associated with either one of five surrounding samples, denoted A0, A1, and B0, B1, B2 (202 through 206, respectively). In H.265, the MV prediction can use predictors from the same reference picture that the neighboring block uses.

SUMMARY

[0022] This disclosure relates generally to video coding and particularly to methods and systems for signaling various motion vector or motion vector difference related syntax based on whether magnitude-dependent adaptive resolution for motion vector difference in inter-prediction is employed or not.

[0023] In an example implementation, a method for processing a current video block of a video stream is disclosed. The method may include receiving the video stream; determining that the current video block is inter-coded based on a prediction block and a motion vector (MV), wherein the MV is to be derived from a reference motion vector (RMV) and a motion vector difference (MVD) for the current video block. The method further include in response to determining that the MVD is coded with an adaptive MVD pixel resolution: determining a reference MVD pixel precision for the current video block; identifying a maximum allowed MVD pixel precision; determining a set of allowable MVD levels for the current video block based on the reference MVD pixel precision and the maximum allowed MVD pixel precision; and deriving the MVD from the video stream according to at least one MVD parameters signaled in the video stream for the current video block and the set of allowable MVD levels.

[0024] In the implementation above, the reference MVD pixel precision for the current video block is specified/signaled/derived at a sequence level, a picture level, a frame level, a superblock level, or coding block level.

[0025] In any one of the implementations above, the reference MVD pixel precision for the current video block depends on an MVD class associated with the MVD of the current video block.

[0026] In any one of the implementations above, the reference MVD pixel precision for the current video block depends on an MVD magnitude of the MVD of the current video

block. In any one of the implementations above the maximum allowed MVD pixel precision is predefined.

[0027] In any one of the implementations above, the method may further include determining a current MVD class among a predefined set of MVD classes. Determining the set of allowable MVD levels for the MVD based on the reference MVD pixel precision and the maximum allowed MVD pixel precision may include excluding, from a reference MVD level set determined based on the reference MVD pixel precision and the current MVD class, MVD levels associated with an MVD pixel precision at or higher than the maximum allowed MVD pixel precision, to determine the set of allowable MVD levels for the current video block.

[0028] In any one of the implementations above the maximum allowed MVD pixel precision is $\frac{1}{4}$ pixel. In any one of the implementations above, MVD levels associated with a $\frac{1}{8}$ pixel or higher precision or higher are excluded in the set of allowable MVD levels for the current video block.

[0029] In any one of the implementations above, the method may further include determining a current MVD class among a predefined set of MVD classes. MVD levels associated with fractional MVD precision may be included in the set of allowable MVD levels regardless of the reference MVD precision when the current MVD class is at or lower than a threshold MVD class.

[0030] In any one of the implementations above, the threshold MVD class may be a lowest MVD class among the predefined set of MVD classes.

[0031] In any one of the implementations above, the method may further include determining a magnitude of the MVD, wherein MVD levels associated with an MVD precision higher than a threshold MVD precision are allowed in the set of allowable MVD levels only when the magnitude of the MVD is at or lower than a threshold MVD magnitude.

[0032] In any one of the implementations above, the threshold MVD magnitude is 2 pixels or smaller. In any one of the implementations above the threshold MVD precision is 1 pixel. In any one of the implementations above MVD levels associated with an MVD precision of $\frac{1}{4}$ pixel or higher are only allowed when the magnitude of the MVD is at or lower than $\frac{1}{2}$ pixel. In any one of the implementations above, the maximum allowed MVD pixel precision may not be greater than the reference MVD pixel precision.

[0033] In another example implementation, a method for processing a current video block of a video stream is disclosed. The method may include receiving the video stream; determining that the current video block is inter-coded and associated with multiple reference

frames; and determining whether adaptive motion vector difference (MVD) pixel resolution is applied to at least one of the multiple reference frames based on a signaling in the video stream.

[0034] In the implementation above, the signaling may include a single-bit flag to indicate whether adaptive MVD pixel resolution is applied to all or none of the multiple reference frames.

[0035] In any one of the implementations above, the signaling may include separate flags each corresponding to one of the multiple reference frames to indicate whether adaptive MVD pixel resolution is applied.

[0036] In any one of the implementations above the signaling may include, for each of the multiple reference frames: implicit indication that no adaptive MVD pixel resolution is applied when an MVD corresponding to the each of the multiple reference frames is zero; and a single-bit flag for indicating whether adaptive MVD pixel resolution is applied when the MVD corresponding to the each of the multiple reference frames is non-zero.

[0037] In another example implementation, a method for processing a current video block of a video stream is disclosed. The method may include receiving the video stream; determining that the current video block is inter-coded based on a prediction block and a motion vector (MV), wherein the MV is to be derived from a reference motion vector (RMV) and a motion vector difference (MVD) for the current video block; determining a current MVD class of the MVD among a predefined set of MVD classes; deriving, based on the current MVD class, at least one context for entropy-decoding at least one explicit signaling in the video stream, the at least one explicit signaling being included in the video stream to specify an MVD pixel resolution for at least one component of the MVD; and entropy-decoding the at least one explicit signaling from the video stream using the at least one context to determine the MVD pixel resolution for the at least one component of the MVD.

[0038] In the implementation above, the at least one component of the MVD may include a horizontal component and a vertical component of the MVD, and the at least one context may include two separate contexts each associated with one of the horizontal component and the vertical component of the MVD, the horizontal component and the vertical component being associated with separate MVD pixel resolutions.

[0039] Aspects of the disclosure also provide a video encoding or decoding device or apparatus including a circuitry configured to carry out any of the method implementations above.

[0040] Aspects of the disclosure also provide non-transitory computer-readable mediums storing instructions which when executed by a computer for video decoding and/or encoding cause the computer to perform the methods for video decoding and/or encoding.

BRIEF DESCRIPTION OF THE DRAWINGS

[0041] Further features, the nature, and various advantages of the disclosed subject matter will be more apparent from the following detailed description and the accompanying drawings in which:

[0042] FIG. 1A shows a schematic illustration of an exemplary subset of intra prediction directional modes;

[0043] FIG. 1B shows an illustration of exemplary intra prediction directions;

[0044] FIG. 2 shows a schematic illustration of a current block and its surrounding spatial merge candidates for motion vector prediction in one example;

[0045] FIG. 3 shows a schematic illustration of a simplified block diagram of a communication system (300) in accordance with an example embodiment;

[0046] FIG. 4 shows a schematic illustration of a simplified block diagram of a communication system (400) in accordance with an example embodiment;

[0047] FIG. 5 shows a schematic illustration of a simplified block diagram of a video decoder in accordance with an example embodiment;

[0048] FIG. 6 shows a schematic illustration of a simplified block diagram of a video encoder in accordance with an example embodiment;

[0049] FIG. 7 shows a block diagram of a video encoder in accordance with another example embodiment;

[0050] FIG. 8 shows a block diagram of a video decoder in accordance with another example embodiment;

[0051] FIG. 9 shows a scheme of coding block partitioning according to example embodiments of the disclosure;

[0052] FIG. 10 shows another scheme of coding block partitioning according to example embodiments of the disclosure;

[0053] FIG. 11 shows another scheme of coding block partitioning according to example embodiments of the disclosure;

[0054] FIG. 12 shows an example partitioning of a base block into coding blocks according to an example partitioning scheme;

[0055] FIG. 13 shows an example ternary partitioning scheme;

[0056] FIG. 14 shows an example quadtree binary tree coding block partitioning scheme;

[0057] FIG. 15 shows a scheme for partitioning a coding block into multiple transform blocks and coding order of the transform blocks according to example embodiments of the disclosure;

[0058] FIG. 16 shows another scheme for partitioning a coding block into multiple transform blocks and coding order of the transform block according to example embodiments of the disclosure;

[0059] FIG. 17 shows another scheme for partitioning a coding block into multiple transform blocks according to example embodiments of the disclosure;

[0060] FIG. 18 shows a flow chart of a method according to an example embodiment of the disclosure;

[0061] FIG. 19 shows another flow chart of a method according to an example embodiment of the disclosure;

[0062] FIG. 20 shows another flow chart of a method according to an example embodiment of the disclosure; and

[0063] FIG. 21 shows a schematic illustration of a computer system in accordance with example embodiments of the disclosure.

DETAILED DESCRIPTION OF EMBODIMENTS

[0064] Throughout the specification and claims, terms may have nuanced meanings suggested or implied in context beyond an explicitly stated meaning. The phrase “in one embodiment” or “in some embodiments” as used herein does not necessarily refer to the same embodiment and the phrase “in another embodiment” or “in other embodiments” as used herein does not necessarily refer to a different embodiment. Likewise, the phrase “in one implementation” or “in some implementations” as used herein does not necessarily refer to the same implementation and the phrase “in another implementation” or “in other implementations” as used herein does not necessarily refer to a different implementation. It is intended, for example, that claimed subject matter includes combinations of exemplary embodiments/implementations in whole or in part.

[0065] In general, terminology may be understood at least in part from usage in context. For example, terms, such as “and”, “or”, or “and/or,” as used herein may include a variety of meanings that may depend at least in part upon the context in which such terms are used. Typically, “or” if used to associate a list, such as A, B or C, is intended to mean A, B,

and C, here used in the inclusive sense, as well as A, B or C, here used in the exclusive sense. In addition, the term “one or more” or “at least one” as used herein, depending at least in part upon context, may be used to describe any feature, structure, or characteristic in a singular sense or may be used to describe combinations of features, structures or characteristics in a plural sense. Similarly, terms, such as “a”, “an”, or “the”, again, may be understood to convey a singular usage or to convey a plural usage, depending at least in part upon context. In addition, the term “based on” or “determined by” may be understood as not necessarily intended to convey an exclusive set of factors and may, instead, allow for existence of additional factors not necessarily expressly described, again, depending at least in part on context. FIG. 3 illustrates a simplified block diagram of a communication system (300) according to an embodiment of the present disclosure. The communication system (300) includes a plurality of terminal devices that can communicate with each other, via, for example, a network (350). For example, the communication system (300) includes a first pair of terminal devices (310) and (320) interconnected via the network (350). In the example of FIG. 3, the first pair of terminal devices (310) and (320) may perform unidirectional transmission of data. For example, the terminal device (310) may code video data (e.g., of a stream of video pictures that are captured by the terminal device (310)) for transmission to the other terminal device (320) via the network (350). The encoded video data can be transmitted in the form of one or more coded video bitstreams. The terminal device (320) may receive the coded video data from the network (350), decode the coded video data to recover the video pictures and display the video pictures according to the recovered video data. Unidirectional data transmission may be implemented in media serving applications and the like.

[0066] In another example, the communication system (300) includes a second pair of terminal devices (330) and (340) that perform bidirectional transmission of coded video data that may be implemented, for example, during a videoconferencing application. For bidirectional transmission of data, in an example, each terminal device of the terminal devices (330) and (340) may code video data (e.g., of a stream of video pictures that are captured by the terminal device) for transmission to the other terminal device of the terminal devices (330) and (340) via the network (350). Each terminal device of the terminal devices (330) and (340) also may receive the coded video data transmitted by the other terminal device of the terminal devices (330) and (340), and may decode the coded video data to recover the video pictures and may display the video pictures at an accessible display device according to the recovered video data.

[0067] In the example of FIG. 3, the terminal devices (310), (320), (330) and (340) may be implemented as servers, personal computers and smart phones but the applicability of the underlying principles of the present disclosure may not be so limited. Embodiments of the present disclosure may be implemented in desktop computers, laptop computers, tablet computers, media players, wearable computers, dedicated video conferencing equipment, and/or the like. The network (350) represents any number or types of networks that convey coded video data among the terminal devices (310), (320), (330) and (340), including for example wireline (wired) and/or wireless communication networks. The communication network (350) may exchange data in circuit-switched, packet-switched, and/or other types of channels. Representative networks include telecommunications networks, local area networks, wide area networks and/or the Internet. For the purposes of the present discussion, the architecture and topology of the network (350) may be immaterial to the operation of the present disclosure unless explicitly explained herein.

[0068] FIG. 4 illustrates, as an example for an application for the disclosed subject matter, a placement of a video encoder and a video decoder in a video streaming environment. The disclosed subject matter may be equally applicable to other video applications, including, for example, video conferencing, digital TV broadcasting, gaming, virtual reality, storage of compressed video on digital media including CD, DVD, memory stick and the like, and so on.

[0069] A video streaming system may include a video capture subsystem (413) that can include a video source (401), e.g., a digital camera, for creating a stream of video pictures or images (402) that are uncompressed. In an example, the stream of video pictures (402) includes samples that are recorded by a digital camera of the video source 401. The stream of video pictures (402), depicted as a bold line to emphasize a high data volume when compared to encoded video data (404) (or coded video bitstreams), can be processed by an electronic device (420) that includes a video encoder (403) coupled to the video source (401). The video encoder (403) can include hardware, software, or a combination thereof to enable or implement aspects of the disclosed subject matter as described in more detail below. The encoded video data (404) (or encoded video bitstream (404)), depicted as a thin line to emphasize a lower data volume when compared to the stream of uncompressed video pictures (402), can be stored on a streaming server (405) for future use or directly to downstream video devices (not shown). One or more streaming client subsystems, such as client subsystems (406) and (408) in FIG. 4 can access the streaming server (405) to retrieve copies (407) and (409) of the encoded video data (404). A client subsystem (406) can include a

video decoder (410), for example, in an electronic device (430). The video decoder (410) decodes the incoming copy (407) of the encoded video data and creates an outgoing stream of video pictures (411) that are uncompressed and that can be rendered on a display (412) (e.g., a display screen) or other rendering devices (not depicted). The video decoder 410 may be configured to perform some or all of the various functions described in this disclosure. In some streaming systems, the encoded video data (404), (407), and (409) (e.g., video bitstreams) can be encoded according to certain video coding/compression standards. Examples of those standards include ITU-T Recommendation H.265. In an example, a video coding standard under development is informally known as Versatile Video Coding (VVC). The disclosed subject matter may be used in the context of VVC, and other video coding standards.

[0070] It is noted that the electronic devices (420) and (430) can include other components (not shown). For example, the electronic device (420) can include a video decoder (not shown) and the electronic device (430) can include a video encoder (not shown) as well.

[0071] FIG. 5 shows a block diagram of a video decoder (510) according to any embodiment of the present disclosure below. The video decoder (510) can be included in an electronic device (530). The electronic device (530) can include a receiver (531) (e.g., receiving circuitry). The video decoder (510) can be used in place of the video decoder (410) in the example of FIG. 4.

[0072] The receiver (531) may receive one or more coded video sequences to be decoded by the video decoder (510). In the same or another embodiment, one coded video sequence may be decoded at a time, where the decoding of each coded video sequence is independent from other coded video sequences. Each video sequence may be associated with multiple video frames or images. The coded video sequence may be received from a channel (501), which may be a hardware/software link to a storage device which stores the encoded video data or a streaming source which transmits the encoded video data. The receiver (531) may receive the encoded video data with other data such as coded audio data and/or ancillary data streams, that may be forwarded to their respective processing circuitry (not depicted). The receiver (531) may separate the coded video sequence from the other data. To combat network jitter, a buffer memory (515) may be disposed in between the receiver (531) and an entropy decoder / parser (520) ("parser (520)" henceforth). In certain applications, the buffer memory (515) may be implemented as part of the video decoder (510). In other applications, it can be outside of and separate from the video decoder (510) (not depicted). In still other

applications, there can be a buffer memory (not depicted) outside of the video decoder (510) for the purpose of, for example, combating network jitter, and there may be another additional buffer memory (515) inside the video decoder (510), for example to handle playback timing. When the receiver (531) is receiving data from a store/forward device of sufficient bandwidth and controllability, or from an isosynchronous network, the buffer memory (515) may not be needed, or can be small. For use on best-effort packet networks such as the Internet, the buffer memory (515) of sufficient size may be required, and its size can be comparatively large. Such buffer memory may be implemented with an adaptive size, and may at least partially be implemented in an operating system or similar elements (not depicted) outside of the video decoder (510).

