SEPARATE MERGE LIST FOR SUBBLOCK MERGE CANDIDATES AND INTRA-INTER TECHNIQUES HARMONIZATION FOR VIDEO CODING

TECHNICAL FIELD

Embodiments of the present application (disclosure) generally relate to the field of picture processing and more particularly to harmonization of two methods with each other. More specifically, embodiments propose methods of joint usage and signaling a separate merge list for subblock merge candidates and multi-hypothesis prediction for intra and inter mode techniques.

BACKGROUND

Video coding (video encoding and decoding) is used in a wide range of digital video applications, for example broadcast digital TV, video transmission over internet and mobile networks, real-time conversational applications such as video chat, video conferencing, DVD and Blu-ray discs, video content acquisition and editing systems, and camcorders of security applications.

The amount of video data needed to depict even a relatively short video can be substantial, which may result in difficulties when the data is to be streamed or otherwise communicated across a communications network with limited bandwidth capacity. Thus, video data is generally compressed before being communicated across modern day telecommunications networks. The size of a video could also be an issue when the video is stored on a storage device because memory resources may be limited. Video compression devices often use software and/or hardware at the source to code the video data prior to transmission or storage, thereby decreasing the quantity of data needed to represent digital video images. The compressed data is then received at the destination by a video decompression device that decodes the video data. With limited network resources and ever increasing demands of higher video quality, improved compression and decompression techniques that improve compression ratio with little to no sacrifice in picture quality are desirable.

SUMMARY

The present disclosure provides apparatuses and methods for encoding and decoding video.

The present invention aims to harmonize usage and signaling of multi-hypothesis prediction for intra and inter mode together with separate merge lists for subblock merge candidates.

The foregoing and other objects are achieved by the subject matter of the independent claims. Further implementation forms are apparent from the dependent claims, the description and the figures.

According to a first aspect of the invention, a method of encoding video data into a bitstream is provided. The method comprises using a first technique and/or a second technique. The first technique comprises using a separate merge list for subblock merge candidates. The second technique comprises multi-hypothesis prediction for intra and inter mode. The method comprises, for a coding block, transmitting a first control block in the bitstream and, depending on whether the first technique is used for the coding block, transmitting or not transmitting a second flag in the bitstream, The first control flag indicates whether to use the first technique. The second control flag indicates whether to use the second technique.

According to a second aspect of the invention, a method of decoding video data received in the bitstream is provided. The method comprises using a first technique and/or a second technique. The first technique comprises using a separate merge list for subblock merge candidates. The second technique comprises multi-hypothesis prediction for intra and inter mode. The method comprises, for a coding block, receiving the first control flag from the bitstream, wherein the first control flag indicates whether to use the first technique, and and, depending on whether the first technique is used for the coding block, receiving a second control flag from the bitstream. The second control flag indicates whether to use the second technique.

It is the particular approach of the present invention to generate and transmit a second control flag indicating whether or not to use multi-hypothesis prediction for intra and inter mode only conditionally, upon a determination whether or not separate merge list technique for subblock merge candidates is used. On the other hand, the decoder is capable to decide about usage of multi-hypothesis prediction for intra and inter mode and separate merge list technique for subblock merge candidates although the second control flag is transmitted only conditionally.

In a possible implementation form of the method according to the first aspect as such, the second coding flag is transmitted if and only if the first technique is not used for the coding block.

In a possible implementation form of the method according to the preceding implementation or the first aspect as such, the second control flag is transmitted if the coding block is quoted in merge mode. In a possible implementation form of the method according the preceding implementation of the first aspect, the second control flag is not transmitted if the coding block is not coded in merge mode. In a possible implementation form of the method according to a preceding implementation of the first aspect or the first aspect as such, the second control flag is transmitted if the coding block is coded in skip mode. In a possible implementation form of the preceding implementation of the first aspect, the second control flag is not transmitted if the coding block is not coded in skip mode or in merge mode.

Hence, signaling according to the particular approach of the present invention is applicable either in merge mode, or in skip mode, or in both merge and skip modes.

In a possible implementation form of the method according to the second aspect as such, only of the first technique is not used, the second control flag is received.

Hence, the decoder may directly infer from the presence of the second control flag in the received bit stream that the separate merge list technique is not used for a current coding block. Thus, evaluation of the first control flag indicating whether or not to use the separate merge list technique for subblock candidates is only necessary if the second control flag is not included in the received bitstream.

In a possible implementation form of the method according to the preceding implementation of the second aspect or the second aspect as such, the second control flag is received if the coding block is coded in merge mode. In a possible implementation form of the method of the preceding implementation of the second aspect, the second control flag is not received if the coding block is not coded in merge mode. in a possible implementation form of the method according to a preceding implementation form of the second aspect of the second aspect as such, second control

flag is received if the coding block is coded in skip mode. In accordance with a possible implementation form of the method according to the preceding implementation form of the second aspect, the second control flag is not received if the coding block is not coded in skip mode.

The encoding and the decoding methods defined in the claims, the description and the figures can each be performed by an encoding apparatus and a decoding apparatus, respectively.

According to a third aspect, the invention relates to an encoder comprising processing circuitry for carrying out a method according to the first aspect as such for any of its implementation forms.

According to a fourth aspect, the invention relates to a decoder comprising processing circuitry for carrying out a method according to the second aspect as such any of its implementation forms.

According to a fifth aspect, the invention relates to an encoder. The encoder comprises one or more processors and a non-transitory computer-readable storage medium coupled to the processors and storing programming for execution by the processors. The programming, when executed by the processors, configures the decoder to carry out the method according to the first aspect as such or any of its implementation forms.

According to a sixth aspect, the invention relates to a decoder. The decoder comprises one or more processors and a non-transitory computer-readable storage medium coupled to the processors and storing programming for execution by the processors. The programming, when executed by the processors, configures the decoder to carry out the method according to the second aspect as such any of its implementation forms.

According to a seventh aspect, the invention relates to an encoder for video encoding of video data into a bitstream, the encoder comprising means for performing a first technique and/or a second technique. The first technique comprises using a separate merge list for subblock merge candidates. The second technique comprises multi-hypothesis prediction for intra and inter mode. The encoder further comprises means for, for a coding block, transmitting a first control block in the bitstream and means for, depending on whether the first technique is used for the coding block,

transmitting or not transmitting a second flag in the bitstream, The first control flag indicates whether to use the first technique. The second control flag indicates whether to use the second technique.

According to an eighth aspect, the invention relates to a decoder for video decoding of video data received in a bitstream. The decoder comprises means for performing a first technique and/or a second technique. The first technique comprises using a separate merge list for subblock merge candidates. The second technique comprises multi-hypothesis prediction for intra and inter mode. The decoder further comprises means for, for a coding block, receiving the first control flag from the bitstream, wherein the first control flag indicates whether to use the first technique, and means for, depending on whether the first technique is used for the coding block, receiving a second control flag from the bitstream. The second control flag indicates whether to use the second technique.

According to a further aspect, the invention relates to a non-transitory computer-readable medium carrying a program code which, when executed by a computer device, causes the computer device to perform a method according to the first or second aspect.

Possible implementation forms of the encoder and decoder caudal the seventh and sixth aspects correspond to the possible implementation forms of the methods according to the first and second aspects.

An apparatus for encoding or for decoding a video stream may include a processor and a memory. The memory stores instructions that cause the processor to perform the encoding or decoding method.

For each of the encoding or decoding methods disclosed herein, a computer-readable storage medium is proposed, the storage medium having stored thereon instructions that when executed cause one or more processors to encode or decode video data. The instructions cause the one or more processors to perform the respective encoding or decoding method.

Furthermore, for each of the encoding or decoding methods disclosed herein, a computer program product is proposed. The computer program product comprises program code for performing the respective method.

Details of one or more embodiments are set forth in the accompanying drawings and the description below. Other features, objects, and advantages will be apparent from the description, drawings, and claims.

In addition, the present invention further provides the following embodiments.

A encoded bit stream of a video data, comprises, multiply syntax elements which include a first control flag, and a second flag being conditionally signaled based on the first control flag, the first control flag indicates that if a first technique is used, wherein, the first technique (S101) comprises using a separate merge list for subblock merge candidates; and the second control flag indicates that if a second technique is used, wherein, the second technique (S103) comprises multi-hypothesis prediction for intra and inter mode.

A computing storage medium, storing a bit stream decoded by a video decoding device, the bit stream comprise a number of coding blocks of an image or video signal, and a number of syntax elements which include a first control flag, and a second flag being conditionally signaled based on the first control flag, the first control flag indicates that if a first technique is used, wherein, the first technique (S101) comprises using a separate merge list for subblock merge candidates; and the second control flag indicates that if a second technique is used, wherein, the second technique (S103) comprises multi-hypothesis prediction for intra and inter mode.

A non-transitory computer-readable storage medium storing video information generated by using any of the encoding method of claim 1 to 6 of the pending claims.

BRIEF DESCRIPTION OF THE DRAWINGS

In the following embodiments of the invention are described in more detail with reference to the attached figures and drawings, in which:

Fig. 1A is a block diagram showing an example of a video coding system configured to implement embodiments of the invention;

Fig. 1B is a block diagram showing another example of a video coding system configured to implement embodiments of the invention;

Fig. 2 is a block diagram showing an example of a video encoder configured to implement embodiments of the invention;

Fig. 3 is a block diagram showing an example structure of a video decoder configured to implement embodiments of the invention;

Fig. 4 is a block diagram illustrating an example of an encoding apparatus or a decoding apparatus;

Fig. 5 is a block diagram illustrating another example of an encoding apparatus or a decoding apparatus;

Fig. 6 is a flowchart illustrating an exemplary encoding method according to embodiments of the present invention; and

Fig. 7 is a flowchart illustrating an exemplary decoding method according to embodiments of the present invention.

In the following identical reference signs refer to identical or at least functionally equivalent features if not explicitly specified otherwise.

DETAILED DESCRIPTION OF THE EMBODIMENTS

In the following description, reference is made to the accompanying figures, which form part of the disclosure, and which show, by way of illustration, specific aspects of embodiments of the invention or specific aspects in which embodiments of the present invention may be used. It is understood that embodiments of the invention may be used in other aspects and comprise structural or logical changes not depicted in the figures. The following detailed description, therefore, is not to be taken in a limiting sense, and the scope of the present invention is defined by the appended claims.

For instance, it is understood that a disclosure in connection with a described method may also hold true for a corresponding device or system configured to perform the method and vice versa.

For example, if one or a plurality of specific method steps are described, a corresponding device may include one or a plurality of units, e.g., functional units, to perform the described one or plurality of method steps (e.g., one unit performing the one or plurality of steps, or a plurality of units each performing one or more of the plurality of steps), even if such one or more units are not explicitly described or illustrated in the figures. On the other hand, for example, if a specific apparatus is described based on one or a plurality of units, e.g., functional units, a corresponding method may include one step to perform the functionality of the one or plurality of units (e.g., one step performing the functionality of the one or plurality of units, or a plurality of steps each performing the functionality of one or more of the plurality of units), even if such one or plurality of steps are not explicitly described or illustrated in the figures. Further, it is understood that the features of the various exemplary embodiments and/or aspects described herein may be combined with each other, unless specifically noted otherwise.

Video coding typically refers to the processing of a sequence of pictures, which form the video or video sequence. Instead of the term “picture” the term “frame” or “image” may be used as synonyms in the field of video coding. Video coding (or coding in general) comprises two parts video encoding and video decoding. Video encoding is performed at the source side, typically comprising processing (e.g., by compression) the original video pictures to reduce the amount of data required for representing the video pictures (for more efficient storage and/or transmission). Video decoding is performed at the destination side and typically comprises the inverse processing compared to the encoder to reconstruct the video pictures. Embodiments referring to “coding” of video pictures (or pictures in general) shall be understood to relate to “encoding” or “decoding” of video pictures or respective video sequences. The combination of the encoding part and the decoding part is also referred to as CODEC (Coding and Decoding).

In case of lossless video coding, the original video pictures can be reconstructed, i.e. the reconstructed video pictures have the same quality as the original video pictures (assuming no transmission loss or other data loss during storage or transmission). In case of lossy video coding, further compression, e.g., by quantization, is performed, to reduce the amount of data representing the video pictures, which cannot be completely reconstructed at the decoder, i.e. the quality of the

reconstructed video pictures is lower or worse compared to the quality of the original video pictures.

Several video coding standards belong to the group of “lossy hybrid video codecs” (i.e. combine spatial and temporal prediction in the sample domain and 2D transform coding for applying quantization in the transform domain). Each picture of a video sequence is typically partitioned into a set of non-overlapping blocks and the coding is typically performed on a block level. In other words, at the encoder the video is typically processed, i.e. encoded, on a block (video block) level, e.g., by using spatial (intra picture) prediction and/or temporal (inter picture) prediction to generate a prediction block, subtracting the prediction block from the current block (block currently processed/to be processed) to obtain a residual block, transforming the residual block and quantizing the residual block in the transform domain to reduce the amount of data to be transmitted (compression), whereas at the decoder the inverse processing compared to the encoder is applied to the encoded or compressed block to reconstruct the current block for representation. Furthermore, the encoder duplicates the decoder processing loop such that both will generate identical predictions (e.g., intra- and inter predictions) and/or re-constructions for processing, i.e. coding, the subsequent blocks.

In the following embodiments of a video coding system 10, a video encoder 20 and a video decoder 30 are described based on Figs. 1 to 3.

Fig. 1A is a schematic block diagram illustrating an example coding system 10, e.g., a video coding system 10 (or short coding system 10) that may utilize techniques of this present application. Video encoder 20 (or short encoder 20) and video decoder 30 (or short decoder 30) of video coding system 10 represent examples of devices that may be configured to perform techniques in accordance with various examples described in the present application.

As shown in Fig. 1A, the coding system 10 comprises a source device 12 configured to provide encoded picture data 21 e.g., to a destination device 14 for decoding the encoded picture data 13.

The source device 12 comprises an encoder 20, and may additionally, i.e. optionally, comprise a picture source 16, a pre-processor (or pre-processing unit) 18, e.g., a picture pre-processor 18, and a communication interface or communication unit 22.

The picture source 16 may comprise or be any kind of picture capturing device, for example a camera for capturing a real-world picture, and/or any kind of a picture generating device, for example a computer-graphics processor for generating a computer animated picture, or any kind of other device for obtaining and/or providing a real-world picture, a computer generated picture (e.g., a screen content, a virtual reality (VR) picture) and/or any combination thereof (e.g., an augmented reality (AR) picture). The picture source may be any kind of memory or storage storing any of the aforementioned pictures.

In distinction to the pre-processor 18 and the processing performed by the pre-processing unit 18, the picture or picture data 17 may also be referred to as raw picture or raw picture data 17.

Pre-processor 18 is configured to receive the (raw) picture data 17 and to perform pre-processing on the picture data 17 to obtain a pre-processed picture 19 or pre-processed picture data 19. Pre-processing performed by the pre-processor 18 may, e.g., comprise trimming, color format conversion (e.g., from RGB to YCbCr), color correction, or de-noising. It can be understood that the pre-processing unit 18 may be optional component.

The video encoder 20 is configured to receive the pre-processed picture data 19 and provide encoded picture data 21 (further details will be described below, e.g., based on Fig. 2).

Communication interface 22 of the source device 12 may be configured to receive the encoded picture data 21 and to transmit the encoded picture data 21 (or any further processed version thereof) over communication channel 13 to another device, e.g., the destination device 14 or any other device, for storage or direct reconstruction.

The destination device 14 comprises a decoder 30 (e.g., a video decoder 30), and may additionally, i.e. optionally, comprise a communication interface or communication unit 28, a post-processor 32 (or post-processing unit 32) and a display device 34.

The communication interface 28 of the destination device 14 is configured to receive the encoded picture data 21 (or any further processed version thereof), e.g., directly from the source device 12 or from any other source, e.g., a storage device, e.g., an encoded picture data storage device, and provide the encoded picture data 21 to the decoder 30.

The communication interface 22 and the communication interface 28 may be configured to transmit or receive the encoded picture data 21 or encoded data 13 via a direct communication link between the source device 12 and the destination device 14, e.g., a direct wired or wireless connection, or via any kind of network, e.g., a wired or wireless network or any combination thereof, or any kind of private and public network, or any kind of combination thereof.

The communication interface 22 may be, e.g., configured to package the encoded picture data 21 into an appropriate format, e.g., packets, and/or process the encoded picture data using any kind of transmission encoding or processing for transmission over a communication link or communication network.

The communication interface 28, forming the counterpart of the communication interface 22, may be, e.g., configured to receive the transmitted data and process the transmission data using any kind of corresponding transmission decoding or processing and/or de-packaging to obtain the encoded picture data 21.

Both communication interface 22 and communication interface 28 may be configured as unidirectional communication interfaces as indicated by the arrow for the communication channel 13 in Fig. 1A pointing from the source device 12 to the destination device 14, or bi-directional communication interfaces, and may be configured, e.g., to send and receive messages, e.g., to set up a connection, to acknowledge and exchange any other information related to the communication link and/or data transmission, e.g., encoded picture data transmission.

The decoder 30 is configured to receive the encoded picture data 21 and provide decoded picture data 31 or a decoded picture 31 (further details will be described below, e.g., based on Fig. 3 or Fig. 5).

The post-processor 32 of destination device 14 is configured to post-process the decoded picture data 31 (also called reconstructed picture data), e.g., the decoded picture 31, to obtain post-processed picture data 33, e.g., a post-processed picture 33. The post-processing performed by the post-processing unit 32 may comprise, e.g., color format conversion (e.g., from YCbCr to RGB), color correction, trimming, or re-sampling, or any other processing, e.g., for preparing the decoded picture data 31 for display, e.g., by display device 34.

The display device 34 of the destination device 14 is configured to receive the post-processed picture data 33 for displaying the picture, e.g., to a user or viewer. The display device 34 may be or comprise any kind of display for representing the reconstructed picture, e.g., an integrated or external display or monitor. The displays may, e.g., comprise liquid crystal displays (LCD), organic light emitting diodes (OLED) displays, plasma displays, projectors , micro LED displays, liquid crystal on silicon (LCoS), digital light processor (DLP) or any kind of other display.

Although Fig. 1A depicts the source device 12 and the destination device 14 as separate devices, embodiments of devices may also comprise both or both functionalities, the source device 12 or corresponding functionality and the destination device 14 or corresponding functionality. In such embodiments the source device 12 or corresponding functionality and the destination device 14 or corresponding functionality may be implemented using the same hardware and/or software or by separate hardware and/or software or any combination thereof.

As will be apparent for the skilled person based on the description, the existence and (exact) split of functionalities of the different units or functionalities within the source device 12 and/or destination device 14 as shown in Fig. 1A may vary depending on the actual device and application.

The encoder 20 (e.g., a video encoder 20) or the decoder 30 (e.g., a video decoder 30) or both encoder 20 and decoder 30 may be implemented using processing circuitry as shown in Fig. 1B, such as one or more microprocessors, digital signal processors (DSPs), application-specific integrated circuits (ASICs), field-programmable gate arrays (FPGAs), discrete logic, hardware, video coding dedicated or any combinations thereof. The encoder 20 may be implemented using

processing circuitry 46 to embody the various modules as discussed with respect to encoder 20 of Fig. 2 and/or any other encoder system or subsystem described herein. The decoder 30 may be implemented using processing circuitry 46 to embody the various modules as discussed with respect to decoder 30 of Fig. 3 and/or any other decoder system or subsystem described herein. The processing circuitry may be configured to perform the various operations as discussed later. As shown in Fig. 5, if the techniques are implemented partially in software, a device may store instructions for the software in a suitable, non-transitory computer-readable storage medium and may execute the instructions in hardware using one or more processors to perform the techniques of this disclosure. Either of video encoder 20 and video decoder 30 may be integrated as part of a combined encoder/decoder (CODEC) in a single device, for example, as shown in Fig. 1B.

Source device 12 and destination device 14 may comprise any of a wide range of devices, including any kind of handheld or stationary devices, e.g., notebook or laptop computers, mobile phones, smart phones, tablets or tablet computers, cameras, desktop computers, set-top boxes, televisions, display devices, digital media players, video gaming consoles, video streaming devices(such as content services servers or content delivery servers), broadcast receiver device, broadcast transmitter device, or the like and may use no or any kind of operating system. In some cases, the source device 12 and the destination device 14 may be equipped for wireless communication. Thus, the source device 12 and the destination device 14 may be wireless communication devices.

In some cases, video coding system 10 illustrated in Fig. 1A is merely an example and the techniques of the present application may apply to video coding settings (e.g., video encoding or video decoding) that do not necessarily include any data communication between the encoding and decoding devices. In other examples, data is retrieved from a local memory, streamed over a network, or the like. A video encoding device may encode and store data to memory, and/or a video decoding device may retrieve and decode data from memory. In some examples, the encoding and decoding is performed by devices that do not communicate with one another, but simply encode data to memory and/or retrieve and decode data from memory.

For convenience of description, embodiments of the invention are described herein, for example, by reference to High-Efficiency Video Coding (HEVC) or to the reference software of Versatile Video coding (VVC), the next generation video coding standard developed by the Joint Collaboration Team on Video Coding (JCT-VC) of ITU-T Video Coding Experts Group (VCEG) and ISO/IEC Motion Picture Experts Group (MPEG). One of ordinary skill in the art will understand that embodiments of the invention are not limited to HEVC or VVC.

Encoder and Encoding Method

Fig. 2 shows a schematic block diagram of an example video encoder 20 that is configured to implement the techniques of the present application**.** In the example of Fig. 2, the video encoder 20 comprises an input 201 (or input interface 201), a residual calculation unit 204, a transform processing unit 206, a quantization unit 208, an inverse quantization unit 210, and inverse transform processing unit 212, a reconstruction unit 214, a loop filter unit 220, a decoded picture buffer (DPB) 230, a mode selection unit 260, an entropy encoding unit 270 and an output 272 (or output interface 272). The mode selection unit 260 may include an inter prediction unit 244, an intra prediction unit 254 and a partitioning unit 262. Inter prediction unit 244 may include a motion estimation unit and a motion compensation unit (not shown). A video encoder 20 as shown in Fig. 2 may also be referred to as hybrid video encoder or a video encoder according to a hybrid video codec.