[0073] The video decoder (510) may include the parser (520) to reconstruct symbols (521) from the coded video sequence. Categories of those symbols include information used to manage operation of the video decoder (510), and potentially information to control a rendering device such as display (512) (e.g., a display screen) that may or may not be an integral part of the electronic device (530) but can be coupled to the electronic device (530), as is shown in FIG. 5. The control information for the rendering device(s) may be in the form of Supplemental Enhancement Information (SEI messages) or Video Usability Information (VUI) parameter set fragments (not depicted). The parser (520) may parse/entropy-decode the coded video sequence that is received by the parser (520). The entropy coding of the coded video sequence can be in accordance with a video coding technology or standard, and can follow various principles, including variable length coding, Huffman coding, arithmetic coding with or without context sensitivity, and so forth. The parser (520) may extract from the coded video sequence, a set of subgroup parameters for at least one of the subgroups of pixels in the video decoder, based upon at least one parameter corresponding to the subgroups. The subgroups can include Groups of Pictures (GOPs), pictures, tiles, slices, macroblocks, Coding Units (CUs), blocks, Transform Units (TUs), Prediction Units (PUs) and so forth. The parser (520) may also extract from the coded video sequence information such as transform coefficients (e.g., Fourier transform coefficients), quantizer parameter values, motion vectors, and so forth.

[0074] The parser (520) may perform an entropy decoding / parsing operation on the video sequence received from the buffer memory (515), so as to create symbols (521).

[0075] Reconstruction of the symbols (521) can involve multiple different processing or functional units depending on the type of the coded video picture or parts thereof (such as: inter and intra picture, inter and intra block), and other factors. The units that are involved

and how they are involved may be controlled by the subgroup control information that was parsed from the coded video sequence by the parser (520). The flow of such subgroup control information between the parser (520) and the multiple processing or functional units below is not depicted for simplicity.

[0076] Beyond the functional blocks already mentioned, the video decoder (510) can be conceptually subdivided into a number of functional units as described below. In a practical implementation operating under commercial constraints, many of these functional units interact closely with each other and can, at least partly, be integrated with one another. However, for the purpose of describing the various functions of the disclosed subject matter with clarity, the conceptual subdivision into the functional units is adopted in the disclosure below.

[0077] A first unit may include the scaler / inverse transform unit (551). The scaler / inverse transform unit (551) may receive a quantized transform coefficient as well as control information, including information indicating which type of inverse transform to use, block size, quantization factor/parameters, quantization scaling matrices, and the like as symbol(s) (521) from the parser (520). The scaler / inverse transform unit (551) can output blocks comprising sample values that can be input into aggregator (555).

[0078] In some cases, the output samples of the scaler / inverse transform (551) can pertain to an intra coded block, i.e., a block that does not use predictive information from previously reconstructed pictures, but can use predictive information from previously reconstructed parts of the current picture. Such predictive information can be provided by an intra picture prediction unit (552). In some cases, the intra picture prediction unit (552) may generate a block of the same size and shape of the block under reconstruction using surrounding block information that is already reconstructed and stored in the current picture buffer (558). The current picture buffer (558) buffers, for example, partly reconstructed current picture and/or fully reconstructed current picture. The aggregator (555), in some implementations, may add, on a per sample basis, the prediction information the intra prediction unit (552) has generated to the output sample information as provided by the scaler / inverse transform unit (551).

[0079] In other cases, the output samples of the scaler / inverse transform unit (551) can pertain to an inter coded, and potentially motion compensated block. In such a case, a motion compensation prediction unit (553) can access reference picture memory (557) to fetch samples used for inter-picture prediction. After motion compensating the fetched samples in accordance with the symbols (521) pertaining to the block, these samples can be

added by the aggregator (555) to the output of the scaler / inverse transform unit (551) (output of unit 551 may be referred to as the residual samples or residual signal) so as to generate output sample information. The addresses within the reference picture memory (557) from where the motion compensation prediction unit (553) fetches prediction samples can be controlled by motion vectors, available to the motion compensation prediction unit (553) in the form of symbols (521) that can have, for example X, Y components (shift), and reference picture components (time). Motion compensation may also include interpolation of sample values as fetched from the reference picture memory (557) when sub-sample exact motion vectors are in use, and may also be associated with motion vector prediction mechanisms, and so forth.

[0080] The output samples of the aggregator (555) can be subject to various loop filtering techniques in the loop filter unit (556). Video compression technologies can include in-loop filter technologies that are controlled by parameters included in the coded video sequence (also referred to as coded video bitstream) and made available to the loop filter unit (556) as symbols (521) from the parser (520), but can also be responsive to meta-information obtained during the decoding of previous (in decoding order) parts of the coded picture or coded video sequence, as well as responsive to previously reconstructed and loop-filtered sample values. Several type of loop filters may be included as part of the loop filter unit 556 in various orders, as will be described in further detail below.

[0081] The output of the loop filter unit (556) can be a sample stream that can be output to the rendering device (512) as well as stored in the reference picture memory (557) for use in future inter-picture prediction.

[0082] Certain coded pictures, once fully reconstructed, can be used as reference pictures for future inter-picture prediction. For example, once a coded picture corresponding to a current picture is fully reconstructed and the coded picture has been identified as a reference picture (by, for example, the parser (520)), the current picture buffer (558) can become a part of the reference picture memory (557), and a fresh current picture buffer can be reallocated before commencing the reconstruction of the following coded picture.

[0083] The video decoder (510) may perform decoding operations according to a predetermined video compression technology adopted in a standard, such as ITU-T Rec. H.265. The coded video sequence may conform to a syntax specified by the video compression technology or standard being used, in the sense that the coded video sequence adheres to both the syntax of the video compression technology or standard and the profiles as documented in the video compression technology or standard. Specifically, a profile can

select certain tools from all the tools available in the video compression technology or standard as the only tools available for use under that profile. To be standard-compliant, the complexity of the coded video sequence may be within bounds as defined by the level of the video compression technology or standard. In some cases, levels restrict the maximum picture size, maximum frame rate, maximum reconstruction sample rate (measured in, for example megasamples per second), maximum reference picture size, and so on. Limits set by levels can, in some cases, be further restricted through Hypothetical Reference Decoder (HRD) specifications and metadata for HRD buffer management signaled in the coded video sequence.

[0084] In some example embodiments, the receiver (531) may receive additional (redundant) data with the encoded video. The additional data may be included as part of the coded video sequence(s). The additional data may be used by the video decoder (510) to properly decode the data and/or to more accurately reconstruct the original video data. Additional data can be in the form of, for example, temporal, spatial, or signal noise ratio (SNR) enhancement layers, redundant slices, redundant pictures, forward error correction codes, and so on.

[0085] FIG. 6 shows a block diagram of a video encoder (603) according to an example embodiment of the present disclosure. The video encoder (603) may be included in an electronic device (620). The electronic device (620) may further include a transmitter (640) (e.g., transmitting circuitry). The video encoder (603) can be used in place of the video encoder (403) in the example of FIG. 4.

[0086] The video encoder (603) may receive video samples from a video source (601) (that is not part of the electronic device (620) in the example of FIG. 6) that may capture video image(s) to be coded by the video encoder (603). In another example, the video source (601) may be implemented as a portion of the electronic device (620).

[0087] The video source (601) may provide the source video sequence to be coded by the video encoder (603) in the form of a digital video sample stream that can be of any suitable bit depth (for example: 8 bit, 10 bit, 12 bit, ...), any colorspace (for example, BT.601 YCrCb, RGB, XYZ...), and any suitable sampling structure (for example YCrCb 4:2:0, YCrCb 4:4:4). In a media serving system, the video source (601) may be a storage device capable of storing previously prepared video. In a videoconferencing system, the video source (601) may be a camera that captures local image information as a video sequence. Video data may be provided as a plurality of individual pictures or images that impart motion when viewed in sequence. The pictures themselves may be organized as a spatial array of

pixels, wherein each pixel can comprise one or more samples depending on the sampling structure, color space, and the like being in use. A person having ordinary skill in the art can readily understand the relationship between pixels and samples. The description below focuses on samples.

[0088] According to some example embodiments, the video encoder (603) may code and compress the pictures of the source video sequence into a coded video sequence (643) in real time or under any other time constraints as required by the application. Enforcing appropriate coding speed constitutes one function of a controller (650). In some embodiments, the controller (650) may be functionally coupled to and control other functional units as described below. The coupling is not depicted for simplicity. Parameters set by the controller (650) can include rate control related parameters (picture skip, quantizer, lambda value of rate-distortion optimization techniques, ...), picture size, group of pictures (GOP) layout, maximum motion vector search range, and the like. The controller (650) can be configured to have other suitable functions that pertain to the video encoder (603) optimized for a certain system design.

[0089] In some example embodiments, the video encoder (603) may be configured to operate in a coding loop. As an oversimplified description, in an example, the coding loop can include a source coder (630) (e.g., responsible for creating symbols, such as a symbol stream, based on an input picture to be coded, and a reference picture(s)), and a (local) decoder (633) embedded in the video encoder (603). The decoder (633) reconstructs the symbols to create the sample data in a similar manner as a (remote) decoder would create even though the embedded decoder 633 process coded video stream by the source coder 630 without entropy coding (as any compression between symbols and coded video bitstream in entropy coding may be lossless in the video compression technologies considered in the disclosed subject matter). The reconstructed sample stream (sample data) is input to the reference picture memory (634). As the decoding of a symbol stream leads to bit-exact results independent of decoder location (local or remote), the content in the reference picture memory (634) is also bit exact between the local encoder and remote encoder. In other words, the prediction part of an encoder "sees" as reference picture samples exactly the same sample values as a decoder would "see" when using prediction during decoding. This fundamental principle of reference picture synchronicity (and resulting drift, if synchronicity cannot be maintained, for example because of channel errors) is used to improve coding quality.

[0090] The operation of the "local" decoder (633) can be the same as of a "remote" decoder, such as the video decoder (510), which has already been described in detail above in conjunction with FIG. 5. Briefly referring also to FIG. 5, however, as symbols are available and encoding/decoding of symbols to a coded video sequence by an entropy coder (645) and the parser (520) can be lossless, the entropy decoding parts of the video decoder (510), including the buffer memory (515), and parser (520) may not be fully implemented in the local decoder (633) in the encoder.

[0091] An observation that can be made at this point is that any decoder technology except the parsing/entropy decoding that may only be present in a decoder also may necessarily need to be present, in substantially identical functional form, in a corresponding encoder. For this reason, the disclosed subject matter may at times focus on decoder operation, which allies to the decoding portion of the encoder. The description of encoder technologies can thus be abbreviated as they are the inverse of the comprehensively described decoder technologies. Only in certain areas or aspects a more detail description of the encoder is provided below.

[0092] During operation in some example implementations, the source coder (630) may perform motion compensated predictive coding, which codes an input picture predictively with reference to one or more previously coded picture from the video sequence that were designated as "reference pictures." In this manner, the coding engine (632) codes differences (or residue) in the color channels between pixel blocks of an input picture and pixel blocks of reference picture(s) that may be selected as prediction reference(s) to the input picture. The term "residue" and its adjective form "residual" may be used interchangeably.

[0093] The local video decoder (633) may decode coded video data of pictures that may be designated as reference pictures, based on symbols created by the source coder (630). Operations of the coding engine (632) may advantageously be lossy processes. When the coded video data may be decoded at a video decoder (not shown in FIG. 6), the reconstructed video sequence typically may be a replica of the source video sequence with some errors. The local video decoder (633) replicates decoding processes that may be performed by the video decoder on reference pictures and may cause reconstructed reference pictures to be stored in the reference picture cache (634). In this manner, the video encoder (603) may store copies of reconstructed reference pictures locally that have common content as the reconstructed reference pictures that will be obtained by a far-end (remote) video decoder (absent transmission errors).

[0094] The predictor (635) may perform prediction searches for the coding engine (632). That is, for a new picture to be coded, the predictor (635) may search the reference picture memory (634) for sample data (as candidate reference pixel blocks) or certain metadata such as reference picture motion vectors, block shapes, and so on, that may serve as an appropriate prediction reference for the new pictures. The predictor (635) may operate on a sample block-by-pixel block basis to find appropriate prediction references. In some cases, as determined by search results obtained by the predictor (635), an input picture may have prediction references drawn from multiple reference pictures stored in the reference picture memory (634).

[0095] The controller (650) may manage coding operations of the source coder (630), including, for example, setting of parameters and subgroup parameters used for encoding the video data.

[0096] Output of all aforementioned functional units may be subjected to entropy coding in the entropy coder (645). The entropy coder (645) translates the symbols as generated by the various functional units into a coded video sequence, by lossless compression of the symbols according to technologies such as Huffman coding, variable length coding, arithmetic coding, and so forth.

[0097] The transmitter (640) may buffer the coded video sequence(s) as created by the entropy coder (645) to prepare for transmission via a communication channel (660), which may be a hardware/software link to a storage device which would store the encoded video data. The transmitter (640) may merge coded video data from the video coder (603) with other data to be transmitted, for example, coded audio data and/or ancillary data streams (sources not shown).

[0098] The controller (650) may manage operation of the video encoder (603). During coding, the controller (650) may assign to each coded picture a certain coded picture type, which may affect the coding techniques that may be applied to the respective picture. For example, pictures often may be assigned as one of the following picture types:

[0099] An Intra Picture (I picture) may be one that may be coded and decoded without using any other picture in the sequence as a source of prediction. Some video codecs allow for different types of intra pictures, including, for example Independent Decoder Refresh (“IDR”) Pictures. A person having ordinary skill in the art is aware of those variants of I pictures and their respective applications and features.

[0100] A predictive picture (P picture) may be one that may be coded and decoded using intra prediction or inter prediction using at most one motion vector and reference index to predict the sample values of each block.

[0101] A bi-directionally predictive picture (B Picture) may be one that may be coded and decoded using intra prediction or inter prediction using at most two motion vectors and reference indices to predict the sample values of each block. Similarly, multiple-predictive pictures can use more than two reference pictures and associated metadata for the reconstruction of a single block.

[0102] Source pictures commonly may be subdivided spatially into a plurality of sample coding blocks (for example, blocks of 4 x 4, 8 x 8, 4 x 8, or 16 x 16 samples each) and coded on a block-by-block basis. Blocks may be coded predictively with reference to other (already coded) blocks as determined by the coding assignment applied to the blocks' respective pictures. For example, blocks of I pictures may be coded non-predictively or they may be coded predictively with reference to already coded blocks of the same picture (spatial prediction or intra prediction). Pixel blocks of P pictures may be coded predictively, via spatial prediction or via temporal prediction with reference to one previously coded reference picture. Blocks of B pictures may be coded predictively, via spatial prediction or via temporal prediction with reference to one or two previously coded reference pictures. The source pictures or the intermediate processed pictures may be subdivided into other types of blocks for other purposes. The division of coding blocks and the other types of blocks may or may not follow the same manner, as described in further detail below.

[0103] The video encoder (603) may perform coding operations according to a predetermined video coding technology or standard, such as ITU-T Rec. H.265. In its operation, the video encoder (603) may perform various compression operations, including predictive coding operations that exploit temporal and spatial redundancies in the input video sequence. The coded video data may accordingly conform to a syntax specified by the video coding technology or standard being used.

[0104] In some example embodiments, the transmitter (640) may transmit additional data with the encoded video. The source coder (630) may include such data as part of the coded video sequence. The additional data may comprise temporal/spatial/SNR enhancement layers, other forms of redundant data such as redundant pictures and slices, SEI messages, VUI parameter set fragments, and so on.

[0105] A video may be captured as a plurality of source pictures (video pictures) in a temporal sequence. Intra-picture prediction (often abbreviated to intra prediction) utilizes

spatial correlation in a given picture, and inter-picture prediction utilizes temporal or other correlation between the pictures. For example, a specific picture under encoding/decoding, which is referred to as a current picture, may be partitioned into blocks. A block in the current picture, when similar to a reference block in a previously coded and still buffered reference picture in the video, may be coded by a vector that is referred to as a motion vector. The motion vector points to the reference block in the reference picture, and can have a third dimension identifying the reference picture, in case multiple reference pictures are in use.

[0106] In some example embodiments, a bi-prediction technique can be used for inter-picture prediction. According to such bi-prediction technique, two reference pictures, such as a first reference picture and a second reference picture that both proceed the current picture in the video in decoding order (but may be in the past or future, respectively, in display order) are used. A block in the current picture can be coded by a first motion vector that points to a first reference block in the first reference picture, and a second motion vector that points to a second reference block in the second reference picture. The block can be jointly predicted by a combination of the first reference block and the second reference block.

[0107] Further, a merge mode technique may be used in the inter-picture prediction to improve coding efficiency.

[0108] According to some example embodiments of the disclosure, predictions, such as inter-picture predictions and intra-picture predictions are performed in the unit of blocks. For example, a picture in a sequence of video pictures is partitioned into coding tree units (CTU) for compression, the CTUs in a picture may have the same size, such as 64 x 64 pixels, 32 x 32 pixels, or 16 x 16 pixels. In general, a CTU may include three parallel coding tree blocks (CTBs): one luma CTB and two chroma CTBs. Each CTU can be recursively quadtree split into one or multiple coding units (CUs). For example, a CTU of 64 x 64 pixels can be split into one CU of 64 x 64 pixels, or 4 CUs of 32 x 32 pixels. Each of the one or more of the 32 x 32 block may be further split into 4 CUs of 16 x 16 pixels. In some example embodiments, each CU may be analyzed during encoding to determine a prediction type for the CU among various prediction types such as an inter prediction type or an intra prediction type. The CU may be split into one or more prediction units (PUs) depending on the temporal and/or spatial predictability. Generally, each PU includes a luma prediction block (PB), and two chroma PBs. In an embodiment, a prediction operation in coding (encoding/decoding) is performed in the unit of a prediction block. The split of a CU into PU (or PBs of different color channels) may be performed in various spatial pattern. A luma or

chroma PB, for example, may include a matrix of values (e.g., luma values) for samples, such as 8 x 8 pixels, 16 x 16 pixels, 8 x 16 pixels, 16 x 8 samples, and the like.

[0109] FIG. 7 shows a diagram of a video encoder (703) according to another example embodiment of the disclosure. The video encoder (703) is configured to receive a processing block (e.g., a prediction block) of sample values within a current video picture in a sequence of video pictures, and encode the processing block into a coded picture that is part of a coded video sequence. The example video encoder (703) may be used in place of the video encoder (403) in the FIG. 4 example.

[0110] For example, the video encoder (703) receives a matrix of sample values for a processing block, such as a prediction block of 8 x 8 samples, and the like. The video encoder (703) then determines whether the processing block is best coded using intra mode, inter mode, or bi-prediction mode using, for example, rate-distortion optimization (RDO). When the processing block is determined to be coded in intra mode, the video encoder (703) may use an intra prediction technique to encode the processing block into the coded picture; and when the processing block is determined to be coded in inter mode or bi-prediction mode, the video encoder (703) may use an inter prediction or bi-prediction technique, respectively, to encode the processing block into the coded picture. In some example embodiments, a merge mode may be used as a submode of the inter picture prediction where the motion vector is derived from one or more motion vector predictors without the benefit of a coded motion vector component outside the predictors. In some other example embodiments, a motion vector component applicable to the subject block may be present. Accordingly, the video encoder (703) may include components not explicitly shown in FIG. 7, such as a mode decision module, to determine the prediction mode of the processing blocks.

[0111] In the example of FIG. 7, the video encoder (703) includes an inter encoder (730), an intra encoder (722), a residue calculator (723), a switch (726), a residue encoder (724), a general controller (721), and an entropy encoder (725) coupled together as shown in the example arrangement in FIG. 7.

[0112] The inter encoder (730) is configured to receive the samples of the current block (e.g., a processing block), compare the block to one or more reference blocks in reference pictures (e.g., blocks in previous pictures and later pictures in display order), generate inter prediction information (e.g., description of redundant information according to inter encoding technique, motion vectors, merge mode information), and calculate inter prediction results (e.g., predicted block) based on the inter prediction information using any suitable technique. In some examples, the reference pictures are decoded reference pictures

that are decoded based on the encoded video information using the decoding unit 633 embedded in the example encoder 620 of FIG. 6 (shown as residual decoder 728 of FIG. 7, as described in further detail below).

[0113] The intra encoder (722) is configured to receive the samples of the current block (e.g., a processing block), compare the block to blocks already coded in the same picture, and generate quantized coefficients after transform, and in some cases also to generate intra prediction information (e.g., an intra prediction direction information according to one or more intra encoding techniques). The intra encoder (722) may calculate intra prediction results (e.g., predicted block) based on the intra prediction information and reference blocks in the same picture.

[0114] The general controller (721) may be configured to determine general control data and control other components of the video encoder (703) based on the general control data. In an example, the general controller (721) determines the prediction mode of the block, and provides a control signal to the switch (726) based on the prediction mode. For example, when the prediction mode is the intra mode, the general controller (721) controls the switch (726) to select the intra mode result for use by the residue calculator (723), and controls the entropy encoder (725) to select the intra prediction information and include the intra prediction information in the bitstream; and when the prediction mode for the block is the inter mode, the general controller (721) controls the switch (726) to select the inter prediction result for use by the residue calculator (723), and controls the entropy encoder (725) to select the inter prediction information and include the inter prediction information in the bitstream.

[0115] The residue calculator (723) may be configured to calculate a difference (residue data) between the received block and prediction results for the block selected from the intra encoder (722) or the inter encoder (730). The residue encoder (724) may be configured to encode the residue data to generate transform coefficients. For example, the residue encoder (724) may be configured to convert the residue data from a spatial domain to a frequency domain to generate the transform coefficients. The transform coefficients are then subject to quantization processing to obtain quantized transform coefficients. In various example embodiments, the video encoder (703) also includes a residual decoder (728). The residual decoder (728) is configured to perform inverse-transform, and generate the decoded residue data. The decoded residue data can be suitably used by the intra encoder (722) and the inter encoder (730). For example, the inter encoder (730) can generate decoded blocks based on the decoded residue data and inter prediction information, and the intra encoder

(722) can generate decoded blocks based on the decoded residue data and the intra prediction information. The decoded blocks are suitably processed to generate decoded pictures and the decoded pictures can be buffered in a memory circuit (not shown) and used as reference pictures.

[0116] The entropy encoder (725) may be configured to format the bitstream to include the encoded block and perform entropy coding. The entropy encoder (725) is configured to include in the bitstream various information. For example, the entropy encoder (725) may be configured to include the general control data, the selected prediction information (e.g., intra prediction information or inter prediction information), the residue information, and other suitable information in the bitstream. When coding a block in the merge submode of either inter mode or bi-prediction mode, there may be no residue information.

[0117] FIG. 8 shows a diagram of an example video decoder (810) according to another embodiment of the disclosure. The video decoder (810) is configured to receive coded pictures that are part of a coded video sequence, and decode the coded pictures to generate reconstructed pictures. In an example, the video decoder (810) may be used in place of the video decoder (410) in the example of FIG. 4.

[0118] In the example of FIG. 8, the video decoder (810) includes an entropy decoder (871), an inter decoder (880), a residual decoder (873), a reconstruction module (874), and an intra decoder (872) coupled together as shown in the example arrangement of FIG. 8.

[0119] The entropy decoder (871) can be configured to reconstruct, from the coded picture, certain symbols that represent the syntax elements of which the coded picture is made up. Such symbols can include, for example, the mode in which a block is coded (e.g., intra mode, inter mode, bi-predicted mode, merge submode or another submode), prediction information (e.g., intra prediction information or inter prediction information) that can identify certain sample or metadata used for prediction by the intra decoder (872) or the inter decoder (880), residual information in the form of, for example, quantized transform coefficients, and the like. In an example, when the prediction mode is the inter or bi-predicted mode, the inter prediction information is provided to the inter decoder (880); and when the prediction type is the intra prediction type, the intra prediction information is provided to the intra decoder (872). The residual information can be subject to inverse quantization and is provided to the residual decoder (873).

[0120] The inter decoder (880) may be configured to receive the inter prediction information, and generate inter prediction results based on the inter prediction information.

[0121] The intra decoder (872) may be configured to receive the intra prediction information, and generate prediction results based on the intra prediction information.

[0122] The residual decoder (873) may be configured to perform inverse quantization to extract de-quantized transform coefficients, and process the de-quantized transform coefficients to convert the residual from the frequency domain to the spatial domain. The residual decoder (873) may also utilize certain control information (to include the Quantizer Parameter (QP)) which may be provided by the entropy decoder (871) (data path not depicted as this may be low data volume control information only).

[0123] The reconstruction module (874) may be configured to combine, in the spatial domain, the residual as output by the residual decoder (873) and the prediction results (as output by the inter or intra prediction modules as the case may be) to form a reconstructed block forming part of the reconstructed picture as part of the reconstructed video. It is noted that other suitable operations, such as a deblocking operation and the like, may also be performed to improve the visual quality.

[0124] It is noted that the video encoders (403), (603), and (703), and the video decoders (410), (510), and (810) can be implemented using any suitable technique. In some example embodiments, the video encoders (403), (603), and (703), and the video decoders (410), (510), and (810) can be implemented using one or more integrated circuits. In another embodiment, the video encoders (403), (603), and (603), and the video decoders (410), (510), and (810) can be implemented using one or more processors that execute software instructions.

[0125] Turning to block partitioning for coding and decoding, general partitioning may start from a base block and may follow a predefined ruleset, particular patterns, partition trees, or any partition structure or scheme. The partitioning may be hierarchical and recursive. After dividing or partitioning a base block following any of the example partitioning procedures or other procedures described below, or the combination thereof, a final set of partitions or coding blocks may be obtained. Each of these partitions may be at one of various partitioning levels in the partitioning hierarchy, and may be of various shapes. Each of the partitions may be referred to as a coding block (CB). For the various example partitioning implementations described further below, each resulting CB may be of any of the allowed sizes and partitioning levels. Such partitions are referred to as coding blocks because they may form units for which some basic coding/decoding decisions may be made and coding/decoding parameters may be optimized, determined, and signaled in an encoded video bitstream. The highest or deepest level in the final partitions represents the depth of the

coding block partitioning structure of tree. A coding block may be a luma coding block or a chroma coding block. The CB tree structure of each color may be referred to as coding block tree (CBT).

[0126] The coding blocks of all color channels may collectively be referred to as a coding unit (CU). The hierarchical structure of for all color channels may be collectively referred to as coding tree unit (CTU). The partitioning patterns or structures for the various color channels in in a CTU may or may not be the same.

[0127] In some implementations, partition tree schemes or structures used for the luma and chroma channels may not need to be the same. In other words, luma and chroma channels may have separate coding tree structures or patterns. Further, whether the luma and chroma channels use the same or different coding partition tree structures and the actual coding partition tree structures to be used may depend on whether the slice being coded is a P, B, or I slice. For example, For an I slice, the chroma channels and luma channel may have separate coding partition tree structures or coding partition tree structure modes, whereas for a P or B slice, the luma and chroma channels may share a same coding partition tree scheme. When separate coding partition tree structures or modes are applied, a luma channel may be partitioned into CBs by one coding partition tree structure, and a chroma channel may be partitioned into chroma CBs by another coding partition tree structure.

[0128] In some example implementations, a predetermined partitioning pattern may be applied to a base block. As shown in FIG. 9, an example 4-way partition tree may start from a first predefined level (e.g., 64 x 64 block level or other sizes, as a base block size) and a base block may be partitioned hierarchically down to a predefined lowest level (e.g., 4 x 4 level). For example, a base block may be subject to four predefined partitioning options or patterns indicated by 902, 904, 906, and 908, with the partitions designated as R being allowed for recursive partitioning in that the same partition options as indicated in FIG. 9 may be repeated at a lower scale until the lowest level (e.g., 4 x 4 level). In some implementations, additional restrictions may be applied to the partitioning scheme of FIG. 9. In the implementation of FIG. 9, rectangular partitions (e.g., 1:2/2:1 rectangular partitions) may be allowed but they may not be allowed to be recursive, whereas square partitions are allowed to be recursive. The partitioning following FIG. 9 with recursion, if needed, generates a final set of coding blocks. A coding tree depth may be further defined to indicate the splitting depth from the root node or root block. For example, the coding tree depth for the root node or root block, e.g. a 64 x 64 block, may be set to 0, and after the root block is further split once following FIG. 9, the coding tree depth is increased by 1. The maximum or

deepest level from 64 x 64 base block to a minimum partition of 4 x 4 would be 4 (starting from level 0) for the scheme above. Such partitioning scheme may apply to one or more of the color channels. Each color channel may be partitioned independently following the scheme of FIG. 9 (e.g., partitioning pattern or option among the predefined patterns may be independently determined for each of the color channels at each hierarchical level).

Alternatively, two or more of the color channels may share the same hierarchical pattern tree of FIG. 9 (e.g., the same partitioning pattern or option among the predefined patterns may be chosen for the two or more color channels at each hierarchical level).

[0129] FIG. 10 shows another example predefined partitioning pattern allowing recursive partitioning to form a partitioning tree. As shown in FIG. 10, an example 10-way partitioning structure or pattern may be predefined. The root block may start at a predefined level (e.g. from a base block at 128 x 128 level, or 64 x 64 level). The example partitioning structure of FIG. 10 includes various 2:1/1:2 and 4:1/1:4 rectangular partitions. The partition types with 3 sub-partitions indicated 1002, 1004, 1006, and 1008 in the second row of FIG. 10 may be referred to “T-type” partitions. The “T-Type” partitions 1002, 1004, 1006, and 1008 may be referred to as Left T-Type, Top T-Type, Right T-Type and Bottom T-Type. In some example implementations, none of the rectangular partitions of FIG. 10 is allowed to be further subdivided. A coding tree depth may be further defined to indicate the splitting depth from the root node or root block. For example, the coding tree depth for the root node or root block, e.g., a 128 x 128 block, may be set to 0, and after the root block is further split once following FIG. 10, the coding tree depth is increased by 1. In some implementations, only the all-square partitions in 1010 may be allowed for recursive partitioning into the next level of the partitioning tree following pattern of FIG. 10. In other words, recursive partitioning may not be allowed for the square partitions within the T-type patterns 1002, 1004, 1006, and 1008. The partitioning procedure following FIG. 10 with recursion, if needed, generates a final set of coding blocks. Such scheme may apply to one or more of the color channels. In some implementations, more flexibility may be added to the use of partitions below 8 x 8 level. For example, 2 x 2 chroma inter prediction may be used in certain cases.

[0130] In some other example implementations for coding block partitioning, a quadtree structure may be used for splitting a base block or an intermediate block into quadtree partitions. Such quadtree splitting may be applied hierarchically and recursively to any square shaped partitions. Whether a base block or an intermediate block or partition is further quadtree split may be adapted to various local characteristics of the base block or intermediate block/partition. Quadtree partitioning at picture boundaries may be further

adapted. For example, implicit quadtree split may be performed at picture boundary so that a block will keep quadtree splitting until the size fits the picture boundary.

[0131] In some other example implementations, a hierarchical binary partitioning from a base block may be used. For such a scheme, the base block or an intermediate level block may be partitioned into two partitions. A binary partitioning may be either horizontal or vertical. For example, a horizontal binary partitioning may split a base block or intermediate block into equal right and left partitions. Likewise, a vertical binary partitioning may split a base block or intermediate block into equal upper and lower partitions. Such binary partitioning may be hierarchical and recursive. Decision may be made at each of the base block or intermediate block whether the binary partitioning scheme should continue, and if the scheme does continue further, whether a horizontal or vertical binary partitioning should be used. In some implementations, further partitioning may stop at a predefined lowest partition size (in either one or both dimensions). Alternatively, further partitioning may stop once a predefined partitioning level or depth from the base block is reached. In some implementations, the aspect ratio of a partition may be restricted. For example, the aspect ratio of a partition may not be smaller than 1:4 (or larger than 4:1). As such, a vertical strip partition with vertical to horizontal aspect ratio of 4:1, may only be further binary partitioned vertically into an upper and lower partitions each having a vertical to horizontal aspect ratio of 2:1.

[0132] In yet some other examples, a ternary partitioning scheme may be used for partitioning a base block or any intermediate block, as shown in FIG. 13. The ternary pattern may be implemented vertical, as shown in 1302 of FIG. 13, or horizontal, as shown in 1304 of FIG. 13. While the example split ratio in FIG. 13, either vertically or horizontally, is shown as 1:2:1, other ratios may be predefined. In some implementations, two or more different ratios may be predefined. Such ternary partitioning scheme may be used to complement the quadtree or binary partitioning structures in that such triple-tree partitioning is capable of capturing objects located in block center in one contiguous partition while quadtree and binary-tree are always splitting along block center and thus would split the object into separate partitions. In some implementations, the width and height of the partitions of the example triple trees are always power of 2 to avoid additional transforms.