The residual calculation unit 204, the transform processing unit 206, the quantization unit 208, the mode selection unit 260 may be referred to as forming a forward signal path of the encoder 20, whereas the inverse quantization unit 210, the inverse transform processing unit 212, the reconstruction unit 214, the buffer 216, the loop filter 220, the decoded picture buffer (DPB) 230, the inter prediction unit 244 and the intra-prediction unit 254 may be referred to as forming a backward signal path of the video encoder 20, wherein the backward signal path of the video encoder 20 corresponds to the signal path of the decoder (see video decoder 30 in Fig. 3). The inverse quantization unit 210, the inverse transform processing unit 212, the reconstruction unit 214, the loop filter 220, the decoded picture buffer (DPB) 230, the inter prediction unit 244 and the intra-prediction unit 254 are also referred to forming the “built-in decoder” of video encoder 20.

Pictures & Picture Partitioning (Pictures & Blocks)

The encoder 20 may be configured to receive, e.g., via input 201, a picture 17 (or picture data 17), e.g., picture of a sequence of pictures forming a video or video sequence. The received picture or picture data may also be a pre-processed picture 19 (or pre-processed picture data 19). For sake of simplicity the following description refers to the picture 17. The picture 17 may also be referred to as current picture or picture to be coded (in particular in video coding to distinguish the current picture from other pictures, e.g., previously encoded and/or decoded pictures of the same video sequence, i.e. the video sequence which also comprises the current picture).

A (digital) picture is or can be regarded as a two-dimensional array or matrix of samples with intensity values. A sample in the array may also be referred to as pixel (short form of picture element) or a pel. The number of samples in horizontal and vertical direction (or axis) of the array or picture define the size and/or resolution of the picture. For representation of color, typically three color components are employed, i.e. the picture may be represented or include three sample arrays. In RBG format or color space a picture comprises a corresponding red, green and blue sample array. However, in video coding each pixel is typically represented in a luminance and chrominance format or color space, e.g., YCbCr, which comprises a luminance component indicated by Y (sometimes also L is used instead) and two chrominance components indicated by Cb and Cr. The luminance (or short luma) component Y represents the brightness or grey level intensity (e.g., like in a grey-scale picture), while the two chrominance (or short chroma) components Cb and Cr represent the chromaticity or color information components. Accordingly, a picture in YCbCr format comprises a luminance sample array of luminance sample values (Y), and two chrominance sample arrays of chrominance values (Cb and Cr). Pictures in RGB format may be converted or transformed into YCbCr format and vice versa, the process is also known as color transformation or conversion. If a picture is monochrome, the picture may comprise only a luminance sample array. Accordingly, a picture may be, for example, an array of luma samples in monochrome format or an array of luma samples and two corresponding arrays of chroma samples in 4:2:0, 4:2:2, and 4:4:4 color format.

Embodiments of the video encoder 20 may comprise a picture partitioning unit (not depicted in Fig. 2) configured to partition the picture 17 into a plurality of (typically non-overlapping)

picture blocks 203. These blocks may also be referred to as root blocks, macro blocks (H.264/AVC) or coding tree blocks (CTB) or coding tree units (CTU) (H.265/HEVC and VVC). The picture partitioning unit may be configured to use the same block size for all pictures of a video sequence and the corresponding grid defining the block size, or to change the block size between pictures or subsets or groups of pictures, and partition each picture into the corresponding blocks.

In further embodiments, the video encoder may be configured to receive directly a block 203 of the picture 17, e.g., one, several or all blocks forming the picture 17. The picture block 203 may also be referred to as current picture block or picture block to be coded.

Like the picture 17, the picture block 203 again is or can be regarded as a two-dimensional array or matrix of samples with intensity values (sample values), although of smaller dimension than the picture 17. In other words, the block 203 may comprise, e.g., one sample array (e.g., a luma array in case of a monochrome picture 17, or a luma or chroma array in case of a color picture) or three sample arrays (e.g., a luma and two chroma arrays in case of a color picture 17) or any other number and/or kind of arrays depending on the color format applied. The number of samples in horizontal and vertical direction (or axis) of the block 203 define the size of block 203. Accordingly, a block may, for example, be an MxN (M-column by N-row) array of samples, or an MxN array of transform coefficients.

Embodiments of the video encoder 20 as shown in Fig. 2 may be configured encode the picture 17 block by block, e.g., the encoding and prediction is performed per block 203.

Residual Calculation

The residual calculation unit 204 may be configured to calculate a residual block 205 (also referred to as residual 205) based on the picture block 203 and a prediction block 265 (further details about the prediction block 265 are provided later), e.g., by subtracting sample values of the prediction block 265 from sample values of the picture block 203, sample by sample (pixel by pixel) to obtain the residual block 205 in the sample domain.

Transform

The transform processing unit 206 may be configured to apply a transform, e.g., a discrete cosine transform (DCT) or discrete sine transform (DST), on the sample values of the residual block 205 to obtain transform coefficients 207 in a transform domain. The transform coefficients 207 may also be referred to as transform residual coefficients and represent the residual block 205 in the transform domain.

The transform processing unit 206 may be configured to apply integer approximations of DCT/DST, such as the transforms specified for H.265/HEVC. Compared to an orthogonal DCT transform, such integer approximations are typically scaled by a certain factor. In order to preserve the norm of the residual block which is processed by forward and inverse transforms, additional scaling factors are applied as part of the transform process. The scaling factors are typically chosen based on certain constraints like scaling factors being a power of two for shift operations, bit depth of the transform coefficients, tradeoff between accuracy and implementation costs, etc. Specific scaling factors are, for example, specified for the inverse transform, e.g., by inverse transform processing unit 212 (and the corresponding inverse transform, e.g., by inverse transform processing unit 312 at video decoder 30) and corresponding scaling factors for the forward transform, e.g., by transform processing unit 206, at an encoder 20 may be specified accordingly.

Embodiments of the video encoder 20 (respectively transform processing unit 206) may be configured to output transform parameters, e.g., a type of transform or transforms, e.g., directly or encoded or compressed via the entropy encoding unit 270, so that, e.g., the video decoder 30 may receive and use the transform parameters for decoding.

Quantization

The quantization unit 208 may be configured to quantize the transform coefficients 207 to obtain quantized coefficients 209, e.g., by applying scalar quantization or vector quantization. The quantized coefficients 209 may also be referred to as quantized transform coefficients 209 or quantized residual coefficients 209.

The quantization process may reduce the bit depth associated with some or all of the transform coefficients 207. For example, an n-bit transform coefficient may be rounded down to an m-bit Transform coefficient during quantization, where n is greater than m. The degree of quantization may be modified by adjusting a quantization parameter (QP). For example for scalar quantization, different scaling may be applied to achieve finer or coarser quantization. Smaller quantization step sizes correspond to finer quantization, whereas larger quantization step sizes correspond to coarser quantization. The applicable quantization step size may be indicated by a quantization parameter (QP). The quantization parameter may for example be an index to a predefined set of applicable quantization step sizes. For example, small quantization parameters may correspond to fine quantization (small quantization step sizes) and large quantization parameters may correspond to coarse quantization (large quantization step sizes) or vice versa. The quantization may include division by a quantization step size and a corresponding and/or the inverse dequantization, e.g., by inverse quantization unit 210, may include multiplication by the quantization step size. Embodiments according to some standards, e.g., HEVC, may be configured to use a quantization parameter to determine the quantization step size. Generally, the quantization step size may be calculated based on a quantization parameter using a fixed point approximation of an equation including division. Additional scaling factors may be introduced for quantization and dequantization to restore the norm of the residual block, which might get modified because of the scaling used in the fixed point approximation of the equation for quantization step size and quantization parameter. In one example implementation, the scaling of the inverse transform and dequantization might be combined. Alternatively, customized quantization tables may be used and signaled from an encoder to a decoder, e.g., in a bitstream. The quantization is a lossy operation, wherein the loss increases with increasing quantization step sizes.

Embodiments of the video encoder 20 (respectively quantization unit 208) may be configured to output quantization parameters (QP), e.g., directly or encoded via the entropy encoding unit 270, so that, e.g., the video decoder 30 may receive and apply the quantization parameters for decoding.

Inverse Quantization

The inverse quantization unit 210 is configured to apply the inverse quantization of the quantization unit 208 on the quantized coefficients to obtain dequantized coefficients 211, e.g., by applying the inverse of the quantization scheme applied by the quantization unit 208 based on or using the same quantization step size as the quantization unit 208. The dequantized coefficients 211 may also be referred to as dequantized residual coefficients 211 and correspond - although typically not identical to the transform coefficients due to the loss by quantization - to the transform coefficients 207.

Inverse Transform

The inverse transform processing unit 212 is configured to apply the inverse transform of the transform applied by the transform processing unit 206, e.g., an inverse discrete cosine transform (DCT) or inverse discrete sine transform (DST) or other inverse transforms, to obtain a reconstructed residual block 213 (or corresponding dequantized coefficients 213) in the sample domain. The reconstructed residual block 213 may also be referred to as transform block 213.

Reconstruction

The reconstruction unit 214 (e.g., adder or summer 214) is configured to add the transform block 213 (i.e. reconstructed residual block 213) to the prediction block 265 to obtain a reconstructed block 215 in the sample domain, e.g., by adding – sample by sample - the sample values of the reconstructed residual block 213 and the sample values of the prediction block 265.

Filtering

The loop filter unit 220 (or short “loop filter” 220), is configured to filter the reconstructed block 215 to obtain a filtered block 221, or in general, to filter reconstructed samples to obtain filtered samples. The loop filter unit is, e.g., configured to smooth pixel transitions, or otherwise improve the video quality. The loop filter unit 220 may comprise one or more loop filters such as a de-blocking filter, a sample-adaptive offset (SAO) filter or one or more other filters, e.g., a bilateral filter, an adaptive loop filter (ALF), a sharpening, a smoothing filter or a collaborative filter, or any combination thereof. Although the loop filter unit 220 is shown in Fig. 2 as being an in loop filter, in other configurations, the loop filter unit 220 may be implemented as a post loop filter. The filtered block 221 may also be referred to as filtered reconstructed block 221.

Embodiments of the video encoder 20 (respectively loop filter unit 220) may be configured to output loop filter parameters (such as sample adaptive offset information), e.g., directly or encoded via the entropy encoding unit 270, so that, e.g., a decoder 30 may receive and apply the same loop filter parameters or respective loop filters for decoding.

Decoded Picture Buffer

The decoded picture buffer (DPB) 230 may be a memory that stores reference pictures, or in general reference picture data, for encoding video data by video encoder 20. The DPB 230 may be formed by any of a variety of memory devices, such as dynamic random access memory (DRAM), including synchronous DRAM (SDRAM), magnetoresistive RAM (MRAM), resistive RAM (RRAM), or other types of memory devices. The decoded picture buffer (DPB) 230 may be configured to store one or more filtered blocks 221. The decoded picture buffer 230 may be further configured to store other previously filtered blocks, e.g., previously reconstructed and filtered blocks 221, of the same current picture or of different pictures, e.g., previously reconstructed pictures, and may provide complete previously reconstructed, i.e. decoded, pictures (and corresponding reference blocks and samples) and/or a partially reconstructed current picture (and corresponding reference blocks and samples), for example for inter prediction. The decoded picture buffer (DPB) 230 may be also configured to store one or more unfiltered reconstructed blocks 215, or in general unfiltered reconstructed samples, e.g., if the reconstructed block 215 is not filtered by loop filter unit 220, or any other further processed version of the reconstructed blocks or samples.