[0133] The above partitioning schemes may be combined in any manner at different partitioning levels. As one example, the quadtree and the binary partitioning schemes described above may be combined to partition a base block into a quadtree-binary-tree (QTBT) structure. In such a scheme, a base block or an intermediate block/partition may be

either quadtree split or binary split, subject to a set of predefined conditions, if specified. A particular example is illustrated in FIG. 14. In the example of FIG. 14, a base block is first quadtree split into four partitions, as shown by 1402, 1404, 1406, and 1408. Thereafter, each of the resulting partitions is either quadtree partitioned into four further partitions (such as 1408), or binarily split into two further partitions (either horizontally or vertically, such as 1402 or 1406, both being symmetric, for example) at the next level, or non-split (such as 1404). Binary or quadtree splitting may be allowed recursively for square shaped partitions, as shown by the overall example partition pattern of 1410 and the corresponding tree structure/representation in 1420, in which the solid lines represent quadtree splitting, and the dashed lines represent binary splitting. Flags may be used for each binary splitting node (non-leaf binary partitions) to indicate whether the binary splitting is horizontal or vertical. For example, as shown in 1420, consistent with the partitioning structure of 1410, flag “0” may represent horizontal binary splitting, and flag “1” may represent vertical binary splitting. For the quadtree-split partition, there is no need to indicate the splitting type since quadtree splitting always splits a block or a partition both horizontally and vertically to produce 4 sub-blocks/partitions with an equal size. In some implementations, flag “1” may represent horizontal binary splitting, and flag “0” may represent vertical binary splitting.

[0134] In some example implementations of the QTBT, the quadtree and binary splitting ruleset may be represented by the following predefined parameters and the corresponding functions associated therewith:

- CTU size: the root node size of a quadtree (size of a base block)
- *MinQTSIZE*: the minimum allowed quadtree leaf node size
- *MaxBTSIZE*: the maximum allowed binary tree root node size
- *MaxBTDepth*: the maximum allowed binary tree depth
- *MinBTSIZE*: the minimum allowed binary tree leaf node size

In some example implementations of the QTBT partitioning structure, the CTU size may be set as 128×128 luma samples with two corresponding 64×64 blocks of chroma samples (when an example chroma sub-sampling is considered and used), the *MinQTSIZE* may be set as 16×16 , the *MaxBTSIZE* may be set as 64×64 , the *MinBTSIZE* (for both width and height) may be set as 4×4 , and the *MaxBTDepth* may be set as 4. The quadtree partitioning may be applied to the CTU first to generate quadtree leaf nodes. The quadtree leaf nodes may have a size from its minimum allowed size of 16×16 (i.e., the *MinQTSIZE*) to 128×128 (i.e., the CTU size). If a node is 128×128 , it will not be first split by the binary tree since the size exceeds the *MaxBTSIZE* (i.e., 64×64). Otherwise, nodes which do not exceed *MaxBTSIZE*

could be partitioned by the binary tree. In the example of FIG. 14, the base block is 128×128 . The basic block can only be quadtree split, according to the predefined ruleset. The base block has a partitioning depth of 0. Each of the resulting four partitions are 64×64 , not exceeding *MaxBTSIZE*, may be further quadtree or binary-tree split at level 1. The process continues. When the binary tree depth reaches *MaxBTDepth* (i.e., 4), no further splitting may be considered. When the binary tree node has width equal to *MinBTSIZE* (i.e., 4), no further horizontal splitting may be considered. Similarly, when the binary tree node has height equal to *MinBTSIZE*, no further vertical splitting is considered.

[0135] In some example implementations, the QTBT scheme above may be configured to support a flexibility for the luma and chroma to have the same QTBT structure or separate QTBT structures. For example, for P and B slices, the luma and chroma CTBs in one CTU may share the same QTBT structure. However, for I slices, the luma CTBs maybe partitioned into CBs by a QTBT structure, and the chroma CTBs may be partitioned into chroma CBs by another QTBT structure. This means that a CU may be used to refer to different color channels in an I slice, e.g., the I slice may consist of a coding block of the luma component or coding blocks of two chroma components, and a CU in a P or B slice may consist of coding blocks of all three colour components.

[0136] In some other implementations, the QTBT scheme may be supplemented with ternary scheme described above. Such implementations may be referred to as multi-type-tree (MTT) structure. For example, in addition to binary splitting of a node, one of the ternary partition patterns of FIG. 13 may be chosen. In some implementations, only square nodes may be subject to ternary splitting. An additional flag may be used to indicate whether a ternary partitioning is horizontal or vertical.

[0137] The design of two-level or multi-level tree such as the QTBT implementations and QTBT implementations supplemented by ternary splitting may be mainly motivated by complexity reduction. Theoretically, the complexity of traversing a tree is T^D , where T denotes the number of split types, and D is the depth of tree. A tradeoff may be made by using multiple types (T) while reducing the depth (D).

[0138] In some implementations, a CB may be further partitioned. For example, a CB may be further partitioned into multiple prediction blocks (PBs) for purposes of intra or inter-frame prediction during coding and decoding processes. In other words, a CB may be further divided into different subpartitions, where individual prediction decision/configuration may be made. In parallel, a CB may be further partitioned into a plurality of transform blocks (TBs) for purposes of delineating levels at which transform or

inverse transform of video data is performed. The partitioning scheme of a CB into PBs and TBs may or may not be the same. For example, each partitioning scheme may be performed using its own procedure based on, for example, the various characteristics of the video data. The PB and TB partitioning schemes may be independent in some example implementations. The PB and TB partitioning schemes and boundaries may be correlated in some other example implementations. In some implementations, for example, TBs may be partitioned after PB partitions, and in particular, each PB, after being determined following partitioning of a coding block, may then be further partitioned into one or more TBs. For example, in some implementations, a PB may be split into one, two, four, or other number of TBs.

[0139] In some implementations, for partitioning of a base block into coding blocks and further into prediction blocks and/or transform blocks, the luma channel and the chroma channels may be treated differently. For example, in some implementations, partitioning of a coding block into prediction blocks and/or transform blocks may be allowed for the luma channel, whereas such partitioning of a coding block into prediction blocks and/or transform blocks may not be allowed for the chroma channel(s). In such implementations, transform and/or prediction of luma blocks thus may be performed only at the coding block level. For another example, minimum transform block size for luma channel and chroma channel(s) may be different, e.g., coding blocks for luma channel may be allowed to be partitioned into smaller transform and/or prediction blocks than the chroma channels. For yet another example, the maximum depth of partitioning of a coding block into transform blocks and/or prediction blocks may be different between the luma channel and the chroma channels, e.g., coding blocks for luma channel may be allowed to be partitioned into deeper transform and/or prediction blocks than the chroma channel(s). For a specific example, luma coding blocks may be partitioned into transform blocks of multiple sizes that can be represented by a recursive partition going down by up to 2 levels, and transform block shapes such as square, 2:1/1:2, and 4:1/1:4 and transform block size from 4 x 4 to 64 x 64 may be allowed. For chroma blocks, however, only the largest possible transform blocks specified for the luma blocks may be allowed.

[0140] In some example implementations for partitioning of a coding block into PBs, the depth, the shape, and/or other characteristics of the PB partitioning may depend on whether the PB is intra or inter coded.

[0141] The partitioning of a coding block (or a prediction block) into transform blocks may be implemented in various example schemes, including but not limited to quadtree splitting and predefined pattern splitting, recursively or non-recursively, and with

additional consideration for transform blocks at the boundary of the coding block or prediction block. In general, the resulting transform blocks may be at different split levels, may not be of the same size, and may not need to be square in shape (e.g., they can be rectangular with some allowed sizes and aspect ratios). Further examples are described in further detail below in relation to FIGs. 15, 16 and 17.

[0142] In some other implementations, however, the CBs obtained via any of the partitioning schemes above may be used as a basic or smallest coding block for prediction and/or transform. In other words, no further splitting is performed for perform inter-prediction/intra-prediction purposes and/or for transform purposes. For example, CBs obtained from the QTBT scheme above may be directly used as the units for performing predictions. Specifically, such a QTBT structure removes the concepts of multiple partition types, i.e. it removes the separation of the CU, PU and TU, and supports more flexibility for CU/CB partition shapes as described above. In such QTBT block structure, a CU/CB can have either a square or rectangular shape. The leaf nodes of such QTBT are used as units for prediction and transform processing without any further partitioning. This means that the CU, PU and TU have the same block size in such example QTBT coding block structure.

[0143] The various CB partitioning schemes above and the further partitioning of CBs into PBs and/or TBs (including no PB/TB partitioning) may be combined in any manner. The following particular implementations are provided as non-limiting examples.

[0144] A specific example implementation of coding block and transform block partitioning is described below. In such an example implementation, a base block may be split into coding blocks using recursive quadtree splitting, or a predefined splitting pattern described above (such as those in FIG. 9 and FIG. 10). At each level, whether further quadtree splitting of a particular partition should continue may be determined by local video data characteristics. The resulting CBs may be at various quadtree splitting levels, and of various sizes. The decision on whether to code a picture area using inter-picture (temporal) or intra-picture (spatial) prediction may be made at the CB level (or CU level, for all three-color channels). Each CB may be further split into one, two, four, or other number of PBs according to predefined PB splitting type. Inside one PB, the same prediction process may be applied and the relevant information may be transmitted to the decoder on a PB basis. After obtaining the residual block by applying the prediction process based on the PB splitting type, a CB can be partitioned into TBs according to another quadtree structure similar to the coding tree for the CB. In this particular implementation, a CB or a TB may but does not have to be limited to square shape. Further in this particular example, a PB may be square or

rectangular shape for an inter-prediction and may only be square for intra-prediction. A coding block may be split into, e.g., four square-shaped TBs. Each TB may be further split recursively (using quadtree split) into smaller TBs, referred to as Residual Quadtree (RQT).

[0145] Another example implementation for partitioning of a base block into CBs, PBs and or TBs is further described below. For example, rather than using a multiple partition unit types such as those shown in FIG. 9 or FIG. 10, a quadtree with nested multi-type tree using binary and ternary splits segmentation structure (e.g., the QTBT or QTBT with ternary splitting as described above) may be used. The separation of the CB, PB and TB (i.e., the partitioning of CB into PBs and/or TBs, and the partitioning of PBs into TBs) may be abandoned except when needed for CBs that have a size too large for the maximum transform length, where such CBs may need further splitting. This example partitioning scheme may be designed to support more flexibility for CB partition shapes so that the prediction and transform can both be performed on the CB level without further partitioning. In such a coding tree structure, a CB may have either a square or rectangular shape. Specifically, a coding tree block (CTB) may be first partitioned by a quadtree structure. Then the quadtree leaf nodes may be further partitioned by a nested multi-type tree structure. An example of the nested multi-type tree structure using binary or ternary splitting is shown in FIG. 11. Specifically, the example multi-type tree structure of FIG. 11 includes four splitting types, referred to as vertical binary splitting (SPLIT_BT_VER) (1102), horizontal binary splitting (SPLIT_BT_HOR) (1104), vertical ternary splitting (SPLIT_TT_VER) (1106), and horizontal ternary splitting (SPLIT_TT_HOR) (1108). The CBs then correspond to leaves of the multi-type tree. In this example implementation, unless the CB is too large for the maximum transform length, this segmentation is used for both prediction and transform processing without any further partitioning. This means that, in most cases, the CB, PB and TB have the same block size in the quadtree with nested multi-type tree coding block structure. The exception occurs when maximum supported transform length is smaller than the width or height of the colour component of the CB. In some implementations, in addition to the binary or ternary splitting, the nested patterns of FIG. 11 may further include quadtree splitting.

[0146] One specific example for the quadtree with nested multi-type tree coding block structure of block partition (including quadtree, binary, and ternary splitting options) for one base block is shown in FIG. 12. In more detail, FIG. 12 shows that the base block 1200 is quadtree split into four square partitions 1202, 1204, 1206, and 1208. Decision to further use the multi-type tree structure of FIG. 11 and quadtree for further splitting is made

for each of the quadtree-split partitions. In the example of FIG. 12, partition 1204 is not further split. Partitions 1202 and 1208 each adopt another quadtree split. For partition 1202, the second level quadtree-split top-left, top-right, bottom-left, and bottom-right partitions adopts third level splitting of quadtree, horizontal binary splitting 1104 of FIG. 11, non-splitting, and horizontal ternary splitting 1108 of FIG. 11, respectively. Partition 1208 adopts another quadtree split, and the second level quadtree-split top-left, top-right, bottom-left, and bottom-right partitions adopts third level splitting of vertical ternary splitting 1106 of FIG. 11, non-splitting, non-splitting, and horizontal binary splitting 1104 of FIG. 11, respectively. Two of the subpartitions of the third-level top-left partition of 1208 are further split according to horizontal binary splitting 1104 and horizontal ternary splitting 1108 of FIG. 11, respectively. Partition 1206 adopts a second level split pattern following the vertical binary splitting 1102 of FIG. 11 into two partitions which are further split in a third-level according to horizontal ternary splitting 1108 and vertical binary splitting 1102 of the FIG. 11. A fourth level splitting is further applied to one of them according to horizontal binary splitting 1104 of FIG. 11.

[0147] For the specific example above, the maximum luma transform size may be 64×64 and the maximum supported chroma transform size could be different from the luma at, e.g., 32×32. Even though the example CBs above in FIG. 12 are generally not further split into smaller PBs and/or TBs, when the width or height of the luma coding block or chroma coding block is larger than the maximum transform width or height, the luma coding block or chroma coding block may be automatically split in the horizontal and/or vertical direction to meet the transform size restriction in that direction.

[0148] In the specific example for partitioning of a base block into CBs above, and as described above, the coding tree scheme may support the ability for the luma and chroma to have a separate block tree structure. For example, for P and B slices, the luma and chroma CTBs in one CTU may share the same coding tree structure. For I slices, for example, the luma and chroma may have separate coding block tree structures. When separate block tree structures are applied, luma CTB may be partitioned into luma CBs by one coding tree structure, and the chroma CTBs are partitioned into chroma CBs by another coding tree structure. This means that a CU in an I slice may consist of a coding block of the luma component or coding blocks of two chroma components, and a CU in a P or B slice always consists of coding blocks of all three colour components unless the video is monochrome.

[0149] When a coding block is further partitioned into multiple transform blocks, the transform blocks therein may be order in the bitstream following various order or scanning

manners. Example implementations for partitioning a coding block or prediction block into transform blocks, and a coding order of the transform blocks are described in further detail below. In some example implementations, as described above, a transform partitioning may support transform blocks of multiple shapes, e.g., 1:1 (square), 1:2/2:1, and 1:4/4:1, with transform block sizes ranging from, e.g., 4×4 to 64×64 . In some implementations, if the coding block is smaller than or equal to 64×64 , the transform block partitioning may only apply to luma component, such that for chroma blocks, the transform block size is identical to the coding block size. Otherwise, if the coding block width or height is greater than 64, then both the luma and chroma coding blocks may be implicitly split into multiples of $\min(W, 64) \times \min(H, 64)$ and $\min(W, 32) \times \min(H, 32)$ transform blocks, respectively.

[0150] In some example implementations of transform block partitioning, for both intra and inter coded blocks, a coding block may be further partitioned into multiple transform blocks with a partitioning depth up to a predefined number of levels (e.g., 2 levels). The transform block partitioning depth and sizes may be related. For some example implementations, a mapping from the transform size of the current depth to the transform size of the next depth is shown in the following in Table 1.

Table 1: Transform partition size setting

Transform Size of Current Depth	Transform Size of Next Depth
TX_4X4	TX_4X4
TX_8X8	TX_4X4
TX_16X16	TX_8X8
TX_32X32	TX_16X16
TX_64X64	TX_32X32
TX_4X8	TX_4X4
TX_8X4	TX_4X4
TX_8X16	TX_8X8
TX_16X8	TX_8X8
TX_16X32	TX_16X16
TX_32X16	TX_16X16
TX_32X64	TX_32X32
TX_64X32	TX_32X32
TX_4X16	TX_4X8
TX_16X4	TX_8X4
TX_8X32	TX_8X16
TX_32X8	TX_16X8
TX_16X64	TX_16X32
TX_64X16	TX_32X16

[0151] Based on the example mapping of Table 1, for 1:1 square block, the next level transform split may create four 1:1 square sub-transform blocks. Transform partition may stop, for example, at 4 x 4. As such, a transform size for a current depth of 4 x 4 corresponds to the same size of 4 x 4 for the next depth. In the example of Table 1, for 1:2/2:1 non-square block, the next level transform split may create two 1:1 square sub-transform blocks, whereas for 1:4/4:1 non-square block, the next level transform split may create two 1:2/2:1 sub-transform blocks.

[0152] In some example implementations, for luma component of an intra coded block, additional restriction may be applied with respect to transform block partitioning. For example, for each level of transform partitioning, all the sub-transform blocks may be restricted to having equal size. For example, for a 32 x 16 coding block, level 1 transform split creates two 16 x 16 sub-transform blocks, level 2 transform split creates eight 8 x 8 sub-transform blocks. In other words, the second level splitting must be applied to all first level sub blocks to keep the transform units at equal sizes. An example of the transform block partitioning for intra coded square block following Table 1 is shown in FIG. 15, together with coding order illustrated by the arrows. Specifically, 1502 shows the square coding block. A first-level split into 4 equal sized transform blocks according to Table 1 is shown in 1504 with coding order indicated by the arrows. A second-level split of all of the first-level equal sized blocks into 16 equal sized transform blocks according to Table 1 is shown in 1506 with coding order indicated by the arrows.

[0153] In some example implementations, for luma component of inter coded block, the above restriction for intra coding may not be applied. For example, after the first level of transform splitting, any one of sub-transform block may be further split independently with one more level. The resulting transform blocks thus may or may not be of the same size. An example split of an inter coded block into transform locks with their coding order is show in FIG. 16. In the Example of FIG. 16, the inter coded block 1602 is split into transform blocks at two levels according to Table 1. At the first level, the inter coded block is split into four transform blocks of equal size. Then only one of the four transform blocks (not all of them) is further split into four sub-transform blocks, resulting in a total of 7 transform blocks having two different sizes, as shown by 1604. The example coding order of these 7 transform blocks is shown by the arrows in 1604 of FIG. 16.

[0154] In some example implementations, for chroma component(s), some additional restriction for transform blocks may apply. For example, for chroma component(s) the

transform block size can be as large as the coding block size, but not smaller than a predefined size, e.g., 8×8 .

[0155] In some other example implementations, for the coding block with either width (W) or height (H) being greater than 64, both the luma and chroma coding blocks may be implicitly split into multiples of $\min(W, 64) \times \min(H, 64)$ and $\min(W, 32) \times \min(H, 32)$ transform units, respectively. Here, in the present disclosure, a “min (a , b)” may return a smaller value between a and b .

[0156] FIG. 17 further shows another alternative example scheme for partitioning a coding block or prediction block into transform blocks. As shown in FIG. 17, instead of using recursive transform partitioning, a predefined set of partitioning types may be applied to a coding block according to a transform type of the coding block. In the particular example shown in FIG. 17, one of the 6 example partitioning types may be applied to split a coding block into various number of transform blocks. Such scheme of generating transform block partitioning may be applied to either a coding block or a prediction block.

[0157] In more detail, the partitioning scheme of FIG. 17 provides up to 6 example partition types for any given transform type (transform type refers to the type of, e.g., primary transform, such as ADST and others). In this scheme, every coding block or prediction block may be assigned a transform partition type based on, for example, a rate-distortion cost. In an example, the transform partition type assigned to the coding block or prediction block may be determined based on the transform type of the coding block or prediction block. A particular transform partition type may correspond to a transform block split size and pattern, as shown by the 6 transform partition types illustrated in FIG. 17. A correspondence relationship between various transform types and the various transform partition types may be predefined. An example is shown below with the capitalized labels indicating the transform partition types that may be assigned to the coding block or prediction block based on rate distortion cost:

[0158] • PARTITION_NONE: Assigns a transform size that is equal to the block size.

[0159] • PARTITION_SPLIT: Assigns a transform size that is $\frac{1}{2}$ the width of the block size and $\frac{1}{2}$ the height of the block size.

[0160] • PARTITION_HORZ: Assigns a transform size with the same width as the block size and $\frac{1}{2}$ the height of the block size.

[0161] • PARTITION_VERT: Assigns a transform size with $\frac{1}{2}$ the width of the block size and the same height as the block size.

[0162] • PARTITION_HORZ4: Assigns a transform size with the same width as the block size and $\frac{1}{4}$ the height of the block size.

[0163] • PARTITION_VERT4: Assigns a transform size with $\frac{1}{4}$ the width of the block size and the same height as the block size.

[0164] In the example above, the transform partition types as shown in FIG. 17 all contain uniform transform sizes for the partitioned transform blocks. This is a mere example rather than a limitation. In some other implementations, mixed transform blocks sizes may be used for the partitioned transform blocks in a particular partition type (or pattern).

[0165] The PBs (or CBs, also referred to as PBs when not being further partitioned into prediction blocks) obtained from any of the partitioning schemes above may then become the individual blocks for coding via either intra or inter predictions. For inter-prediction for a current PB, a residual between the current block and a prediction block may be generated, coded, and included in the coded bitstream.

[0166] Inter-prediction may be implemented, for example, in a single-reference mode or a compound-reference mode. In some implementations, a skip flag may be first included in the bitstream for a current block (or at a higher level) to indicate whether the current block is inter-coded and is not to be skipped. If the current block is inter-coded, then another flag may be further included in the bitstream as a signal to indicate whether the single-reference mode or compound-reference mode is used for the prediction of the current block. For the single-reference mode, one reference block may be used to generate the prediction block for the current block. For the compound-reference mode, two or more reference blocks may be used to generate the prediction block by, for example, weighted average. The compound-reference mode may be referred as more-than-one-reference mode, two-reference mode, or multiple-reference mode. The reference block or reference blocks may be identified using reference frame index or indices and additionally using corresponding motion vector or motion vectors which indicate shift(s) between the reference block(s) and the current blocks in location, e.g., in horizontal and vertical pixels. For example, the inter-prediction block for the current block may be generated from a single-reference block identified by one motion vector in a reference frame as the prediction block in the single-reference mode, whereas for the compound-reference mode, the prediction block may be generated by a weighted average of two reference blocks in two reference frames indicated by two reference frame indices and two corresponding motion vectors. The motion vector(s) may be coded and included in the bitstream in various manners.

[0167] In some implementations, an encoding or decoding system may maintain a decoded picture buffer (DPB). Some images/pictures may be maintained in the DPB waiting for being displayed (in a decoding system) and some images/pictures in the DPB may be used as reference frames to enable inter-prediction (in a decoding system or encoding system). In some implementations, the reference frames in the DPB may be tagged as either short-term references or long-term references for a current image being encoded or decoded. For example, short-term reference frames may include frames that are used for inter-prediction for blocks in a current frame or in a predefined number (e.g., 2) of closest subsequent video frames to the current frame in a decoding order. The long-term reference frames may include frames in the DPB that can be used to predict image blocks in frames that are more than the predefined number of frames away from the current frame in the order of decoding. Information about such tags for short and long-term reference frames may be referred to as Reference Picture Set (RPS) and may be added to a header of each frame in the encoded bitstream. Each frame in the encoded video stream may be identified by a Picture Order Counter (POC), which is numbered according to playback sequence in an absolute manner or relevant to a picture group starting from, for example, an I-frame.

[0168] In some example implementations, one or more reference picture lists containing identification of short-term and long-term reference frames for inter-prediction may be formed based on the information in the RPS. For example, a single picture reference list may be formed for uni-directional inter-prediction, denoted as L0 reference (or reference list 0) whereas two picture referenced lists may be formed for bi-direction inter-prediction, denoted as L0 (or reference list 0) and L1 (or reference list 1) for each of the two prediction directions. The reference frames included in the L0 and L1 lists may be ordered in various predetermined manners. The lengths of the L0 and L1 lists may be signaled in the video bitstream. Uni-directional inter-prediction may be either in the single-reference mode, or in the compound-reference mode when the multiple references for the generation of prediction block by weighted average in the compound prediction mode are on a same side of the block to be predicted. Bi-directional inter-prediction may only be compound mode in that bi-directional inter-prediction involves at least two reference blocks.

[0169] In some implementations, a merge mode (MM) for inter-prediction may be implemented. Generally, for the merge mode, the motion vector in single-reference prediction or one or more of the motion vectors in compound-reference prediction for the current PB may be derived from other motion vector(s) rather than being computed and signaled independently. For example, in an encoding system, the current motion vector(s) for

the current PB may be represented by difference(s) between the current motion vector(s) and other one or more already encoded motion vectors (referred to as reference motion vectors). Such difference(s) in motion vector(s) rather than the entirety of the current motion vector(s) may be encoded and included in the bit stream and may be linked to the reference motion vector(s). Correspondingly in a decoding system, the motion vector(s) corresponding to the current PB may be derived based on the decoded motion vector difference(s) and decoded reference motion vector(s) linked therewith. As a specific form of the general merge mode (MM) inter-prediction, such inter-prediction based on motion vector difference(s) may be referred to as Merge Mode with Motion Vector Difference (MMVD). MM in general or MMVD in particular may thus be implemented to leverage correlations between motion vectors associated with different PBs to improve coding efficiency. For example, neighboring PBs may have similar motion vectors and thus the MVD may be small and can be efficiently coded. For another example, motion vectors may correlate temporally (between frames) for similarly located/positioned blocks in space.

[0170] In some example implementations, an MM flag may be included in a bitstream during an encoding process for indicating whether the current PB is in a merge mode. Additionally, or alternatively, an MMVD flag may be included during the encoding process and signaled in the bitstream to indicate whether the current PB is in an MMVD mode. The MM and/or MMVD flags or indicators may be provided at the PB level, the CB level, the CU level, the CTB level, the CTU level, slice level, picture level, and the like. For a particular example, both an MM flag and an MMVD flag may be included for a current CU, and the MMVD flag may be signalled right after the skip flag and the MM flag to specify whether the MMVD mode is used for the current CU.

[0171] In some example implementations of MMVD, a list of reference motion vector (RMV) or MV predictor candidates for motion vector prediction may be formed for a block being predicted. The list of RMV candidates may contain a predetermined number (e.g., 2) of MV predictor candidate blocks whose motion vectors may be used for predicting the current motion vector. The RMV candidate blocks may include blocks selected from neighboring blocks in the same frame and/or temporal blocks (e.g., identically located blocks in proceeding or subsequent frame of the current frame). These options represent blocks at spatial or temporal locations relative to the current block that are likely to have similar or identical motion vectors to the current block. The size of the list of MV predictor candidates may be predetermined. For example, the list may contain two or more candidates. To be on the list of RMV candidates, a candidate block, for example, may be required to have the same

reference frame (or frames) as the current block, must exist (e.g., when the current block is near the edge of the frame, a boundary check needs to be performed), and must be already encoded during an encoding process, and/or already decoded during a decoding process. In some implementations, the list of merge candidates may be first populated with spatially neighboring blocks (scanned in particular predefined order) if available and meeting the conditions above, and then the temporal blocks if space is still available in the list. The neighboring RMV candidate blocks, for example, may be selected from left and top blocks of the current block. The list of RMV predictor candidates may be dynamically formed at various levels (sequence, picture, frame, slice, superblock, etc.) as a Dynamic Reference List (DRL). DRL may be signaled in the bitstream.

[0172] In some implementations, an actual MV predictor candidate being used as a reference motion vector for predicting a motion vector of the current block may be signaled. In the case that the RMV candidate list contains two candidates, a one-bit flag, referred to as merge candidate flag may be used to indicate the selection of the reference merge candidate. For a current block being predicted in compound mode, each of the multiple motion vectors predicted using a MV predictor may be associated with reference motion vector from the merge candidate list. The encoder may determine which of the RMV candidate more closely predicts a current coding block and signal the selection as an index into the DRL.

[0173] In some example implementations of MMVD, after a RMV candidate is selected and used as base motion vector predictor for a motion vector to be predicted, a motion vector difference (MVD or a delta MV, representing the difference between the motion vector to be predicted and the reference candidate motion vector) may be calculated in the encoding system. Such MVD may include information representing a magnitude of MV difference and a direction of the MV difference, both of which may be signaled in the bitstream. The motion difference magnitude and the motion difference direction may be signaled in various manners.

[0174] In some example implementations of the MMVD, a distance index may be used to specify magnitude information of the motion vector difference and to indicate one of a set of pre-defined offsets representing predefined motion vector difference from the starting point (the reference motion vector). An MV offset according to the signaled index may then be added to either horizontal component or vertical component of the starting (reference) motion vector. Whether the horizontal or vertical component of the reference motion vector should be offset may be determined by a directional information of the MVD. An example predefined relation between distance index and predefined offsets is specified in Table 2.

Table 2 – Example relation of distance index and pre-defined MV offset

Distance Index	0	1	2	3	4	5	6	7
Offset (in unit of luma sample)	1/4	1/2	1	2	4	8	16	32

[0175] In some example implementations of the MMVD, a direction index may be further signaled and used to represent a direction of the MVD relative to the reference motion vector. In some implementations, the direction may be restricted to either one of the horizontal and vertical directions. An example 2-bit direction index is shown in Table 3. In the example of Table 3, the interpretation of the MVD could be variant according to the information of the starting/reference MVs. For example, when the starting/reference MV corresponds to a uni-prediction block or corresponds to a bi-prediction block with both reference frame lists point to the same side of the current picture (i.e. POCs of the two reference pictures are both larger than the POC of the current picture, or are both smaller than the POC of the current picture), the sign in Table 3 may specify the sign (direction) of MV offset added to the starting/reference MV. When the starting/reference MV corresponds to a bi-prediction block with the two reference pictures at different sides of the current picture (i.e. the POC of one reference picture is larger than the POC of the current picture, and the POC of the other reference picture is smaller than the POC of the current picture), and a difference between the reference POC in picture reference list 0 and the current frame is greater than that between the reference POC in picture reference list 1 and the current frame, the sign in Table 3 may specify the sign of MV offset added to the reference MV corresponding to the reference picture in picture reference list 0, and the sign for the offset of the MV corresponding to the reference picture in picture reference list 1 may have an opposite value (opposite sign for the offset). Otherwise, if the difference between the reference POC in picture reference list 1 and the current frame is greater than that between the reference POC in picture reference list 0 and the current frame, the sign in Table 3 may then specify the sign of MV offset added to the reference MV associated with the picture reference list 1 and the sign for the offset to the reference MV associated with the picture reference list 0 has opposite value.

Table 3 – Example implementations for sign of MV offset specified by direction index

Direction IDX	00	01	10	11
x-axis (horizontal)	+	–	N/A	N/A

y-axis (vertical)	N/A	N/A	+	–
-------------------	-----	-----	---	---

[0176] In some example implementations, the MVD may be scaled according to the difference of POCs in each direction. If the differences of POCs in both lists are the same, no scaling is needed. Otherwise, if the difference of POC in reference list 0 is larger than the one of reference list 1, the MVD for reference list 1 is scaled. If the POC difference of reference list 1 is greater than list 0, the MVD for list 0 may be scaled in the same way. If the starting MV is uni-predicted, the MVD is added to the available or reference MV.

[0177] In some example implementations of MVD coding and signaling for bi-directional compound prediction, in addition or alternative to separately coding and signaling the two MVDs, a symmetric MVD coding may be implemented such that only one MVD needs signaling and the other MVD may be derived from the signaled MVD. In such implementations, motion information including reference picture indices of both list-0 and list-1 is signaled. However, only MVD associated with, e.g., reference list-0 is signaled and MVD associated with reference list-1 is not signaled but derived. Specifically, at a slice level, a flag may be included in the bitstream, referred to as “mvd_l1_zero_flag,” for indicating whether the reference list-1 is not signaled in the bitstream. If this flag is 1, indicating that reference list-1 is equal to zero (and thus not signaled), then a bi-directional-prediction flag, referred to as “BiDirPredFlag” may be set to 0, meaning that there is no bi-directional-prediction. Otherwise, if mvd_l1_zero_flag is zero, if the nearest reference picture in list-0 and the nearest reference picture in list-1 form a forward and backward pair of reference pictures or a backward and forward pair of reference pictures, BiDirPredFlag may be set to 1, and both list-0 and list-1 reference pictures are short-term reference pictures. Otherwise BiDirPredFlag is set to 0. BiDirPredFlag of 1 may indicate that a symmetrical mode flag is additionally signalled in the bitstream. The decoder may extract the symmetrical mode flag from the bitstream when BiDirPredFlag is 1. The symmetrical mode flag, for example, may be signaled (if needed) at the CU level and it may indicate whether the symmetrical MVD coding mode is being used for the corresponding CU. When the symmetrical mode flag is 1, it indicates the use of the symmetrical MVD coding mode, and that only reference picture indices of both list-0 and list-1 (referred to as “mvp_l0_flag” and “mvp_l1_flag”) are signaled with MVD associated with the list-0 (referred to as “MVD0”), and that the other motion vector difference, “MVD1”, is to be derived rather than signaled. For example, MVD1 may be derived as -MVD0. As such, only one MVD is signaled in the example symmetrical MVD mode. In some other example implementations for MV

prediction, a harmonized scheme may be used to implement a general merge mode, MMVD, and some other types of MV prediction, for both single-reference mode and compound-reference mode MV prediction. Various syntax elements may be used to signal the manner in which the MV for a current block is predicted.

[0178] For example, for single-reference mode, the following MV prediction modes may be signaled:

[0179] NEARMV – use one of the motion vector predictors (MVP) in the list indicated by a DRL (Dynamic Reference List) index directly without any MVD.

[0180] NEWMV – use one of the motion vector predictors (MVP) in the list signaled by a DRL index as reference and apply a delta to the MVP (e.g., using MVD).