Mode Selection (Partitioning & Prediction)

The mode selection unit 260 comprises partitioning unit 262, inter-prediction unit 244 and intra-prediction unit 254, and is configured to receive or obtain original picture data, e.g., an original block 203 (current block 203 of the current picture 17), and reconstructed picture data, e.g., filtered and/or unfiltered reconstructed samples or blocks of the same (current) picture and/or from one or a plurality of previously decoded pictures, e.g., from decoded picture buffer 230 or other buffers (e.g., line buffer, not shown).. The reconstructed picture data is used as reference picture data for prediction, e.g., inter-prediction or intra-prediction, to obtain a prediction block 265 or predictor 265.

Mode selection unit 260 may be configured to determine or select a partitioning for a current block prediction mode (including no partitioning) and a prediction mode (e.g., an intra or inter prediction mode) and generate a corresponding prediction block 265, which is used for the calculation of the residual block 205 and for the reconstruction of the reconstructed block 215.

Embodiments of the mode selection unit 260 may be configured to select the partitioning and the prediction mode (e.g., from those supported by or available for mode selection unit 260), which provide the best match or in other words the minimum residual (minimum residual means better compression for transmission or storage), or a minimum signaling overhead (minimum signaling overhead means better compression for transmission or storage), or which considers or balances both. The mode selection unit 260 may be configured to determine the partitioning and prediction mode based on rate distortion optimization (RDO), i.e. select the prediction mode which provides a minimum rate distortion. Terms like “best”, “minimum”, “optimum” etc. in this context do not necessarily refer to an overall “best”, “minimum”, “optimum”, etc. but may also refer to the fulfillment of a termination or selection criterion like a value exceeding or falling below a threshold or other constraints leading potentially to a “sub-optimum selection” but reducing complexity and processing time.

In other words, the partitioning unit 262 may be configured to partition the block 203 into smaller block partitions or sub-blocks (which form again blocks), e.g., iteratively using quad-tree-partitioning (QT), binary partitioning (BT) or triple-tree-partitioning (TT) or any combination thereof, and to perform, e.g., the prediction for each of the block partitions or sub-blocks, wherein the mode selection comprises the selection of the tree-structure of the partitioned block 203 and the prediction modes are applied to each of the block partitions or sub-blocks.

In the following the partitioning (e.g., by partitioning unit 260) and prediction processing (by inter-prediction unit 244 and intra-prediction unit 254) performed by an example video encoder 20 will be explained in more detail.

Partitioning

The partitioning unit 262 may partition (or split) a current block 203 into smaller partitions, e.g., smaller blocks of square or rectangular size. These smaller blocks (which may also be referred to as sub-blocks) may be further partitioned into even smaller partitions. This is also referred to tree-partitioning or hierarchical tree-partitioning, wherein a root block, e.g., at root tree-level 0 (hierarchy-level 0, depth 0), may be recursively partitioned, e.g., partitioned into two or more blocks of a next lower tree-level, e.g., nodes at tree-level 1 (hierarchy-level 1, depth 1), wherein these blocks may be again partitioned into two or more blocks of a next lower level, e.g., tree-level 2 (hierarchy-level 2, depth 2), etc. until the partitioning is terminated, e.g., because a termination criterion is fulfilled, e.g., a maximum tree depth or minimum block size is reached. Blocks which are not further partitioned are also referred to as leaf-blocks or leaf nodes of the tree. A tree using partitioning into two partitions is referred to as binary-tree (BT), a tree using partitioning into three partitions is referred to as ternary-tree (TT), and a tree using partitioning into four partitions is referred to as quad-tree (QT).

As mentioned before, the term “block” as used herein may be a portion, in particular a square or rectangular portion, of a picture. With reference, for example, to HEVC and VVC, the block may be or correspond to a coding tree unit (CTU), a coding unit (CU), prediction unit (PU), and transform unit (TU) and/or to the corresponding blocks, e.g., a coding tree block (CTB), a coding block (CB), a transform block (TB) or prediction block (PB).

For example, a coding tree unit (CTU) may be or comprise a CTB of luma samples, two corresponding CTBs of chroma samples of a picture that has three sample arrays, or a CTB of samples of a monochrome picture or a picture that is coded using three separate colour planes and syntax structures used to code the samples. Correspondingly, a coding tree block (CTB) may be an NxN block of samples for some value of N such that the division of a component into CTBs is a partitioning. A coding unit (CU) may be or comprise a coding block of luma samples, two corresponding coding blocks of chroma samples of a picture that has three sample arrays, or a coding block of samples of a monochrome picture or a picture that is coded using three separate colour planes and syntax structures used to code the samples. Correspondingly a coding block (CB) may be an MxN block of samples for some values of M and N such that the division of a CTB into coding blocks is a partitioning.

In embodiments, e.g., according to HEVC, a coding tree unit (CTU) may be split into CUs by using a quad-tree structure denoted as coding tree. The decision whether to code a picture area using inter-picture (temporal) or intra-picture (spatial) prediction is made at the CU level. Each CU can be further split into one, two or four PUs according to the PU splitting type. Inside one PU, the same prediction process is applied and the relevant information is transmitted to the decoder on a PU basis. After obtaining the residual block by applying the prediction process based on the PU splitting type, a CU can be partitioned into transform units (TUs) according to another quadtree structure similar to the coding tree for the CU.

In embodiments, e.g., according to the latest video coding standard currently in development, which is referred to as Versatile Video Coding (VVC), Quad-tree and binary tree (QTBT) partitioning is used to partition a coding block. In the QTBT block structure, a CU can have either a square or rectangular shape. For example, a coding tree unit (CTU) is first partitioned by a quadtree structure. The quadtree leaf nodes are further partitioned by a binary tree or ternary (or triple) tree structure. The partitioning tree leaf nodes are called coding units (CUs), and that segmentation is used for prediction and transform processing without any further partitioning. This means that the CU, PU and TU have the same block size in the QTBT coding block structure. In parallel, multiple partition, for example, triple tree partition was also proposed to be used together with the QTBT block structure.

In one example, the mode selection unit 260 of video encoder 20 may be configured to perform any combination of the partitioning techniques described herein.

As described above, the video encoder 20 is configured to determine or select the best or an optimum prediction mode from a set of (pre-determined) prediction modes. The set of prediction modes may comprise, e.g., intra-prediction modes and/or inter-prediction modes.

Intra-Prediction

The set of intra-prediction modes may comprise 35 different intra-prediction modes, e.g., non-directional modes like DC (or mean) mode and planar mode, or directional modes, e.g., as defined

in HEVC, or may comprise 67 different intra-prediction modes, e.g., non-directional modes like DC (or mean) mode and planar mode, or directional modes, e.g., as defined for VVC.

The intra-prediction unit 254 is configured to use reconstructed samples of neighboring blocks of the same current picture to generate an intra-prediction block 265 according to an intra-prediction mode of the set of intra-prediction modes.

The intra prediction unit 254 (or in general the mode selection unit 260) is further configured to output intra-prediction parameters (or in general information indicative of the selected intra prediction mode for the block) to the entropy encoding unit 270 in form of syntax elements 266 for inclusion into the encoded picture data 21, so that, e.g., the video decoder 30 may receive and use the prediction parameters for decoding.

Inter-Prediction

The set of (or possible) inter-prediction modes depends on the available reference pictures (i.e. previous at least partially decoded pictures, e.g., stored in DBP 230) and other inter-prediction parameters, e.g., whether the whole reference picture or only a part, e.g., a search window area around the area of the current block, of the reference picture is used for searching for a best matching reference block, and/or e.g., whether pixel interpolation is applied, e.g., half/semi-pel and/or quarter-pel interpolation, or not.

Additional to the above prediction modes, skip mode and/or direct mode may be applied.

The inter prediction unit 244 may include a motion estimation (ME) unit and a motion compensation (MC) unit (both not shown in Fig. 2). The motion estimation unit may be configured to receive or obtain the picture block 203 (current picture block 203 of the current picture 17) and a decoded picture 231, or at least one or a plurality of previously reconstructed blocks, e.g., reconstructed blocks of one or a plurality of other/different previously decoded pictures 231, for motion estimation. E.g., a video sequence may comprise the current picture and the previously decoded pictures 231, or in other words, the current picture and the previously decoded pictures 231 may be part of or form a sequence of pictures forming a video sequence.

The encoder 20 may, e.g., be configured to select a reference block from a plurality of reference blocks of the same or different pictures of the plurality of other pictures and provide a reference picture (or reference picture index) and/or an offset (spatial offset) between the position (x, y coordinates) of the reference block and the position of the current block as inter prediction parameters to the motion estimation unit. This offset is also called motion vector (MV).

The motion compensation unit is configured to obtain, e.g., receive, an inter prediction parameter and to perform inter prediction based on or using the inter prediction parameter to obtain an inter prediction block 265. Motion compensation, performed by the motion compensation unit, may involve fetching or generating the prediction block based on the motion/block vector determined by motion estimation, possibly performing interpolations to sub-pixel precision. Interpolation filtering may generate additional pixel samples from known pixel samples, thus potentially increasing the number of candidate prediction blocks that may be used to code a picture block. Upon receiving the motion vector for the PU of the current picture block, the motion compensation unit may locate the prediction block to which the motion vector points in one of the reference picture lists.

Motion compensation unit may also generate syntax elements associated with the blocks and the video slice for use by video decoder 30 in decoding the picture blocks of the video slice.

Entropy Coding

The entropy encoding unit 270 is configured to apply, for example, an entropy encoding algorithm or scheme (e.g., a variable length coding (VLC) scheme, an context adaptive VLC scheme (CAVLC), an arithmetic coding scheme, a binarization, a context adaptive binary arithmetic coding (CABAC), syntax-based context-adaptive binary arithmetic coding (SBAC), probability interval partitioning entropy (PIPE) coding or another entropy encoding methodology or technique) or bypass (no compression) on the quantized coefficients 209, inter prediction parameters, intra prediction parameters, loop filter parameters and/or other syntax elements to obtain encoded picture data 21 which can be output via the output 272, e.g., in the form of an encoded bitstream 21, so that, e.g., the video decoder 30 may receive and use the parameters for decoding, . The encoded bitstream 21 may be transmitted to video decoder 30, or stored in a memory for later transmission or retrieval by video decoder 30.

Other structural variations of the video encoder 20 can be used to encode the video stream. For example, a non-transform based encoder 20 can quantize the residual signal directly without the transform processing unit 206 for certain blocks or frames. In another implementation, an encoder 20 can have the quantization unit 208 and the inverse quantization unit 210 combined into a single unit.

Decoder and Decoding Method

Fig. 3 shows an example of a video decoder 30 that is configured to implement the techniques of this present application**.** The video decoder 30 is configured to receive encoded picture data 21 (e.g., encoded bitstream 21), e.g., encoded by encoder 20, to obtain a decoded picture 331. The encoded picture data or bitstream comprises information for decoding the encoded picture data, e.g., data that represents picture blocks of an encoded video slice and associated syntax elements.

In the example of Fig. 3, the decoder 30 comprises an entropy decoding unit 304, an inverse quantization unit 310, an inverse transform processing unit 312, a reconstruction unit 314 (e.g., a summer 314), a loop filter 320, a decoded picture buffer (DBP) 330, an inter prediction unit 344 and an intra prediction unit 354. Inter prediction unit 344 may be or include a motion compensation unit. Video decoder 30 may, in some examples, perform a decoding pass generally reciprocal to the encoding pass described with respect to video encoder 100 from Fig. 2.

As explained with regard to the encoder 20, the inverse quantization unit 210, the inverse transform processing unit 212, the reconstruction unit 214 the loop filter 220, the decoded picture buffer (DPB) 230, the inter prediction unit 344 and the intra prediction unit 354 are also referred to as forming the “built-in decoder” of video encoder 20. Accordingly, the inverse quantization unit 310 may be identical in function to the inverse quantization unit 110, the inverse transform processing unit 312 may be identical in function to the inverse transform processing unit 212, the reconstruction unit 314 may be identical in function to reconstruction unit 214, the loop filter 320 may be identical in function to the loop filter 220, and the decoded picture buffer 330 may be identical in function to the decoded picture buffer 230. Therefore, the explanations provided for the respective units and functions of the video 20 encoder apply correspondingly to the respective units and functions of the video decoder 30.

Entropy Decoding

The entropy decoding unit 304 is configured to parse the bitstream 21 (or in general encoded picture data 21) and perform, for example, entropy decoding to the encoded picture data 21 to obtain, e.g., quantized coefficients 309 and/or decoded coding parameters (not shown in Fig. 3), e.g., any or all of inter prediction parameters (e.g., reference picture index and motion vector), intra prediction parameter (e.g., intra prediction mode or index), transform parameters, quantization parameters, loop filter parameters, and/or other syntax elements. Entropy decoding unit 304 maybe configured to apply the decoding algorithms or schemes corresponding to the encoding schemes as described with regard to the entropy encoding unit 270 of the encoder 20. Entropy decoding unit 304 may be further configured to provide inter prediction parameters, intra prediction parameter and/or other syntax elements to the mode selection unit 360 and other parameters to other units of the decoder 30. Video decoder 30 may receive the syntax elements at the video slice level and/or the video block level.

Inverse Quantization

The inverse quantization unit 310 may be configured to receive quantization parameters (QP) (or in general information related to the inverse quantization) and quantized coefficients from the encoded picture data 21 (e.g., by parsing and/or decoding, e.g., by entropy decoding unit 304) and to apply based on the quantization parameters an inverse quantization on the decoded quantized coefficients 309 to obtain dequantized coefficients 311, which may also be referred to as transform coefficients 311. The inverse quantization process may include use of a quantization parameter determined by video encoder 20 for each video block in the video slice to determine a degree of quantization and, likewise, a degree of inverse quantization that should be applied.

Inverse Transform

Inverse transform processing unit 312 may be configured to receive dequantized coefficients 311, also referred to as transform coefficients 311, and to apply a transform to the dequantized coefficients 311 in order to obtain reconstructed residual blocks 213 in the sample domain. The reconstructed residual blocks 313 may also be referred to as transform blocks 313. The transform may be an inverse transform, e.g., an inverse DCT, an inverse DST, an inverse integer transform, or a conceptually similar inverse transform process. The inverse transform processing unit 312

may be further configured to receive transform parameters or corresponding information from the encoded picture data 21 (e.g., by parsing and/or decoding, e.g., by entropy decoding unit 304) to determine the transform to be applied to the dequantized coefficients 311.

Reconstruction

The reconstruction unit 314 (e.g., adder or summer 314) may be configured to add the reconstructed residual block 313, to the prediction block 365 to obtain a reconstructed block 315 in the sample domain, e.g., by adding the sample values of the reconstructed residual block 313 and the sample values of the prediction block 365.

Filtering

The loop filter unit 320 (either in the coding loop or after the coding loop) is configured to filter the reconstructed block 315 to obtain a filtered block 321, e.g., to smooth pixel transitions, or otherwise improve the video quality. The loop filter unit 320 may comprise one or more loop filters such as a de-blocking filter, a sample-adaptive offset (SAO) filter or one or more other filters, e.g., a bilateral filter, an adaptive loop filter (ALF), a sharpening, a smoothing filters or a collaborative filters, or any combination thereof. Although the loop filter unit 320 is shown in Fig. 3 as being an in loop filter, in other configurations, the loop filter unit 320 may be implemented as a post loop filter.

Decoded Picture Buffer

The decoded video blocks 321 of a picture are then stored in decoded picture buffer 330, which stores the decoded pictures 331 as reference pictures for subsequent motion compensation for other pictures and/or for output respectively display.

The decoder 30 is configured to output the decoded picture 311, e.g., via output 312, for presentation or viewing to a user.

Prediction

The inter prediction unit 344 may be identical to the inter prediction unit 244 (in particular to the motion compensation unit) and the intra prediction unit 354 may be identical to the inter

prediction unit 254 in function, and performs split or partitioning decisions and prediction based on the partitioning and/or prediction parameters or respective information received from the encoded picture data 21 (e.g., by parsing and/or decoding, e.g., by entropy decoding unit 304). Mode selection unit 360 may be configured to perform the prediction (intra or inter prediction) per block based on reconstructed pictures, blocks or respective samples (filtered or unfiltered) to obtain the prediction block 365.

When the video slice is coded as an intra coded (I) slice, intra prediction unit 354 of mode selection unit 360 is configured to generate prediction block 365 for a picture block of the current video slice based on a signaled intra prediction mode and data from previously decoded blocks of the current picture. When the video picture is coded as an inter coded (i.e., B, or P) slice, inter prediction unit 344 (e.g., motion compensation unit) of mode selection unit 360 is configured to produce prediction blocks 365 for a video block of the current video slice based on the motion vectors and other syntax elements received from entropy decoding unit 304. For inter prediction, the prediction blocks may be produced from one of the reference pictures within one of the reference picture lists. Video decoder 30 may construct the reference frame lists, List 0 and List 1, using default construction techniques based on reference pictures stored in DPB 330.

Mode selection unit 360 is configured to determine the prediction information for a video block of the current video slice by parsing the motion vectors and other syntax elements, and uses the prediction information to produce the prediction blocks for the current video block being decoded. For example, the mode selection unit 360 uses some of the received syntax elements to determine a prediction mode (e.g., intra or inter prediction) used to code the video blocks of the video slice, an inter prediction slice type (e.g., B slice, P slice, or GPB slice), construction information for one or more of the reference picture lists for the slice, motion vectors for each inter encoded video block of the slice, inter prediction status for each inter coded video block of the slice, and other information to decode the video blocks in the current video slice.

Other variations of the video decoder 30 can be used to decode the encoded picture data 21. For example, the decoder 30 can produce the output video stream without the loop filtering unit 320. For example, a non-transform based decoder 30 can inverse-quantize the residual signal

directly without the inverse-transform processing unit 312 for certain blocks or frames. In another implementation, the video decoder 30 can have the inverse-quantization unit 310 and the inverse-transform processing unit 312 combined into a single unit.

It should be understood that, in the encoder 20 and the decoder 30, a processing result of a current step may be further processed and then output to the next step. For example, after interpolation filtering, motion vector derivation or loop filtering, a further operation, such as Clip or shift, may be performed on the processing result of the interpolation filtering, motion vector derivation or loop filtering.

It should be noted that further operations may be applied to the derived motion vectors of current block (including but not limited to control point motion vectors of affine mode, sub-block motion vectors in affine, planar, ATMVP (advanced temporal motion vector prediction) modes, temporal motion vectors, and so on). For example, the value of motion vector is constrained to a predefined range according to its representing bit. If the representing bit of motion vector is bitDepth, then the range is -2^(bitDepth-1) ~ 2^(bitDepth-1)-1, where "^" means exponentiation. For example, if bitDepth is set equal to 16, the range is -32768 ~ 32767; if bitDepth is set equal to 18, the range is -131072~131071. Here provides two methods for constraining the motion vector.

Method 1: remove the overflow MSB (most significant bit) by flowing operations

$$ux = (mvx + 2^{bitDepth}) \% 2^{bitDepth} \quad (1)$$

$$mvx = (ux >= 2^{bitDepth-1}) ? (ux - 2^{bitDepth}) : ux \quad (2)$$

$$uy = (mvy + 2^{bitDepth}) \% 2^{bitDepth} \quad (3)$$

$$mvy = (uy >= 2^{bitDepth-1}) ? (uy - 2^{bitDepth}) : uy \quad (4)$$

For example, if the value of mvx is -32769, after applying formula (1) and (2), the resulting value is 32767. In computer system, negative decimal numbers are stored as two's complement. The two's complement of -32769 is 1,0111,1111,1111,1111 (17 bits), then the MSB is discarded, so the resulting two's complement is 0111,1111,1111,1111 (decimal number is 32767), which is same as the output by applying formula (1) and (2).

$$ux= (mvpx + mvdx +2^{bitDepth}) \% 2^{bitDepth} \quad (5)$$

$$mvx = (ux >= 2^{bitDepth-1}) ? (ux - 2^{bitDepth}) : ux \quad (6)$$

$$uy= (mvpy + mvdy +2^{bitDepth}) \% 2^{bitDepth} \quad (7)$$

$$mvy = (uy >= 2^{bitDepth-1}) ? (uy - 2^{bitDepth}) : uy \quad (8)$$

The operations may be applied during the sum of mvp (motion vector predictor) and mvd (motion vector difference), as shown in formula (5) to (8).

Method 2: remove the overflow MSB by clipping the value

$$vx = Clip3(-2^{bitDepth-1}, 2^{bitDepth-1} -1, vx)$$

$$vy = Clip3(-2^{bitDepth-1}, 2^{bitDepth-1} -1, vy)$$

where the definition of function Clip3 is as follow:

$$Clip3(x, y, z) = \begin{cases} x & ; \quad z < x \\ y & ; \quad z > y \\ z & ; \quad \text{otherwise} \end{cases}$$

Fig. 4 is a schematic diagram of a video coding device 400 according to an embodiment of the disclosure. The video coding device 400 is suitable for implementing the disclosed embodiments as described herein. In an embodiment, the video coding device 400 may be a decoder such as video decoder 30 of Fig. 1A or an encoder such as video encoder 20 of Fig. 1A.

The video coding device 400 comprises ingress ports 410 (or input ports 410) and receiver units (Rx) 420 for receiving data; a processor, logic unit, or central processing unit (CPU) 430 to process the data; transmitter units (Tx) 440 and egress ports 450 (or output ports 450) for transmitting the data; and a memory 460 for storing the data. The video coding device 400 may also comprise optical-to-electrical (OE) components and electrical-to-optical (EO) components coupled to the ingress ports 410, the receiver units 420, the transmitter units 440, and the egress ports 450 for egress or ingress of optical or electrical signals.

The processor 430 is implemented by hardware and software. The processor 430 may be implemented as one or more CPU chips, cores (e.g., as a multi-core processor), FPGAs, ASICs, and DSPs. The processor 430 is in communication with the ingress ports 410, receiver units 420, transmitter units 440, egress ports 450, and memory 460. The processor 430 comprises a coding module 470. The coding module 470 implements the disclosed embodiments described above. For instance, the coding module 470 implements, processes, prepares, or provides the various coding operations. The inclusion of the coding module 470 therefore provides a substantial improvement to the functionality of the video coding device 400 and effects a transformation of the video coding device 400 to a different state. Alternatively, the coding module 470 is implemented as instructions stored in the memory 460 and executed by the processor 430.