[0181] GLOBALMV – use a motion vector based on frame-level global motion parameters.

[0182] Likewise, for the compound-reference inter-prediction mode using two reference frames corresponding to two MVs to be predicted, the following MV prediction modes may be signaled:

[0183] NEAR_NEARMV – use one of the motion vector predictors (MVP) in the list signaled by a DRL index without MVD for each of the two of MVs to be predicted.

[0184] NEAR_NEWMV – for predicting the first of the two motion vectors, use one of the motion vector predictors (MVP) in the list signaled by a DRL index as reference MV with out MVD; for predicting the second of the two motion vectors, use one of the motion vector predictors (MVP) in the list signaled by a DRL index as reference MV in conjunction with an additionally signaled delta MV (an MVD).

[0185] NEW_NEARMV – for predicting the second of the two motion vectors, use one of the motion vector predictors (MVP) in the list signaled by a DRL index as reference MV with out MVD; for predicting the first of the two motion vectors, use one of the motion vector predictors (MVP) in the list signaled by a DRL index as reference MV in conjunction with an additionally signaled delta MV (an MVD).

[0186] NEW_NEWMV – use one of the motion vector predictors (MVP) in the list signaled by a DRL index as reference MV and use it in conjunction with an additionally signaled delta MV to predict for each of the two MVs.

[0187] GLOBAL_GLOBALMV – use MVs from each reference based on their frame-level global motion parameters.

[0188] The term “NEAR” above thus refers to MV prediction using reference MV without MVD as a general merge mode, whereas the term “NEW” refers to MV prediction

involving using a referend MV and offsetting it with a signaled MVD as in an MMVD mode. For the compound inter-prediction, both the reference base motion vectors and the motion vector deltas above, may be generally different or independent between the two references, even though they may be correlated and such correlation may be leveraged to reduce the amount of information needed for signaling the two motion vector deltas. In such situations, a joint signaling of the two MVDs may be implemented and indicated in the bitstream.

[0189] The dynamic reference list (DRL) above may be used to hold a set of indexed motion vectors that are dynamically maintained and are considered as candidate motion vector predictors.

[0190] In some example implementations, a predefined resolution for the MVD may be allowed. For example, a 1/8-pixel motion vector precision (or accuracy) may be allowed. The MVD described above in the various MV prediction modes may be constructed and signaled in various manners. In some implementations, various syntax elements may be used to signal the motion vector difference(s) above in reference frame list 0 or list 1.

[0191] For example, a syntax element referred to as “mv_joint” may specify which components of the motion vector difference associated therewith are non-zero. For an MVD, this is jointly signaled for all the non-zero components. For example, mv_joint having a value of

0 may indicate that there is no non-zero MVD along either the horizontal or the vertical direction;

1 may indicate that there is non-zero MVD only along the horizontal direction;

2 may indicate that there is non-zero MVD only along the vertical direction;

3 may indicate that there is non-zero MVD along both the horizontal and the vertical directions.

[0192] When the “mv_joint” syntax element for an MVD signals that there is no non-zero MVD component, then no further MVD information may be signaled. However, if the “mv_joint” syntax signals that there is one or two non-zero components, then additional syntax elements may be further signaled for each of the non-zero MVD components as described below.

[0193] For example, a syntax element referred to as “mv_sign” may be used to additionally specify whether the corresponding motion vector difference component is positive or negative.

[0194] For another example, a syntax element referred to as “mv_class” may be used to specify a class of the motion vector difference among a predefined set of classes for the

corresponding non-zero MVD component. The predefined classes for motion vector difference, for example, may be used to divide a contiguous magnitude space of the motion vector difference into non-overlapping ranges with each range corresponding to an MVD class. A signaled MVD class thus indicates the magnitude range of the corresponding MVD component. In the example implementation shown in Table 4 below, a higher class corresponds to motion vector differences having range of a larger magnitude. In Table 4, the symbol (n, m] is used for representing a range of motion vector difference that is greater than n pixels, and smaller than or equal to m pixels.

Table 4: Magnitude class for motion vector difference

MV class	Magnitude of MVD
MV_CLASS_0	(0, 2]
MV_CLASS_1	(2, 4]
MV_CLASS_2	(4, 8]
MV_CLASS_3	(8, 16]
MV_CLASS_4	(16, 32]
MV_CLASS_5	(32, 64]
MV_CLASS_6	(64, 128]
MV_CLASS_7	(128, 256]
MV_CLASS_8	(256, 512]
MV_CLASS_9	(512, 1024]
MV_CLASS_10	(1024, 2048]

[0195] In some other examples, a syntax element referred to as “mv_bit” may be further used to specify an integer part of the offset between the non-zero motion vector difference component and starting magnitude of a correspondingly signaled MV class magnitude range. As such, mv_bit may indicate a magnitude or amplitude of the MVD. The number of bits needed in “mv_bit” for signaling a full range of each MVD class may vary as a function of the MV class. For the example, MV_CLASS 0 and MV_CLASS 1 in the implementation of Table 4 may merely need a single bit to indicate integer pixel offset of 1 or 2 from starting MVD of 0; each higher MV_CLASS in the example implementation of Table 4 may need progressively one more bit for “mv_bit” than the previous MV_CLASS.

[0196] In some other examples, a syntax element referred to as “mv_fr” may be further used to specify first 2 fractional bits of the motion vector difference for a

corresponding non-zero MVD component, whereas a syntax element referred to as “mv_hp” may be used to specify a third fractional bit of the motion vector difference (high resolution bit) for a corresponding non-zero MVD component. The two-bit “mv_fr” essentially provides $\frac{1}{4}$ pixel MVD resolution, whereas the “mv_hp” bit may further provide a $\frac{1}{8}$ -pixel resolution. In some other implementations, more than one “mv_hp” bit may be used to provide MVD pixel resolution finer than $\frac{1}{8}$ pixels. In some example implementations, additional flags may be signaled at one or more of the various levels to indicate whether $\frac{1}{8}$ -pixel or higher MVD resolution is supported. If MVD resolution is not applied to a particular coding unit, then the syntax elements above for the corresponding non-supported MVD resolution may not be signaled.

[0197] In some example implementations above, fractional resolution may be independent of different classes of MVD. In other words, regardless of the magnitude of the motion vector difference, similar options for motion vector resolution may be provided using a predefined number of “mv_fr” and “mv_hp” bits for signaling the fractional MVD of a non-zero MVD component.

[0198] However, in some other example implementations, resolution for motion vector difference in various MVD magnitude classes may be differentiated. Specifically, high resolution MVD for large MVD magnitude of higher MVD classes may not provide statistically significant improvement in compression efficiency. As such, the MVDs may be coded with decreasing resolution (integer pixel resolution or fractional pixel resolution) for larger MVD magnitude ranges, which correspond to higher MVD magnitude classes. Likewise, the MVD may be coded with decreasing resolution (integer pixel resolution or fractional pixel resolution) for larger MVD values in general. Such MVD class-dependent or MVD magnitude-dependent MVD resolution may be generally referred to as adaptive MVD resolution, amplitude-dependent adaptive MVD resolution, or magnitude-dependent MVD resolution. The term “resolution” may be further referred to as “pixel resolution” Adaptive MVD resolution may be implemented in various matter as described by the example implementations below for achieving an overall better compression efficiency. In particular, the reduction of number of signaling bits by aiming at less precise MVD may be greater than the additional bits needed for coding inter-prediction residual as a result of such less precise MVD, due to the statistical observation that treating MVD resolution for large-magnitude or high-class MVD at similar level as that for low-magnitude or low-class MVD in a non-adapted manner may not significantly increase inter-prediction residual coding efficiency for blocks with large-magnitude or high-class MVD. In other words, using higher MVD

resolutions for large-magnitudes or high-class MVD may not produce much coding gain over using lower MVD resolutions.

[0199] In some general example implementations, the pixel resolution or precision for MVD may decrease or may be non-increasing with increasing MVD class. Decreasing pixel resolution for the MVD corresponds to coarser MVD (or larger step from one MVD level to the next). In some implementations, the correspondence between an MVD pixel resolution and MVD class may be specified, predefined, or pre-configured and thus may not need to be signaled in the encode bitstream.

[0200] In some example implementations, the MV classes of Table 3 may each be associated with different MVD pixel resolutions.

[0201] In some example implementations, each MVD class may be associated with a single allowed resolution. In some other implementations, one or more MVD classes may be associated with two or more optional MVD pixel resolutions. A signal in a bitstream for a current MVD component with such an MVD class may thus be followed by an additional signaling for indicating which optional pixel resolution is selected for the current MVD component.

[0202] In some example implementations, the adaptively allowed MVD pixel resolution may include but not limited to 1/64-pel (pixel), 1/32-pel, 1/16-pel, 1/8-pel, 1-4-pel, 1/2-pel, 1-pel, 2-pel, 4-pel...(in descending order of resolution). As such, each one of the ascending MVD classes may be associated with one of these resolutions in a non-ascending manner. In some implementations, an MVD class may be associated with two or more resolutions above and the higher resolution may be lower than or equal to the lower resolution for the preceding MVD class. For example, if the MV_CLASS_3 of Table 4 may be associated with optional 1-pel and 2-pel resolution, then the highest resolution that MV_CLASS_4 of Table 4 could be associated with would be 2-pel. In some other implementations, the highest allowable resolution for an MV class may be higher than the lowest allowable resolution of a preceding (lower) MV class. However, the average of allowed resolution for ascending MV classes may only be non-ascending.

[0203] In some implementations, when fractional pixel resolution higher than 1/8 pel is allowed, the “mv_fr” and “mv_hp” signaling may be correspondingly expanded to more than 3 fractional bits in total.

[0204] In some example implementations, fractional pixel resolution may only be allowed for MVD classes below or equal to a threshold MVD class. For example, fractional pixel resolution may only be allowed for MVD-CLASS 0 and disallowed for all other MV

classes of Table 4. Likewise, fractional pixel resolution may only be allowed for MVD classes below or equal to any one of other MV classes of Table 4. For the other MVD classes above the threshold MVD class, only integer pixel resolutions for MVD is allowed. In Such a manner, fractional resolution signaling such as the one or more of the “mv-fr” and/or “mv-hp” bits may not need be signaled for MVD signaled with an MVD class higher than or equal to the threshold MVD class. For MVD classes having resolution lower than 1 pixel, the number of bits in “mv-bit” signaling may be further reduced. For example, for MV_CLASS_5 in Table 4, the range of MVD pixel offset is (32, 64], thus 5 bits are needed to signal the entire range with 1-pel resolution. However, if MV_CLASS_5 is associated with 2-pel MVD resolution (lower resolution than 1-pixel resolution), then 4 bits rather than 5 bits may be needed for “mv-bit”, and none of “mv-fr” and “mv-hp” needs be signaled following a signaling of “mv_class” as MV-CLASS_5.

[0205] In some example implementations, fractional pixel resolution may only be allowed for MVD with integer value below a threshold integer pixel value. For example, fractional pixel resolution may only be allowed for MVD smaller than 5 pixels. Corresponding to this example, fractional resolution may be allowed for MV_CLASS_0 and MV_CLASS_1 of Table 4 and disallowed for all other MV classes. For another example, fractional pixel resolution may only be allowed for MVD smaller than 7 pixels. Corresponding to this example, fractional resolution may be allowed for MV_CLASS_0 and MV_CLASS_1 of Table 4 (with ranges below 5 pixels) and disallowed for MV_CLASS_3 and higher (with ranges above 5 pixels). For an MVD belonging to MV_CLASS_2, whose pixel range encompasses 5 pixels, fractional pixel resolution for the MVD may or may be allowed depending on the “mv-bit” value. If the “m-bit” value is signaled as 1 or 2 (such that the integer portion of the signaled MVD is 5 or 6, calculated as starting of the pixel range for MV_CLASS_2 with an offset 1 or 2 as indicated by “m-bit”), then fractional pixel resolution may be allowed. Otherwise, if the “mv-bit” value is signaled as 3 or 4 (such that the integer portion of the signaled MVD is 7 or 8), then fractional pixel resolution may not be allowed.

[0206] In some other implementations, for MV classes equal to or higher than a threshold MV class, only a single MVD value may be allowed. For example, such threshold MV class may be MV_CLASS 2. Thus, MV_CLASS_2 and above may only be allowed to have a single MVD value and without fractional pixel resolution. The single allowed MVD value for these MV classes may be predefined. In some examples, the allowed single value may be the higher end values of the respective ranges for these MV classes in Table 4. For example, MV_CLASS_2 through MV_CLASS_10 may be above or equal to the threshold

class of MV_CLASS 2, and the single allowed MVD value for these classes may be predefined as 8, 16, 32, 64, 128, 256, 512, 1024, and 2048, respectively. In some other examples, the allowed single value may be the middle value of the respective ranges for these MV classes in Table 4. For example, MV_CLASS_2 through MV_CLASS_10 may be above the class threshold, and the single allowed MVD value for these classes may be predefined as 3, 6, 12, 24, 48, 96, 192, 384, 768, and 1536, respectively. Any other values within the ranges may also be defined as the single allowed resolutions for the respective MVD classes.

[0207] In the implementations above, only the “mv_class” signaling is sufficient for determining the MVD value when the signaled “mv_class” is equal to or above the predefined MVD class threshold. The magnitude and direction of the MVD would then be determined using “mv_class” and “mv_sign”.

[0208] As such, when MVD is signaled for only one reference frame (either from reference frame list 0 or list 1, but not both), or jointly signaled for two reference frames, the precision (or resolution) of the MVD may depend on the associated class of motion vector difference in Table 3 and/or the magnitude of MVD.

[0209] In some other implementations, the pixel resolution or precision for MVD may decrease or may be non-increasing with increase MVD magnitude. For example, the pixel resolution may depend on integer portion of the MVD magnitude. In some implementations, fractional pixel resolution may be allowed only for MVD magnitude smaller than or equal to an amplitude threshold. For a decoder, the integer portion of the MVD magnitude may first be extracted from a bitstream. The pixel resolution may then be determined, and decision may then be made as to whether any fractional MVD is in existence in the bit stream and needs to be parsed (e.g., if the fractional pixel resolution is disallowed for a particular extracted MVD integer magnitude, then no fractional MVD bits may be included in the bitstream needing extraction). The example implementations above related to MVD-class-dependent adaptive MVD pixel resolution applies to MVD magnitude dependent adaptive MVD pixel resolution. For a particular example, MVD classes above or encompassing the magnitude threshold may be allowed to have only one predefined value.

[0210] The various example implementations above applies to single-reference mode. These implementations also apply to the example NEW_NEARMV, NEAR_NEWMV, and/or NEW_NEWMV modes in compound prediction under MMVD. These implementations apply generally to adaptive resolution for any MVD.

[0211] In a particularly example implementation for adaptive MVD pixel resolution, the MVD pixel resolution for MVD magnitude below 1 may be fractional, and for MV class

of MV_CLASS_1 and above, only a single MVD magnitude equal to the ending value of the corresponding MVD magnitude range of Table 4 may be allowed. In such an example, the allowed MVD values are indicated in Table 4 for allowed fractional pixel resolution of 1/8, 1/4, or 1/2 pixel.

Table 5

MV class	Allowed Values of MVD
MV_CLASS_0	(1/8, 2/8, 3/8, 4/8, 5/8, 6/8, 7/8, 1, 2) when fractional pixel resolution is 1/8 pixel in the adaptive MVD resolution scheme; (2/8, 4/8, 6/8, 1, 2) when fractional pixel resolution is 1/4 pixel in the adaptive MVD resolution scheme; (1/2, 1, 2) when fractional pixel resolution is 1/2 pixel in the adaptive MVD resolution scheme; (1, 2) when pixel resolution is 1 pixel in the adaptive MVD resolution scheme.
MV_CLASS_1	4
MV_CLASS_2	8
MV_CLASS_3	16
MV_CLASS_4	32
MV_CLASS_5	64
MV_CLASS_6	128
MV_CLASS_7	256
MV_CLASS_8	512
MV_CLASS_9	1024
MV_CLASS_10	2048

[0212] For a coding block, whether adaptive MVD pixel resolution is used or not may be signaled explicitly or implicitly (derived). When adaptive MVD pixel resolution is signaled as not used, it indicates that different MVD classes may follow the MVD ranges indicated in Table 4, and a non-adaptive MVD pixel resolution may be defined or signaled. Such non-adaptive resolution may be fractional (such as 1/8, 1/4 or 1/2 pixel) or non-fractional (such as 1, 2, 4, ...pixels) and would apply to all MVD classes. The non-adaptive resolution basically determines the number of bits needed to signal the mv_bit, mv_fr, and mv_hp described above. When the non-adaptive resolution is fractionally, it may only determine the numbers of bits needed for signaling mv_fr, and mv_hp for all MVD classes

(independent of class of the MVD), and the number of bits to signal Mv_bit may MVD class dependent.