The memory 460 may comprise one or more disks, tape drives, and solid-state drives and may be used as an over-flow data storage device, to store programs when such programs are selected for execution, and to store instructions and data that are read during program execution. The memory 460 may be, for example, volatile and/or non-volatile and may be a read-only memory (ROM), random access memory (RAM), ternary content-addressable memory (TCAM), and/or static random-access memory (SRAM).

Fig. 5 is a simplified block diagram of an apparatus 500 that may be used as either or both of the source device 12 and the destination device 14 from Fig. 1 according to an exemplary embodiment**.**

A processor 502 in the apparatus 500 can be a central processing unit. Alternatively, the processor 502 can be any other type of device, or multiple devices, capable of manipulating or processing information now-existing or hereafter developed. Although the disclosed implementations can be practiced with a single processor as shown, e.g., the processor 502, advantages in speed and efficiency can be achieved using more than one processor.

A memory 504 in the apparatus 500 can be a read only memory (ROM) device or a random access memory (RAM) device in an implementation. Any other suitable type of storage device can be used as the memory 504. The memory 504 can include code and data 506 that is accessed by

the processor 502 using a bus 512. The memory 504 can further include an operating system 508 and application programs 510, the application programs 510 including at least one program that permits the processor 502 to perform the methods described here. For example, the application programs 510 can include applications 1 through N, which further include a video coding application that performs the methods described here.

The apparatus 500 can also include one or more output devices, such as a display 518. The display 518 may be, in one example, a touch sensitive display that combines a display with a touch sensitive element that is operable to sense touch inputs. The display 518 can be coupled to the processor 502 via the bus 512.

Although depicted here as a single bus, the bus 512 of the apparatus 500 can be composed of multiple buses. Further, the secondary storage 514 can be directly coupled to the other components of the apparatus 500 or can be accessed via a network and can comprise a single integrated unit such as a memory card or multiple units such as multiple memory cards. The apparatus 500 can thus be implemented in a wide variety of configurations.

In recent development of video coding, more sophisticated techniques and schemes for prediction have emerged.

One such technique is multi-hypothesis prediction. Initially, the term “multi-hypothesis prediction” was introduced to extend motion compensation with one prediction signal to the linear superposition of several motion-compensated prediction signals. More recently, this approach has been generalized to combining an existing prediction mode with an extra merge indexed prediction. This includes, in particular, multi-hypothesis prediction for intra and inter mode (or, in other words, combining intra and inter mode, cf., for instance, Joint Video Experts Team (JVET), document JVET-L0100-v3, entitled “CE10.1.1: Multi-hypothesis prediction for improving AMVP mode, skip or merge mode, and intra mode”, 12th Meeting: Macao, CN, 3-12 Oct. 2018). This approach applies multi-hypothesis prediction to improve intra mode, by combining one intra prediction and one merge indexed prediction. In other words, a linear combination of both an intra and an inter prediction block is used.

Another technique is to introduce and use separate merge lists for sub-block merge candidates, i.e. separate from merge candidate lists for the regular merge mode, on the block level (cf., for instance, Joint Video Experts Team (JVET), document JVET-L0369-v2, entitled “CE4: Separate list for sub-block merge candidates (Test 4.2.8))”, 12th Meeting: Macao, CN, 3-12 Oct. 2018). This technique is particularly applicable to the above mentioned ATMVP and affine modes.

Multi-hypothesis prediction for intra and inter mode usage is controlled by mh_intra_flag, which specifies whether multi-hypothesis prediction for intra and inter mode is enabled or not for a current block. In the prior art, the flag mh_intra_flag is transmitted conditionally depending on merge_affine_flag according to following syntax table:

coding_unit(x0, y0, cbWidth, cbHeight, treeType) {	**Descriptor**
…	
} else { /* MODE_INTER */	
if(cu_skip_flag[x0][y0]) {	
if(… && cbWidth >= 8 && cbHeight >= 8)	
merge_affine_flag[x0][y0]	ae(v)
if(merge_affine_flag[x0][y0] = = 0 && MaxNumMergeCand > 1) {	
if(mvd_l1_zero_flag == 0)	
mh_merge_flag[x0][y0]	ae(v)
merge_idx[x0][y0]	ae(v)
}	
} else {	
merge_flag[x0][y0]	ae(v)
if(merge_flag[x0][y0]) {	
if(sps_affine_enabled_flag && cbWidth >= 8 && cbHeight >= 8 && (MotionModelIdc[x0 − 1][y0 + cbHeight − 1] != 0 || MotionModelIdc[x0 − 1][y0 + cbHeight] != 0 || MotionModelIdc[x0 − 1][y0 − 1] != 0 || MotionModelIdc[x0 + cbWidth − 1][y0 − 1] != 0 || MotionModelIdc[x0 + cbWidth][y0 − 1]] != 0))	
merge_affine_flag[x0][y0]	ae(v)
if(merge_affine_flag[x0][y0] = = 0 && MaxNumMergeCand > 1)	

mh_merge_flag[x0][y0]	ae(v)
if(merge_affine_flag[x0][y0] = = 0 && mh_merge_flag[x0][y0] == 0)	
mh_intra_flag[x0][y0]	ae(v)
if(mh_intra_flag[x0][y0])	
mh_intra_mode_idx [x0][y0]	
if(merge_affine_flag[x0][y0] = = 0 && MaxNumMergeCand > 1)	
merge_idx[x0][y0]	ae(v)
} else {	
…	
}	

Thus, mh_intra_flag was conditionally signaled under the condition that merge_affine_flag is zero. This means that joint usage of affine merge with mh_intra_flag, i.e. multi-hypothesis prediction for intra and inter mode, is disabled. However, it does allow usage of ATMVP with mh_intra_flag.

After recent adoption of separate merge list for subblock merge candidates, affine merge candidates were coupled together with atmvp merge candidates and usage of these subblock candidates is controlled by a newly introduced parameter called merge_subblock_flag. No separate merge_affine flag is used any longer. Thus the syntax does not allow determining a case when merge affine mode is not used but atmvp is used any longer. A new way of signaling multi-hypothesis prediction combining intra and inter mode in the presence of separate merge lists for subblock merge candidates has been developed in the framework of the present invention.

The invention proposes plural methods of usage and signaling multi-hypothesis prediction for intra and inter mode assuming the presence of separate merge lists for subblock merge candidates in the codec. Plural ways of harmonization are possible which are included in the current disclosure. Also usage of multi-hypothesis prediction for intra and inter mode in case of skip mode is disclosed.

According to one general aspect of the present disclosure, a method of video encoding of video data into a bitstream and a method of video decoding of video data received in a bitstream are

provided.

The method of video encoding comprises applying a first technique and/or a second technique. The first technique comprises using a separate merge list for subblock merge candidates. The second technique comprises multi-hypothesis prediction for intra and inter mode. The method comprises transmitting a first control flag in the bitstream, wherein the first control flag indicates whether to use the first technique and transmitting a second control flag in the bitstream, wherein the second control flag indicates whether to use the second technique.

The method of video decoding comprises applying a first technique and/or a second technique. The first technique comprises using a separate merge list for subblock merge candidates. The second technique comprises multi-hypothesis prediction for intra and inter mode. The method comprises receiving a first control flag from the bitstream, wherein the first control flag indicates whether to use the first technique and receiving a second control flag from the bitstream, wherein the second control flag indicates whether to use the second technique.

In accordance with embodiments, the usage of the technique of multi-hypothesis prediction for intra and inter mode is controlled independently of usage of the separate merge list for subblock merge candidates technique. In other words, in such embodiments, signaling of the first control flag (merge_subblock_flag) is performed independently of signaling of the second control flag (mh_intra_flag).

In one embodiment multi-hypothesis prediction for intra and inter mode is controlled by mh_intra_flag which is signaled without dependency of merge_subblock_flag. In this case multi-hypothesis prediction for intra and inter mode is possible for both subblock modes: affine and atmvp as well as for normal merge. Syntax tables below demonstrate possible ways of signaling mh_intra_flag in this embodiment.

.coding_unit(x0, y0, cbWidth, cbHeight, treeType) {	**Descriptor**
…	

} else { /* MODE_INTER */	
if(cu_skip_flag[x0][y0]) {	
if(… && cbWidth >= 8 && cbHeight >= 8)	
merge_subblock_flag[x0][y0]	ae(v)
if(merge_subblock_flag[x0][y0] = = 0 && MaxNumMergeCand > 1)	
merge_idx[x0][y0]	ae(v)
if(merge_subblock_flag[x0][y0] = = 1 && MaxNumSubblockMergeCand > 1)	
merge_subblock_idx[x0][y0]	ae(v)
} else {	
merge_flag[x0][y0]	ae(v)
if(merge_flag[x0][y0]) {	
if(… && cbWidth >= 8 && cbHeight >= 8)	
merge_subblock_flag[x0][y0]	ae(v)
if(merge_subblock_flag[x0][y0] = = 0 && MaxNumMergeCand > 1)	
merge_idx[x0][y0]	ae(v)
if(merge_subblock_flag[x0][y0] = = 1 && MaxNumSubblockMergeCand > 1)	
merge_subblock_idx[x0][y0]	ae(v)
mh_intra_flag[x0][y0]	ae(v)
} else {	
…	
}	

In the table above, use of multi-hypothesis prediction combining intra and intra mode is restricted to coding blocks in the merge mode.

coding_unit(x0, y0, cbWidth, cbHeight, treeType) {	**Descriptor**
…	
} else { /* MODE_INTER */	
if(cu_skip_flag[x0][y0]) {	
if(… && cbWidth >= 8 && cbHeight >= 8)	
merge_subblock_flag[x0][y0]	ae(v)

if(merge_subblock_flag[x0][y0] = = 0 && MaxNumMergeCand > 1)	
merge_idx[x0][y0]	ae(v)
if(merge_subblock_flag[x0][y0] = = 1 && MaxNumSubblockMergeCand > 1)	
merge_subblock_idx[x0][y0]	ae(v)
mh_intra_flag[x0][y0]	ae(v)
} else {	
merge_flag[x0][y0]	ae(v)
if(merge_flag[x0][y0]) {	
if(… && cbWidth >= 8 && cbHeight >= 8)	
merge_subblock_flag[x0][y0]	ae(v)
if(merge_subblock_flag[x0][y0] = = 0 && MaxNumMergeCand > 1)	
merge_idx[x0][y0]	ae(v)
if(merge_subblock_flag[x0][y0] = = 1 && MaxNumSubblockMergeCand > 1)	
merge_subblock_idx[x0][y0]	ae(v)
mh_intra_flag[x0][y0]	ae(v)
} else {	
…	
}	

In the preceding table, use of multi-hypothesis prediction combining intra and intra mode can be applied to both coding blocks in the merge mode and coding blocks in the skip mode.

In an embodiment of the present invention, multi-hypothesis prediction for intra and inter mode is controlled by mh_intra_flag which is signaled based on merge_subblock_flag. In this case multi-hypothesis prediction for intra and inter mode is possible only if both subblock modes: affine and atmvp are disabled. In other words, multi-hypothesis prediction for intra and inter modes is possible in normal merge mode but a combination of multi-hypothesis prediction for intra and inter mode with separate merge lists for subblock merge candidates is disabled. Syntax tables below demonstrate possible ways of signaling mh_intra_flag in this embodiment.

coding_unit(x0, y0, cbWidth, cbHeight, treeType) {	**Descriptor**
…	
} else { /* MODE_INTER */	
if(cu_skip_flag[x0][y0]) {	
if(… && cbWidth >= 8 && cbHeight >= 8)	
merge_subblock_flag[x0][y0]	ae(v)
if(merge_subblock_flag[x0][y0] = = 0 && MaxNumMergeCand > 1)	
merge_idx[x0][y0]	ae(v)
if(merge_subblock_flag[x0][y0] = = 1 && MaxNumSubblockMergeCand > 1)	
merge_subblock_idx[x0][y0]	ae(v)
} else {	
merge_flag[x0][y0]	ae(v)
if(merge_flag[x0][y0]) {	
if(… && cbWidth >= 8 && cbHeight >= 8)	
merge_subblock_flag[x0][y0]	ae(v)
if(merge_subblock_flag[x0][y0] = = 0 && MaxNumMergeCand > 1)	
merge_idx[x0][y0]	ae(v)
if(merge_subblock_flag[x0][y0] = = 1 && MaxNumSubblockMergeCand > 1)	
merge_subblock_idx[x0][y0]	ae(v)
if(merge_subblock_flag[x0][y0] = = 0)	
mh_intra_flag[x0][y0]	ae(v)
} else {	
…	
}	

In the table above, use of multi-hypothesis prediction combining intra and intra mode is restricted to coding blocks in the merge mode.

coding_unit(x0, y0, cbWidth, cbHeight, treeType) {	**Descriptor**
…	
} else { /* MODE_INTER */	

if(cu_skip_flag[x0][y0]) {	
if(… && cbWidth >= 8 && cbHeight >= 8)	
merge_subblock_flag[x0][y0]	ae(v)
if(merge_subblock_flag[x0][y0] = = 0 && MaxNumMergeCand > 1)	
merge_idx[x0][y0]	ae(v)
if(merge_subblock_flag[x0][y0] = = 1 && MaxNumSubblockMergeCand > 1)	
merge_subblock_idx[x0][y0]	ae(v)
if(merge_subblock_flag[x0][y0] = = 0)	
mh_intra_flag[x0][y0]	ae(v)
} else {	
merge_flag[x0][y0]	ae(v)
if(merge_flag[x0][y0]) {	
if(… && cbWidth >= 8 && cbHeight >= 8)	
merge_subblock_flag[x0][y0]	ae(v)
if(merge_subblock_flag[x0][y0] = = 0 && MaxNumMergeCand > 1)	
merge_idx[x0][y0]	ae(v)
if(merge_subblock_flag[x0][y0] = = 1 && MaxNumSubblockMergeCand > 1)	
merge_subblock_idx[x0][y0]	ae(v)
if(merge_subblock_flag[x0][y0] = = 0)	
mh_intra_flag[x0][y0]	ae(v)
} else {	
…	
}	

In the preceding table, use of multi-hypothesis prediction combining intra and intra mode can be applied to both coding blocks in the merge mode and coding blocks in the skip mode.

In the framework of the above embodiment, the encoder signals the mh_intra_flag (“second flag”, also called ciip_flag) only if separate merge lists for subblock candidates are disabled, i.e. if merge_subblock_flag (“first flag”) is zero. On the other hand, a decoder receiving a bitstream in which mh_intra_flag is included determines by default that separate merge lists for subblock candidates are disabled.

In the following, the processing in accordance with embodiments described above will be explained with reference to the flowcharts of Figs. 6 and 7.

Fig. 6 is a flowchart illustrating an encoder side process according to an embodiment of the present invention.

The process starts with a decision whether or not to use a separate merge list for subblock candidates, in step S101. If this is not the case (S101: No), processing proceeds to step S103. In step S103, it is decided, whether or not multi-hypothesis prediction combining intra and inter mode is used. Independently of the outcome of the decision, processing subsequently proceeds to step S105. In step S105, an mh_intra_flag is generated as a parameter to be signaled in the bitstream. More specifically, if multi-hypothesis prediction for intra and inter mode is used (S103: Yes), a value “1” is set to the mh_intra_flag and if multi-hypothesis prediction for intra and inter flag is not used (S103: No), a value “0” is set to the mh_intra_flag (not shown in the drawing).

Subsequently, processing proceeds to step S107. In the case when it is decided in step S101 to use a separate merge list for subblock candidates (S101: Yes), processing proceeds directly from step S101 to step S107.

In step S107, a merge_subblock_flag is generated as a parameter to be signaled in the bitstream. More specifically, in case of usage of a separate merge list, i.e. when the flow proceeds from S101: Yes, a value “1” is set to the merge_subblock_flaf and in case when the flow comes from S101: No through S103 and S105, i.e. when no separate merge list is used, the merge_subblock_flag is set to a value “0” (not shown in the drawing).

Hence, multi-hypothesis prediction for intra and inter mode is used only conditionally upon the condition that separate merge lists for subblock candidates are not used. Therefore, the merge_subblock_flag is always generated but an mh_intra_flag is generated only if no separate merge list for intra and inter mode is used. Subsequently, a bitstream including the merge_subblock_flag and conditionally (if the merge_subbock_flag is 0) also including the mh_intra_flag is generated in step S109 and the processing ends.

Fig. 7 is a flowchart illustrating a decoder side processing according to an embodiment of the present invention.

The processing starts with step S201, in which a received bitstream is parsed. In subsequent step S203, it is checked, whether or not there is an mh_intra_flag in the bitstream. If this is the case (S203: Yes)., processing proceeds to step S207. In step S207, the mh_intra_flag is evaluated in order to decide whether multi-hypothesis prediction for intra and inter mode is to be used in the dedecoding, depending on the value of the parsed mh_intra_flag. In case the mh_intra_flag has the value of 1, decoding is performed with using multi-hypothesis prediction for intra and inter mode in step (S207: Yes -> S211), and the processing ends. Otherwise, i.e. when the mh_intra_flag has a value of 0 (S207: No), multi-hypothesis prediction is not used (and neither is a separate merge list for subblock candidates).

If, on the other hand, it is determined in step S203 that there is no mh_intra flag (S203: No), processing proceeds to step S204. In step S204, a subblock_merge_flag received from the bitstream is evaluated in order to determine in the following step (S205) whether or not to use a separate merge list for subblock candidates.

In step S205, if the subblock_merge_flag has a value of 1, it is determined that the separate merge list is to be used (S205: Yes), and the processing proceeds to step S209, in which decoding is performed using the separate merge list. Otherwise, i.e. when the subblock_merge_flag has a value of 0 (S205: No), it is determined that decoding is performed without _using the separate merge list (and without the multi-hypothesis prediction for intra and inter mode), and the processing flow ends.

Hence, according to the embodiment, from the existence of the mh_intra_flag in the parsed bitstream (S203: Yes) it is directly inferred that separate merge lists for subblock candidates are not used.

In other words, the merge_subblock_flag, which is always included in the bitstream and received, is analyzed only in the case when no mh_intra_flag is included in the bitstream. Thus; in

any case, only a single flag needs to be evaluated on the decoder side.

More generally speaking, according to the present invention, usage of the technique of multi-hypothesis prediction for intra and inter mode is controlled based on usage of the separate merge list for subblock merge candidates technique. More specifically, in accordance with the embodiment of Fig. 7, the technique of multi-hypothesis prediction for intra and inter mode can be used if and only if separate merge list for subblock merge candidates technique is disabled.

The above mentioned embodiments allow to harmonize multi-hypothesis prediction for intra and inter mode with separate merge list for subblock merge candidates. Embodiments which also enable usage of multi-hypothesis prediction for intra and inter mode in block skip mode additionally allow to achieve coding gain in respect to basic design. Embodiments which include conditional signaling of mh_intra_flag based on usage of separate merge lists for subblock merge candidates have the additional benefit of reducing signaling overhead, because the mh_intra_flag does not need to be signaled in each case.

Mathematical Operators

The mathematical operators used in this application are similar to those used in the C programming language. However, the results of integer division and arithmetic shift operations are defined more precisely, and additional operations are defined, such as exponentiation and real-valued division. Numbering and counting conventions generally begin from 0, e.g., "the first" is equivalent to the 0-th, "the second" is equivalent to the 1-th, etc.

Arithmetic operators

The following arithmetic operators are defined as follows:

+	Addition
−	Subtraction (as a two-argument operator) or negation (as a unary prefix operator)
*	Multiplication, including matrix multiplication
x^y	Exponentiation. Specifies x to the power of y. In other contexts, such notation is used for superscripting not intended for interpretation as exponentiation.

/	Integer division with truncation of the result toward zero. For example, 7 / 4 and −7 / −4 are truncated to 1 and −7 / 4 and 7 / −4 are truncated to −1.
÷	Used to denote division in mathematical equations where no truncation or rounding is intended.
$\frac{x}{y}$	Used to denote division in mathematical equations where no truncation or rounding is intended.
$\sum_{i=x}^{y} f(i)$	The summation of f(i) with i taking all integer values from x up to and including y.
x % y	Modulus. Remainder of x divided by y, defined only for integers x and y with x >= 0 and y > 0.

Logical operators

The following logical operators are defined as follows:

x && y	Boolean logical "and" of x and y
x | | y	Boolean logical "or" of x and y
!	Boolean logical "not"
x ? y : z	If x is TRUE or not equal to 0, evaluates to the value of y; otherwise, evaluates to the value of z.

Relational operators

The following relational operators are defined as follows:

>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
= =	Equal to
!=	Not equal to

When a relational operator is applied to a syntax element or variable that has been assigned the value "na" (not applicable), the value "na" is treated as a distinct value for the syntax element or variable. The value "na" is considered not to be equal to any other value.

Bit-wise operators

The following bit-wise operators are defined as follows:

& Bit-wise "and". When operating on integer arguments, operates on a two's complement representation of the integer value. When operating on a binary argument that contains fewer bits than another argument, the shorter argument is extended by adding more significant bits equal to 0.

| Bit-wise "or". When operating on integer arguments, operates on a two's complement representation of the integer value. When operating on a binary argument that contains fewer bits than another argument, the shorter argument is extended by adding more significant bits equal to 0.

^ Bit-wise "exclusive or". When operating on integer arguments, operates on a two's complement representation of the integer value. When operating on a binary argument that contains fewer bits than another argument, the shorter argument is extended by adding more significant bits equal to 0.

x >> y Arithmetic right shift of a two's complement integer representation of x by y binary digits. This function is defined only for non-negative integer values of y. Bits shifted into the most significant bits (MSBs) as a result of the right shift have a value equal to the MSB of x prior to the shift operation.

x << y Arithmetic left shift of a two's complement integer representation of x by y binary digits. This function is defined only for non-negative integer values of y. Bits shifted into the least significant bits (LSBs) as a result of the left shift have a value equal to 0.

Assignment operators

The following arithmetic operators are defined as follows:

= Assignment operator

+ + Increment, i.e., *x*+ + is equivalent to *x* = *x* + 1; when used in an array index, evaluates to the value of the variable prior to the increment operation.