[0213] When adaptive MVD pixel resolution is signaled as being used, the allowed MVD levels or values in an adaptive manner such as the ones shown in Table 5 may be predefined or may be signaled. For example, they may be signaled in the bitstream in various manners depending on the particular scheme for adaptive MVD resolution. In the example of Table 5, a set of signaling syntax may be used to indicate the fractional resolution (e.g., 1/8 pixel), the magnitude threshold below which the signaled fractional resolution applies (e.g., MVD magnitude of 1 pixel). Another sets of syntax (which may be more complex) may be used signal other adaptive MVD resolution scheme. Such indication of the adaptive MVD pixel resolution scheme may be signaled at one of various coding levels, such as the sequence level, the picture level, the frame level, the slice level, the super block level, or the coding block level.

[0214] In some example implementations, an overall adaptive MVD pixel resolution scheme including but not limited to the one shown in Table 5 may be defined or signaled at a particular coding level (e.g., the sequence level, the picture level, the frame level, the slice level, the super block level). Such adaptive MVD pixel resolution scheme may be further modified at the same or another coding level such that the allowed MVD pixel resolution values for the various MVD classes may be adjusted or modified at the same or another coding level. If no adjustment is made at a particular coding level, then the signaled or predefined adaptive MVD pixel resolution scheme applies without modification. For example, an overarching adaptive MVD pixel resolution scheme may be defined or signaled at the frame level, whereas adjustment may be made at one or more superblock level or coding block level, and vice versa.

[0215] Such adjustment may be implemented as a restriction of MVD precision or expansion of MVD precision. The information associated with such adjustment may be predefined or signaled. A predefined adjustment may be applied to all coding blocks. Alternatively, the predefined adjustment may be activated at various coding levels by signaling.

[0216] In some implementations, such adjustment may be embodied as a maximum allowed MVD precision. For a particular coding block, when adaptive MVD resolution is applied, such maximum allowed MVD precision may be different from the MVD pixel precision of the adaptive MVD pixel resolution scheme that is specified/signaled/derived at picture level, or super block level, or coded block level, as described above. In such

situations, the allowed MVD resolution values for the various MVD classes may be determined by taking both the allowed values specified by or derived from the adaptive MVD pixel resolution scheme and the maximum allowed MVD precision.

[0217] For example, assume that the adaptive MVD pixel resolution scheme of Table 5 is predefined/signaled/derived for a certain coding level. Further assume that the maximum allowed precision is $\frac{1}{4}$ pixel, meaning that precisions at or higher than $\frac{1}{8}$ -pixel are disallowed for any MVD class regardless of the adaptive MVD resolution associated with Table 5. Then, by applying the maximum allowed pixel precision as a restriction to Table 5 indiscriminately, the allowed MVD pixel levels or values for the various MVD classes may be modified to:

Table 6

MV class	Allowed Values of MVD
MV_CLASS_0	(2/8, 4/8, 6/8, 1, 2) when fractional pixel resolution is $\frac{1}{8}$ pixel in the adaptive scheme; (1/4, 2/4, 3/4, 1, 2) when fractional pixel resolution is $\frac{1}{4}$ pixel in the adaptive scheme; (1/2, 1, 2) when fractional pixel resolution is $\frac{1}{2}$ pixel in the adaptive scheme; (1, 2) when pixel resolution is 1 pixel in the adaptive scheme.
MV_CLASS_1 through MV_CLASS_10	Same as Table 5

[0218] Disallowing $\frac{1}{8}$ -pixel precision or higher for all MVD classes by defining/signaling a maximum allowed MVD pixel precision of $\frac{1}{4}$ pixel is merely one example. In another example, the maximum allowed pixel precision may defined/signaled as $\frac{1}{2}$ pixel. The corresponding allowed MVD values for MV_CLASS_0 above may become: (1/2, 1, 2) for adaptive resolution schemes with fractional pixel resolution of $\frac{1}{8}$ pixel, $\frac{1}{4}$ pixel, and $\frac{1}{2}$ pixel, and (1, 2) for pixel resolution of 1 pixel.

[0219] As exemplified in the implementations above in relation to Table 6, when adaptive MVD resolution is applied using a defined/signaled/derived adaptive resolution scheme at a particular coding level and with an additional defined/signaled maximum allowed MVD precision at the same or a different coding level, it may be required/restricted that such maximum allowed MVD precision not be greater than the MVD resolution in the adaptive resolution scheme. In other words, the actually applied MVD precision derived from

considering both the adaptive resolution scheme and the maximum allowed precision is clipped by the MVD resolution in the adaptive resolution scheme (i.e., the maximum allowed precision would be ineffective when it is greater than the resolution defined/signaled/derived from the adaptive resolution scheme).

[0220] However, in some other implementations, such clipping may not be required and the defined/signaled maximum allowed MVD precision may control actual MVD resolution for at least some MVD classes. In these implementations, when adaptive MVD resolution is applied (as indicated by definition/signaling/derivation at various coding levels, as described above), the adjustment of the MVD levels for at least some MVD classes may involve increasing rather than restricting the adaptive MVD resolution defined/signaled/derived in the adaptive MVD pixel resolution scheme such as the one associated with Table 5. For example, an adjustment may be made to allow precision higher than specified/signaled/derived from the adaptive resolution scheme for MVD classes at or lower than a defined or signaled threshold MVD class level. Such higher precision, as described above, may be defined/signaled as the maximum allowed MVD precision. Such maximum allowed precision may be imposed at or lower than the threshold MVD class level regardless of the specified/signaled/derived MVD resolution in the adaptive resolution scheme. Specifically, such threshold MVD class level may be (but need not to be) MV_CLASS_0 (or the lowest MVD class level of the set of MVD classes, such as the set of MVD classes in Table 5). The maximum allowed pixel precision may be predefined/signaled. The maximum allowed pixel precision may be fractional. For a particular example, with the threshold MVD class being MV_CLASS_0 in the adaptive resolution scheme of Table 5, and when the MVD pixel resolution for MV_CLASS_0 is non-fractional 1 pixel, and that the maximum allowed fractional pixel precision for adjustment is 1/8, 1/4, or 1/2 pixel, the adjust allowable MVD values would be:

Table 7

MV class	Allowed Values of MVD
MV_CLASS_0	when pixel resolution is 1 pixel in the adaptive resolution scheme: (1/8, 2/8, 3/8, 4/8, 5/8, 6/8, 7/8, 1, 2) when the Maximum allowed fractional resolution is 1/8 pixel; (1/4, 2/4, 3/4, 1, 2) when the maximum allowed fractional resolution is 1/4 pixel; (1/2, 1, 2) when the maximum allowed fractional resolution is 1/2 pixel; (1, 2) when no maximum allowed resolution is defined/signaled.

MV_CLASS_1 through MV_CLASS_10	Same as Table 5
--------------------------------------	-----------------

[0221] In some example implementations alternative to that of Table 7, a threshold MVD amplitude may be used in place of the threshold MVD class level. In these implementations, higher precision may be imposed by the specified/signaled maximum allowed MVD precision for MVD having a magnitude at or lower than the threshold MVD amplitude rather than the threshold MVD class level. In such implementation, the mv_bit information in addition to the mv_class information may be signaled sufficiently early in the video stream such that the magnitude of the MVD can be determined in time in order to determine the allowed MVD values. For example, by replacing the threshold MVD class with a threshold MVD magnitude of 1/2-pixel, and still assume that the adaptive MVD resolution for MV_CLASS_0 is 1-pixel in the adaptive resolution scheme, Table 7 would become Table 8 below:

Table 8

MV class	Allowed Values of MVD
MV_CLASS_0	when pixel resolution is 1 pixel in the adaptive resolution scheme: (1/8, 2/8, 3/8, 1/2, 1, 2) when the Maximum allowed fractional resolution is 1/8 pixel; (1/4, 1/2, 1, 2) when the maximum allowed fractional resolution is 1/4 pixel; (1/2, 1, 2) when the maximum allowed fractional resolution is 1/2 pixel; (1, 2) when no maximum allowed resolution is defined/signaled.
MV_CLASS_1 through MV_CLASS_10	Same as Table 5

[0222] In some other example implementations, the adjustment above may involve only allowing a particular precision and lower precisions (e.g., fractional precision 1/8, 1/4, or 1/2 and lower) when the magnitude of an MVD is at or lower than a threshold MVD magnitude. In such implementation, again, the mv_bit information in addition to the mv_class information may be signaled sufficiently early in the video stream such that the magnitude of the MVD can be determined in time in order to determine the allowed MVD values.

[0223] In such implementations, no additional resolution may be imposed to the MVD values derived from the adaptive resolution scheme (such as Table 5). Instead, MVD

values associated with a resolution at or higher than a defined/signaled precision level may be disallowed when the amplitude of the MVD is higher than the threshold MVD magnitude. Assume, again, in the example of Table 5, and further assume that MVD values associated with a resolution at or higher than a defined/signaled precision of 1/8-pixel precision is disallowed for MVD magnitude higher than a threshold MVD magnitude of 1/2 pixel. Then Table 5 would be adjusted to:

Table 9

MV class	Allowed Values of MVD
MV_CLASS_0	(1/8, 2/8, 3/8, 1/2, 6/8, 1, 2) when fractional pixel resolution is 1/8 pixel in the adaptive scheme; (1/4, 1/2, 3/4, 1, 2) when fractional pixel resolution is 1/4 pixel in the adaptive scheme; (1/2, 1, 2) when fractional pixel resolution is 1/2 pixel in the adaptive scheme; (1, 2) when pixel resolution is 1 pixel in the adaptive scheme.
MV_CLASS_1 through MV_CLASS_10	Same as Table 5

[0224] In particular, as shown above, the allowed MVD values for MV_CLASS_0 and fractional resolution of 1/8 pixel, (1/8, 2/8, 3/8, 1/2, 5/8, 6/8, 7/8, 1, 2) is adjusted to (1/8, 2/8, 3/8, 1/2, 6/8, 1, 2), where the MVD values associated with 1/8 precision are only allowed and retained at or below the threshold MVD magnitude of 1/2 pixel. Above the 1/2 pixel magnitude, the MVD values associated with 1/8 precision, such as the 5/8-pixel value and the 7/8-pixel value are disallowed.

[0225] Similarly, assume, in the example of Table 5, that MVD values associated with a resolution at or higher than a defined/signaled precision of 1/4-pixel precision are disallowed when MVD magnitude is higher than a threshold magnitude of 1/2 pixel. Then Table 5 would be adjusted to:

Table 9

MV class	Allowed Values of MVD
MV_CLASS_0	(1/8, 2/8, 3/8, 1/2, 1, 2) when fractional pixel resolution is 1/8 pixel in the adaptive scheme; (1/4, 1/2, 1, 2) when fractional pixel resolution is 1/4 pixel in the adaptive scheme;

	(1/2, 1, 2) when fractional pixel resolution is 1/2 pixel in the adaptive scheme; (1, 2) when pixel resolution is 1 pixel in the adaptive scheme;
MV_CLASS_1 through MV_CLASS_10	Same as Table 5

[0226] In some of the implementations above, the threshold MVD magnitude may be 2 pixels or smaller, such as the ½ pixel magnitude threshold given in the examples above.

[0227] The example implementations above are described with respect to a particular MVD irrespective of whether the inter-prediction mode is either in the single-reference mode or compound-reference mode. In some other example implementations in the compound-reference mode, where an MV is predicted by multiple reference frames, a set of definition/signaling may be used to indicate whether an adaptive MVD resolution is applied and which one(s) of the reference frame of the multiple reference frames it is applied to.

[0228] In some example implementations, when MVDs are signaled for multiple reference frames, one (or more) flag(s)/index(s) may be signaled to indicate whether adaptive MVD resolution is applied or not.

[0229] For example, when MVDs are signaled for multiple reference frames (for example, in the NEW_NEWMV mode described above, or other compound-reference inter-prediction modes), one flag/index may be signaled in the video stream to indicate whether adaptive MVD resolution is applied to the signaling of MVDs for all of the multiple reference frames or not. If this flag/index is 1 (or 0), it indicates that adaptive MVD resolution is applied to the signaling of MVDs for all of the multiple reference frames. Otherwise, if this flag/index is 0 (or 1), adaptive MVD coding is applied to the signaling of MVDs for none of the multiple reference frames. In such implementations, with respect to the multiple inter-prediction reference frames, the adaptive MVD resolution is applied in an all-or-none scheme.

[0230] In some other examples, when MVDs are signaled for multiple reference frames (for example, in the NEW_NEWMV mode described above for two-reference frame compound inter-prediction mode) or other compound inter prediction modes), one flag/index may be signaled for each reference frame separately to indicate whether adaptive MVD resolution is applied to each reference frame or not. In such implementations, whether adaptive MVD resolution is applied or not may be determined individual for each of the reference frames. Decision of whether to apply the adaptive MVD resolution may be made at

the encoder independently for each of the multiple reference frames and signaled separately in the video stream.

[0231] In some example implementations, when MVDs are signaled for multiple reference frames, for each of the multiple reference frames, if MVD for that reference frame is non-zero, one flag/index may be signaled to indicate whether adaptive MVD resolution is applied to that reference frame or not. Otherwise, no flag/index needs to be signaled. In other words, if the MVD for a particular reference frame is signaled/indicated as zero, then there is no need to determine whether adaptive MVD resolution is applied or not and thus there is no need for any corresponding signaling in the video stream. However, in such implementations, an indication that the MVD is zero need to be signaled prior to when the determination is made as to whether adaptive resolution is applied or not.

[0232] Turning further to the signaling of MVD resolution, and in some example implementations, a flag/index may be signaled to explicitly indicate the MVD resolution for the current coding block, and the context used for entropy coding such flag/index may depend on the MVD class associated with the MVD. Such flag/index may be any of the MVD resolution used to derive the adaptive resolution scheme such as Table 5 or the maximum allowed MVD precision described above.

[0233] In some example implementations with respect the signaling of the MVD resolution, the various components of an MVD may be separately signaled. An MVD, for example, may include a horizontal component and vertical component. A flag/index may be signaled for each of the horizontal and vertical component of MVD to indicate the MVD resolution of horizontal and vertical component, respectively.

[0234] In some example implementations, the MVD resolution flag/index may be signaled after the MVD class information. Depending on the value of signaled MVD class information, such as MV_CLASS_0, MV_CLASS_1, MV_CLASS_2, and the like, a context value may be derived and used for signalling the MVD resolution flag/index for indicating the MVD resolution. In other words, the syntax(es) for signaling of the MVD resolution may be entropy coded using different contexts for different MVD classes or different MVD class groups.

[0235] Figure 18 shows a flow chart 1800 of an example method following the principles underlying the implementations above for adaptive MVD resolution. The example decoding method flow starts at S1801. In S1810, a video stream is received. In S1820, it is determined that the video block is inter-coded based on a prediction block and a motion vector (MV), wherein the MV is to be derived from a reference motion vector (RMV) and a

motion vector difference (MVD) for the video block. In S1830, in response to determining that the MVD is coded with an adaptive MVD pixel resolution: a reference MVD pixel precision for the current video block is determined; a maximum allowed MVD pixel precision is identified; a set of allowable MVD levels for the current video block are determined based on the reference MVD pixel precision and the maximum allowed MVD pixel precision; and the MVD from the video stream is derived according to at least one MVD parameters signaled in the video stream for the current video block and the set of allowable MVD levels. The example method stops at S1899.

[0236] . Figure 19 shows a flow chart 1900 of another example method following the principles underlying the implementations above for adaptive MVD resolution. The example decoding method flow starts at S1901. In S1910, a video stream is received. In S1920, it is determined that the current video block is inter-coded and associated with multiple reference frames. In S1930, it is further determined whether adaptive motion vector difference (MVD) pixel resolution is applied to at least one of the multiple reference frames based on a signaling in the video stream. The example method stops at S1999.

[0237] Figure 20 shows a flow chart 2000 of an example method following the principles underlying the implementations above for adaptive MVD resolution. The example decoding method flow starts at S2001. In S2010, a video stream is received. In S2020, it is determined that the video block is inter-coded based on a prediction block and a motion vector (MV), wherein the MV is to be derived from a reference motion vector (RMV) and a motion vector difference (MVD) for the video block. In S2030, a current MVD class of the MVD among a predefined set of MVD classes is determined. In S2040, at least one context for entropy-decoding at least one explicit signaling in the video stream is derived based on the current MVD class, the at least one explicit signaling being included in the video stream to specify an MVD pixel resolution for at least one component of the MVD. In S2050, the at least one explicit signaling from the video stream is entropy-decoded using the at least one context to determine the MVD pixel resolution for the at least one component of the MVD. The example method stops at S2099.