− − Decrement, i.e., *x*− − is equivalent to *x* = *x* − 1; when used in an array index, evaluates to the value of the variable prior to the decrement operation.

+= Increment by amount specified, i.e., x += 3 is equivalent to x = x + 3, and x += (−3) is equivalent to x = x + (−3).

−= Decrement by amount specified, i.e., x −= 3 is equivalent to x = x − 3, and x −= (−3) is equivalent to x = x − (−3).

Range notation

The following notation is used to specify a range of values:

x = y..z x takes on integer values starting from y to z, inclusive, with x, y, and z being integer numbers and z being greater than y.

Mathematical functions

The following mathematical functions are defined:

$$\text{Abs}(x) = \begin{cases} x & ; \quad x >= 0 \\ -x & ; \quad x < 0 \end{cases}$$

Asin(x) the trigonometric inverse sine function, operating on an argument x that is in the range of −1.0 to 1.0, inclusive, with an output value in the range of −π÷2 to π÷2, inclusive, in units of radians

Atan(x) the trigonometric inverse tangent function, operating on an argument x, with an output value in the range of −π÷2 to π÷2, inclusive, in units of radians

$$\text{Atan2}(y, x) = \begin{cases} \text{Atan}\left(\frac{y}{x}\right) & ; \quad x > 0 \\ \text{Atan}\left(\frac{y}{x}\right) + \pi & ; \quad x < 0 \ \&\& \ y >= 0 \\ \text{Atan}\left(\frac{y}{x}\right) - \pi & ; \quad x < 0 \ \&\& \ y < 0 \\ +\frac{\pi}{2} & ; \quad x == 0 \ \&\& \ y >= 0 \\ -\frac{\pi}{2} & ; \quad \text{otherwise} \end{cases}$$

Ceil(x) the smallest integer greater than or equal to x.

$$\text{Clip1}_Y(x) = \text{Clip3}(0, (1 << \text{BitDepth}_Y) - 1, x)$$

$$\text{Clip1}_C(x) = \text{Clip3}(0, (1 << \text{BitDepth}_C) - 1, x)$$

$$\text{Clip3}(x, y, z) = \begin{cases} x & ; \quad z < x \\ y & ; \quad z > y \\ z & ; \quad \text{otherwise} \end{cases}$$

Cos(x) the trigonometric cosine function operating on an argument x in units of radians.

Floor(x) the largest integer less than or equal to x.

$$\text{GetCurrMsb}(a, b, c, d) = \begin{cases} c + d & ; \quad b - a >= d / 2 \\ c - d & ; \quad a - b > d / 2 \\ c & ; \quad \text{otherwise} \end{cases}$$

Ln(x) the natural logarithm of x (the base-e logarithm, where e is the natural logarithm base constant 2.718 281 828...).

Log2(x) the base-2 logarithm of x.

Log10(x) the base-10 logarithm of x.

$$\text{Min}(x, y) = \begin{cases} x & ; \quad x <= y \\ y & ; \quad x > y \end{cases}$$

$$Max(x, y) = \begin{cases} x & ; \quad x >= y \\ y & ; \quad x < y \end{cases}$$

$$Round(x) = Sign(x) * Floor(Abs(x) + 0.5)$$

$$Sign(x) = \begin{cases} 1 & ; \quad x > 0 \\ 0 & ; \quad x == 0 \\ -1 & ; \quad x < 0 \end{cases}$$

Sin(x) the trigonometric sine function operating on an argument x in units of radians

$$Sqrt(x) = \sqrt{x}$$

Swap(x, y) = (y, x)

Tan(x) the trigonometric tangent function operating on an argument x in units of radians

Order of operation precedence

When an order of precedence in an expression is not indicated explicitly by use of parentheses, the following rules apply:

- Operations of a higher precedence are evaluated before any operation of a lower precedence.
- Operations of the same precedence are evaluated sequentially from left to right.

The table below specifies the precedence of operations from highest to lowest; a higher position in the table indicates a higher precedence.

For those operators that are also used in the C programming language, the order of precedence used in this Specification is the same as used in the C programming language.

Table: Operation precedence from highest (at top of table) to lowest (at bottom of table)

operations (with operands x, y, and z)
"x++", "x– –"
"!x", "–x" (as a unary prefix operator)
x^y
"x * y", "x / y", "x ÷ y", "$\frac{x}{y}$", "x % y"
"x + y", "x – y" (as a two-argument operator), "$\sum_{i=x}^{y} f(i)$"
"x << y", "x >> y"
"x < y", "x <= y", "x > y", "x >= y"
"x = = y", "x != y"
"x & y"
"x | y"
"x && y"
"x || y"
"x ? y : z"
"x..y"
"x = y", "x += y", "x –= y"

Text description of logical operations

In the text, a statement of logical operations as would be described mathematically in the following form:

```
if( condition 0 )
    statement 0
else if( condition 1 )
```

```
    statement 1
...
else /* informative remark on remaining condition */
    statement n
```

may be described in the following manner:

... as follows / ... the following applies:

- If condition 0, statement 0
- Otherwise, if condition 1, statement 1
- ...
- Otherwise (informative remark on remaining condition), statement n

Each "If ... Otherwise, if ... Otherwise, ..." statement in the text is introduced with "... as follows" or "... the following applies" immediately followed by "If ... ". The last condition of the "If ... Otherwise, if ... Otherwise, ..." is always an "Otherwise, ...". Interleaved "If ... Otherwise, if ... Otherwise, ..." statements can be identified by matching "... as follows" or "... the following applies" with the ending "Otherwise, ...".

In the text, a statement of logical operations as would be described mathematically in the following form:

```
if( condition 0a    &&    condition 0b )
    statement 0
else if( condition 1a    ||    condition 1b )
    statement 1
...
else
    statement n
```

may be described in the following manner:

... as follows / ... the following applies:

- If all of the following conditions are true, statement 0:
  - condition 0a
  - condition 0b
- Otherwise, if one or more of the following conditions are true, statement 1:
  - condition 1a
  - condition 1b
- ...
- Otherwise, statement n

In the text, a statement of logical operations as would be described mathematically in the following form:

```
if( condition 0 )
  statement 0
if( condition 1 )
  statement 1
```

may be described in the following manner:

When condition 0, statement 0

When condition 1, statement 1

Embodiments, e.g., of the encoder 20 and the decoder 30, and functions described herein, e.g., with reference to the encoder 20 and the decoder 30, may be implemented in hardware, software, firmware, or any combination thereof. If implemented in software, the functions may be stored on a computer-readable medium or transmitted over communication media as one or more instructions or code and executed by a hardware-based processing unit. Computer-readable media may include computer-readable storage media, which corresponds to a tangible medium such as data storage media, or communication media including any medium that facilitates transfer of a computer program from one place to another, e.g., according to a communication protocol. In this manner, computer-readable media generally may correspond to (1) tangible computer-readable storage media which is non-transitory or (2) a communication medium such as a signal or carrier wave. Data storage media may be any available media that can be accessed by one or more computers or one or more processors to retrieve instructions, code and/or data structures for implementation of the techniques described in this disclosure. A computer program product may include a computer-readable medium.

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

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

The techniques of this disclosure may be implemented in a wide variety of devices or apparatuses, including a wireless handset, an integrated circuit (IC) or a set of ICs (e.g., a chip set). Various components, modules, or units are described in this disclosure to emphasize functional aspects of devices configured to perform the disclosed techniques, but do not necessarily require realization by different hardware units. Rather, as described above, various units may be combined in a codec hardware unit or provided by a collection of interoperative hardware units, including one or more processors as described above, in conjunction with suitable software and/or firmware.

CLAIMS

1. A method of encoding video data into a bitstream (21), the method comprising using a first technique and/or a second technique, characterized in,

the first technique (S101) comprises using a separate merge list for subblock merge candidates; and

the second technique (S103) comprises multi-hypothesis prediction for intra and inter mode;

wherein the method comprises, for a coding block:

transmitting (S107, S109) a first control flag in the bitstream (21), wherein the first control flag indicates whether to use the first technique (S101); and

depending on whether the first technique is used (S101) for the coding block, transmitting (S105, S109) or not transmitting a second control flag in the bitstream (21), wherein the second control flag indicates whether to use the second technique (S103).

2. The method according to claim 1, characterized in, the second control flag is transmitted if and only if the first technique is not used (S101: No) for the coding block.

3. The method according to claim 1 or 2, characterized in, if the coding block is coded in merge mode, the second control flag is transmitted.

4. The method according to claim 3, characterized in, if the coding block is not coded in merge mode, the second control flag is not transmitted.

5. The method according to any of claims 1 to 3, characterized in, if the coding block is coded in skip mode, the second control flag is transmitted.

6. The method according to claim 5, characterized in, if the coding block is not coded in skip mode or in merge mode, the second control flag is not transmitted.

7. A method of decoding video data received in a bitstream (21), characterized in, the method comprising using a first technique (S209) and/or a second technique (S211), wherein

the first technique (S209) comprises using a separate merge list for subblock merge candidates; and

the second technique (S211) comprises multi-hypothesis prediction for intra and inter mode; wherein the method comprises, for a coding block:

receiving (S201) a first control flag from the bitstream (21), wherein the first control flag indicates whether to use (S205) the first technique (S209); and

depending on whether the first technique is used (S209) for the coding block, receiving (S201, S203) a second control flag from the bitstream (21), wherein the second control flag indicates whether to use (S203) the second technique.

8. The method according to claim 7, characterized in, only if the first technique is not used, the second control flag is received.

9. The method according to claim 7 or 8, characterized in, if the coding block is coded in merge mode, the second control flag is received.

10. The method according to claim 9, characterized in, if the coding block is not coded in merge mode, the second control flag is not received.

11. The method according to any of claims 7 to 9, characterized in, if the coding block is coded in skip mode, the second control flag is received.

12. The method according to claim 11, characterized in, if the coding block is not coded in skip mode or in merge mode, the second control flag is not received.

13. An encoder, characterized in, comprising processing circuitry (46) for carrying out the method according to any one of claims 1 to 6.

14. A decoder, characterized in, comprising processing circuitry (46) for carrying out the method according to any one of claims 7 to 12.

15. A computer program comprising program code for performing a method according to any one of claims 1 to 12 when executed on a computer (20, 30) or a processor (43, 430, 502).

16. A encoded bit stream of a video data, characterized in, comprises, multiply syntax elements which include a first control flag, and a second flag being conditionally signaled based on the first control flag, the first control flag indicates that if a first technique is used, wherein, the first technique (S101) comprises using a separate merge list for subblock merge candidates; and the second control flag indicates that if a second technique is used, wherein, the second technique (S103) comprises multi-hypothesis prediction for intra and inter mode.

17. A computing storage medium, characterized in, storing a bit stream decoded by a video decoding device, the bit stream comprise a number of coding blocks of an image or video signal, and a number of syntax elements which include a first control flag, and a second flag being conditionally signaled based on the first control flag, the first control flag indicates that if a first technique is used, wherein, the first technique (S101) comprises using a separate merge list for subblock merge candidates; and the second control flag indicates that if a second technique is used, wherein, the second technique (S103) comprises multi-hypothesis prediction for intra and inter mode.

18. A non-transitory computer-readable storage medium, characterized in, storing video information generated by using any of the encoding method of claim 1 to 6 of the pending claims.