[0238] In the embodiments and implementation of this disclosure, any steps and/or operations may be combined or arranged in any amount or order, as desired. Two or more of the steps and/or operations may be performed in parallel. Embodiments and implementations in the disclosure may be used separately or combined in any order. Further, each of the methods (or embodiments), an encoder, and a decoder may be implemented by processing circuitry (e.g., one or more processors or one or more integrated circuits). In one example,

the one or more processors execute a program that is stored in a non-transitory computer-readable medium. Embodiments in the disclosure may be applied to a luma block or a chroma block. The term block may be interpreted as a prediction block, a coding block, or a coding unit, i.e. CU. The term block here may also be used to refer to the transform block. In the following items, when saying block size, it may refer to either the block width or height, or maximum value of width and height, or minimum of width and height, or area size (width * height), or aspect ratio (width:height, or height:width) of the block.

[0239] The techniques described above, can be implemented as computer software using computer-readable instructions and physically stored in one or more computer-readable media. For example, FIG. 21 shows a computer system (2100) suitable for implementing certain embodiments of the disclosed subject matter.

[0240] The computer software can be coded using any suitable machine code or computer language, that may be subject to assembly, compilation, linking, or like mechanisms to create code comprising instructions that can be executed directly, or through interpretation, micro-code execution, and the like, by one or more computer central processing units (CPUs), Graphics Processing Units (GPUs), and the like.

[0241] The instructions can be executed on various types of computers or components thereof, including, for example, personal computers, tablet computers, servers, smartphones, gaming devices, internet of things devices, and the like.

[0242] The components shown in FIG. 21 for computer system (2100) are exemplary in nature and are not intended to suggest any limitation as to the scope of use or functionality of the computer software implementing embodiments of the present disclosure. Neither should the configuration of components be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the exemplary embodiment of a computer system (2100).

[0243] Computer system (2100) may include certain human interface input devices. Such a human interface input device may be responsive to input by one or more human users through, for example, tactile input (such as: keystrokes, swipes, data glove movements), audio input (such as: voice, clapping), visual input (such as: gestures), olfactory input (not depicted). The human interface devices can also be used to capture certain media not necessarily directly related to conscious input by a human, such as audio (such as: speech, music, ambient sound), images (such as: scanned images, photographic images obtain from a still image camera), video (such as two-dimensional video, three-dimensional video including stereoscopic video).

[0244] Input human interface devices may include one or more of (only one of each depicted): keyboard (2101), mouse (2102), trackpad (2103), touch screen (2110), data-glove (not shown), joystick (2105), microphone (2106), scanner (2107), camera (2108).

[0245] Computer system (2100) may also include certain human interface output devices. Such human interface output devices may be stimulating the senses of one or more human users through, for example, tactile output, sound, light, and smell/taste. Such human interface output devices may include tactile output devices (for example tactile feedback by the touch-screen (2110), data-glove (not shown), or joystick (2105), but there can also be tactile feedback devices that do not serve as input devices), audio output devices (such as: speakers (2109), headphones (not depicted)), visual output devices (such as screens (2110) to include CRT screens, LCD screens, plasma screens, OLED screens, each with or without touch-screen input capability, each with or without tactile feedback capability—some of which may be capable to output two dimensional visual output or more than three dimensional output through means such as stereographic output; virtual-reality glasses (not depicted), holographic displays and smoke tanks (not depicted)), and printers (not depicted).

[0246] Computer system (2100) can also include human accessible storage devices and their associated media such as optical media including CD/DVD ROM/RW (2120) with CD/DVD or the like media (2121), thumb-drive (2122), removable hard drive or solid state drive (2123), legacy magnetic media such as tape and floppy disc (not depicted), specialized ROM/ASIC/PLD based devices such as security dongles (not depicted), and the like.

[0247] Those skilled in the art should also understand that term “computer readable media” as used in connection with the presently disclosed subject matter does not encompass transmission media, carrier waves, or other transitory signals.

[0248] Computer system (2100) can also include an interface (2154) to one or more communication networks (2155). Networks can for example be wireless, wireline, optical. Networks can further be local, wide-area, metropolitan, vehicular and industrial, real-time, delay-tolerant, and so on. Examples of networks include local area networks such as Ethernet, wireless LANs, cellular networks to include GSM, 3G, 4G, 5G, LTE and the like, TV wireline or wireless wide area digital networks to include cable TV, satellite TV, and terrestrial broadcast TV, vehicular and industrial to include CAN bus, and so forth. Certain networks commonly require external network interface adapters that attached to certain general-purpose data ports or peripheral buses (2149) (such as, for example USB ports of the computer system (2100)); others are commonly integrated into the core of the computer system (2100) by attachment to a system bus as described below (for example Ethernet

interface into a PC computer system or cellular network interface into a smartphone computer system). Using any of these networks, computer system (2100) can communicate with other entities. Such communication can be uni-directional, receive only (for example, broadcast TV), uni-directional send-only (for example CANbus to certain CANbus devices), or bi-directional, for example to other computer systems using local or wide area digital networks. Certain protocols and protocol stacks can be used on each of those networks and network interfaces as described above.

[0249] Aforementioned human interface devices, human-accessible storage devices, and network interfaces can be attached to a core (2140) of the computer system (2100).

[0250] The core (2140) can include one or more Central Processing Units (CPU) (2141), Graphics Processing Units (GPU) (2142), specialized programmable processing units in the form of Field Programmable Gate Areas (FPGA) (2143), hardware accelerators for certain tasks (2144), graphics adapters (2150), and so forth. These devices, along with Read-only memory (ROM) (2145), Random-access memory (2146), internal mass storage such as internal non-user accessible hard drives, SSDs, and the like (2147), may be connected through a system bus (2148). In some computer systems, the system bus (2148) can be accessible in the form of one or more physical plugs to enable extensions by additional CPUs, GPU, and the like. The peripheral devices can be attached either directly to the core's system bus (2148), or through a peripheral bus (2149). In an example, the screen (2110) can be connected to the graphics adapter (2150). Architectures for a peripheral bus include PCI, USB, and the like.

[0251] CPUs (2141), GPUs (2142), FPGAs (2143), and accelerators (2144) can execute certain instructions that, in combination, can make up the aforementioned computer code. That computer code can be stored in ROM (2145) or RAM (2146). Transitional data can also be stored in RAM (2146), whereas permanent data can be stored for example, in the internal mass storage (2147). Fast storage and retrieve to any of the memory devices can be enabled through the use of cache memory, that can be closely associated with one or more CPU (2141), GPU (2142), mass storage (2147), ROM (2145), RAM (2146), and the like.

[0252] The computer readable media can have computer code thereon for performing various computer-implemented operations. The media and computer code can be those specially designed and constructed for the purposes of the present disclosure, or they can be of the kind well known and available to those having skill in the computer software arts.

[0253] As a non-limiting example, the computer system having architecture (2100), and specifically the core (2140) can provide functionality as a result of processor(s)

(including CPUs, GPUs, FPGA, accelerators, and the like) executing software embodied in one or more tangible, computer-readable media. Such computer-readable media can be media associated with user-accessible mass storage as introduced above, as well as certain storage of the core (2140) that are of non-transitory nature, such as core-internal mass storage (2147) or ROM (2145). The software implementing various embodiments of the present disclosure can be stored in such devices and executed by core (2140). A computer-readable medium can include one or more memory devices or chips, according to particular needs. The software can cause the core (2140) and specifically the processors therein (including CPU, GPU, FPGA, and the like) to execute particular processes or particular parts of particular processes described herein, including defining data structures stored in RAM (2146) and modifying such data structures according to the processes defined by the software. In addition, or as an alternative, the computer system can provide functionality as a result of logic hardwired or otherwise embodied in a circuit (for example: accelerator (2144)), which can operate in place of or together with software to execute particular processes or particular parts of particular processes described herein. Reference to software can encompass logic, and vice versa, where appropriate. Reference to a computer-readable media can encompass a circuit (such as an integrated circuit (IC)) storing software for execution, a circuit embodying logic for execution, or both, where appropriate. The present disclosure encompasses any suitable combination of hardware and software.

[0254] While this disclosure has described several exemplary embodiments, there are alterations, permutations, and various substitute equivalents, which fall within the scope of the disclosure. It will thus be appreciated that those skilled in the art will be able to devise numerous systems and methods which, although not explicitly shown or described herein, embody the principles of the disclosure and are thus within the spirit and scope thereof.

Appendix A: Acronyms

JEM: joint exploration model

VVC: versatile video coding

BMS: benchmark set

MV: Motion Vector

HEVC: High Efficiency Video Coding

SEI: Supplementary Enhancement Information

VUI: Video Usability Information

GOPs: Groups of Pictures

TUs: Transform Units,
PUs: Prediction Units
CTUs: Coding Tree Units
CTBs: Coding Tree Blocks
PBs: Prediction Blocks
HRD: Hypothetical Reference Decoder
SNR: Signal Noise Ratio
CPUs: Central Processing Units
GPUs: Graphics Processing Units
CRT: Cathode Ray Tube
LCD: Liquid-Crystal Display
OLED: Organic Light-Emitting Diode
CD: Compact Disc
DVD: Digital Video Disc
ROM: Read-Only Memory
RAM: Random Access Memory
ASIC: Application-Specific Integrated Circuit
PLD: Programmable Logic Device
LAN: Local Area Network
GSM: Global System for Mobile communications
LTE: Long-Term Evolution
CANBus: Controller Area Network Bus
USB: Universal Serial Bus
PCI: Peripheral Component Interconnect
FPGA: Field Programmable Gate Areas
SSD: solid-state drive
IC: Integrated Circuit
HDR: high dynamic range
SDR: standard dynamic range
JVET: Joint Video Exploration Team
MPM: most probable mode
WAIP: Wide-Angle Intra Prediction
CU: Coding Unit
PU: Prediction Unit

TU: Transform Unit
CTU: Coding Tree Unit
PDPC: Position Dependent Prediction Combination
ISP: Intra Sub-Partitions
SPS: Sequence Parameter Setting
PPS: Picture Parameter Set
APS: Adaptation Parameter Set
VPS: Video Parameter Set
DPS: Decoding Parameter Set
ALF: Adaptive Loop Filter
SAO: Sample Adaptive Offset
CC-ALF: Cross-Component Adaptive Loop Filter
CDEF: Constrained Directional Enhancement Filter
CCSO: Cross-Component Sample Offset
LSO: Local Sample Offset
LR: Loop Restoration Filter
AV1: AOMedia Video 1
AV2: AOMedia Video 2
MVD: Motion Vector difference
CfL: Chroma from Luma
SDT: Semi Decoupled Tree
SDP: Semi Decoupled Partitioning
SST: Semi Separate Tree
SB: Super Block
IBC (or IntraBC): Intra Block Copy
CDF: Cumulative Density Function
SCC: Screen Content Coding
GBI: Generalized Bi-prediction
BCW: Bi-prediction with CU-level Weights
CIIP: Combined intra-inter prediction
POC: Picture Order Count
RPS: Reference Picture Set
DPB: Decoded Picture Buffer
MMVD: Merge Mode with Motion Vector Difference

WHAT IS CLAIMED IS:

1. A method for processing a current video block of a video stream, comprising;
receiving the video stream;
determining that the current video block is inter-coded based on a prediction block and a motion vector (MV), wherein the MV is to be derived from a reference motion vector (RMV) and a motion vector difference (MVD) for the current video block; and
in response to determining that the MVD is coded with an adaptive MVD pixel resolution:
 - determining a reference MVD pixel precision for the current video block;
 - identifying a maximum allowed MVD pixel precision;
 - determining a set of allowable MVD levels for the current video block based on the reference MVD pixel precision and the maximum allowed MVD pixel precision; and
 - deriving the MVD from the video stream according to at least one MVD parameters signaled in the video stream for the current video block and the set of allowable MVD levels.
2. The method of claim 1, wherein the reference MVD pixel precision for the current video block is specified/signaled/derived at a sequence level, a picture level, a frame level, a superblock level, or coding block level.
3. The method of claim 2, wherein the reference MVD pixel precision for the current video block depends on an MVD class associated with the MVD of the current video block.
4. The method of claim 2, wherein the reference MVD pixel precision for the current video block depends on an MVD magnitude of the MVD of the current video block.
5. The method of claim 2, wherein the maximum allowed MVD pixel precision is predefined.
6. The method of any one of claims 1-5, further comprising determining a current MVD class among a predefined set of MVD classes, wherein determining the set of allowable

MVD levels for the MVD based on the reference MVD pixel precision and the maximum allowed MVD pixel precision comprises:

excluding, from a reference MVD level set determined based on the reference MVD pixel precision and the current MVD class, MVD levels associated with an MVD pixel precision at or higher than the maximum allowed MVD pixel precision, to determine the set of allowable MVD levels for the current video block.

7. The method of claim 6, wherein the maximum allowed MVD pixel precision is $\frac{1}{4}$ pixel.
8. The method of any one of claims 1-5, wherein MVD levels associated with a $\frac{1}{8}$ pixel or higher precision or higher are excluded in the set of allowable MVD levels for the current video block.
9. The method of any one of claims 1-5, further comprising determining a current MVD class among a predefined set of MVD classes wherein:

MVD levels associated with fractional MVD precision is included in the set of allowable MVD levels regardless of the reference MVD precision when the current MVD class is at or lower than a threshold MVD class.
10. The method of claim 9, wherein the threshold MVD class is a lowest MVD class among the predefined set of MVD classes.
11. The method of any one of claims 1-5, further comprising determining a magnitude of the MVD, wherein MVD levels associated with an MVD precision higher than a threshold MVD precision are allowed in the set of allowable MVD levels only when the magnitude of the MVD is at or lower than a threshold MVD magnitude.
12. The method of claim 11, wherein the threshold MVD magnitude is 2 pixels or smaller.
13. The method of claim 12, wherein the threshold MVD precision is 1 pixel.

14. The method of claim 11, wherein MVD levels associated with an MVD precision of $\frac{1}{4}$ pixel or higher are only allowed when the magnitude of the MVD is at or lower than $\frac{1}{2}$ pixel.
15. The method of any one of claims 1-5, wherein the maximum allowed MVD pixel precision is not greater than the reference MVD pixel precision.
16. A method for processing a current video block of a video stream, comprising:
 - receiving the video stream;
 - determining that the current video block is inter-coded and associated with multiple reference frames; and
 - determining whether adaptive motion vector difference (MVD) pixel resolution is applied to at least one of the multiple reference frames based on a signaling in the video stream.
17. The method of claim 16, wherein the signaling comprises a single-bit flag to indicate whether adaptive MVD pixel resolution is applied to all or none of the multiple reference frames.
18. The method of claim 16, wherein the signaling comprises separate flags each corresponding to one of the multiple reference frames to indicate whether adaptive MVD pixel resolution is applied.
19. The method of claim 16, wherein the signaling comprises, for each of the multiple reference frames:
 - implicit indication that no adaptive MVD pixel resolution is applied when an MVD corresponding to the each of the multiple reference frames is zero; and
 - a single-bit flag for indicating whether adaptive MVD pixel resolution is applied when the MVD corresponding to the each of the multiple reference frames is non-zero.
20. A method for processing a current video block of a video stream, comprising:
 - receiving the video stream;

determining that the current video block is inter-coded based on a prediction block and a motion vector (MV), wherein the MV is to be derived from a reference motion vector (RMV) and a motion vector difference (MVD) for the current video block;

determining a current MVD class of the MVD among a predefined set of MVD classes;

deriving, based on the current MVD class, at least one context for entropy-decoding at least one explicit signaling in the video stream, the at least one explicit signaling being included in the video stream to specify an MVD pixel resolution for at least one component of the MVD; and

entropy-decoding the at least one explicit signaling from the video stream using the at least one context to determine the MVD pixel resolution for the at least one component of the MVD.

21. The method of claim 20, wherein the at least one component of the MVD comprises a horizontal component and a vertical component of the MVD, and the at least one context comprises two separate contexts each associated with one of the horizontal component and the vertical component of the MVD, the horizontal component and the vertical component being associated with separate MVD pixel resolutions.

22. A video processing device, comprising a memory for storing computer instructions and a processor, the processor, when executing the computer instructions, is configured to perform the method in any one of claims 1-5 and 16-21.