METHOD AND APPARATUS OF CROSS-COMPONENT LINEAR MODELING FOR INTRA PREDICTION

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims the benefit of U.S. Provisional Application No. 62/786,563, filed on December 31, 2018, entitled "METHOD AND APPARATUS OF CROSS-COMPONENT LINEAR MODELING FOR INTRA PREDICTION", which is incorporated herein by its reference.

TECHNICAL FIELD

Embodiments of the present application (disclosure) generally relate to the field of picture processing and more particularly to the intra prediction using cross component linear modelling.

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

Embodiments of the present application provide apparatuses and methods for encoding and decoding according to the independent claims.

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 the invention relates to a method for intra predicting a chroma sample of a block by applying cross-component linear model. The method includes: obtaining reconstructed luma samples; determining maximum and minimum luma sample values based on the reconstructed luma samples; obtaining a difference of the maximum and minimum luma sample values; determining the position of the most-significant bit of the difference of the maximum and minimum luma sample values. The method also includes: fetching a value out of a lookup table (LUT) by using a set of bits as an index, the set of bits following the position of the most-significant bit of the difference of the maximum and minimum luma sample values; obtaining linear model parameters α and β based on the fetched value; and calculating a predicted chroma sample value by using the obtained linear model parameters α and β.

According to the first aspect the invention, index for the LUT is calculated on an elegant way which extracts several bits in binary representation. As a result, the efficiency to fetch the value out of the LUT is increased.

In a possible implementation form of the method according to the first aspect as such, the method obtains the linear model parameters α and β by multiplying the fetched value by the difference of the maximum and minimum values of the reconstructed chroma samples. Since the efficiency to fetch the value out of the LUT is increased, magnitude of multiplier to obtain linear model parameters α and β is minimized.

In a possible implementation form of the method according to the first aspect as such, the LUT includes at least two neighboring values stored in LUT that correspond to different steps of the obtained difference, and the value of this step increases with the value of difference or is a constant. Index for the LUT is calculated on an elegant way which extracts several bits in binary representation, and correspondingly, the size of entry in the LUT corresponding to the index is minimized. As a result, the size of LUT is minimized.

An apparatus for intra predicting a chroma sample of a block by applying cross-component linear model is provided according to the second aspect of the invention. The apparatus according to the second aspect of the invention includes an obtaining unit, a determining unit, and a calculating unit. The obtaining unit, configured to obtain reconstructed luma samples. The determining unit, configured to determine maximum and minimum luma sample values based on the reconstructed luma samples. The obtaining unit, further configured to obtain a difference of the maximum and minimum luma sample values. The determining unit, further configured to determine the position of the most-significant bit of the difference of the maximum and minimum luma sample values. The calculating unit, configured to fetch a value out of a lookup table (LUT) by using a set of bits as an index, the set of bits following the position of the most-significant bit of the difference of the maximum and minimum luma sample values, obtain linear model parameters α and β based on the fetched value; and calculate a predicted chroma sample value by using the obtained linear model parameters α and β.

According to the second aspect of the invention, the apparatus calculate the index for the LUT on an elegant way which extracts several bits in binary representation. As a result, the efficiency to fetch the value out of the LUT is increased.

According to a third aspect of the invention relates to an apparatus for decoding a video stream includes a processor and a memory. The memory is storing instructions that cause the processor to perform the method according to the first aspect or any possible embodiment of the first aspect.

According to a fourth aspect of the invention relates to an apparatus for encoding a video stream includes a processor and a memory. The memory is storing instructions that cause the processor to perform the method according to the first aspect or any possible embodiment of the first aspect.

According to a fifth aspect, a computer-readable storage medium having stored thereon instructions that when executed cause one or more processors configured to code video data is proposed. The instructions cause the one or more processors to perform a method according to the first aspect or any possible embodiment of the first aspect.

According to a sixth aspect, the invention relates to a computer program comprising program code for performing the method according to the first aspect or any possible embodiment of the first aspect when executed on a computer.

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.

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 drawing illustrating a concept of Cross-component Linear Model for chroma intra prediction;

FIG. 7 is a drawing illustrating simplified method of linear model parameter derivation;

FIG. 8 is a drawing illustrating occurrence probability distribution of the difference between maximum and minimum values of the reference luma samples;

FIG. 9 is a flowchart illustrating exemplary lookup table generation process in accordance with an embodiment of the invention;

FIG. 10 is a drawing with flowcharts illustrating an embodiment of index derivation for the exemplary lookup table;

FIG. 11 is a flowchart illustrating exemplary intra prediction of a chroma sample of a block by applying cross-component linear model;

FIG. 12 is a block diagram showing an example structure of an apparatus for intra prediction of a chroma sample of a block by applying cross-component linear model;

FIG. 13 is a block diagram showing an example structure of a content supply system 3100 which realizes a content delivery service; and

FIG. 14 is a block diagram showing a structure of an example of a terminal device.

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 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 via 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 via processing circuitry 46 to embody the various modules as discussed with respect to encoder 20of FIG. 2 and/or any other encoder system or subsystem described herein. The decoder 30 may be implemented via 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 colour 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, 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 filters or

a collaborative filters, 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 exemple 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 213 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 limit to control point motion vectors of affine mode, sub-block motion vectors in affine, planar, ATMVP 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. For example, the value of the derived motion vector (e.g. the MVs of four 4x4 sub-blocks within one 8x8 block) is constrained such that the max difference between integer parts of the four 4x4 sub-block MVs is no more than N pixels, such as no more than 1 pixel. Here provides two methods for constraining the motion vector according to the bitDepth.

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

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

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

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

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

where mvx is a horizontal component of a motion vector of an image block or a sub-block, mvy is a vertical component of a motion vector of an image block or a sub-block, and ux and uy indicates an intermediate value;

For example, if the value of mvx is -32769, after applying formula (1) and (2), the resulting value is 32767. In computer system, 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 and mvd, 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 vx is a horizontal component of a motion vector of an image block or a sub-block, vy is a vertical component of a motion vector of an image block or a sub-block; x, y and z respectively correspond to three input value of the MV clipping process, and 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.

Intra-prediction of chroma samples could be performed using samples of reconstructed luma block.

During HEVC development Cross-component Linear Model (CCLM) chroma intra prediction was proposed [J. Kim, S.-W. Park, J.-Y. Park, and B.-M. Jeon, Intra Chroma Prediction Using Inter Channel Correlation, document JCTVC-B021, Jul. 2010]. CCLM uses linear correlation between a chroma sample and a luma sample at the corresponding position in a coding block. When a chroma block is coded using CCLM, a linear model is derived from the reconstructed neighboring luma and chroma samples by linear regression. The chroma samples in the current block can then be predicted by the reconstructed luma samples in the current block with the derived linear model (see Fig. 6):

$$C(x,y) = \alpha \times L(x,y) + \beta ,$$

where C and L indicate chroma and luma values, respectively. Parameters α and β are derived by the least-squares method as follows:

$$\alpha = \frac{R(L,C)}{R(L,L)}$$
$$\beta = M(C) - \alpha \times M(L) ,$$

where $M(A)$ represents mean of A and $R(A,B)$ is defined as follows:

$$R(A,B) = M\big((A - M(A)) \times (B - M(B))\big).$$

If encoded or decoded picture has a format that specifies different number of samples for luma and chroma components (e.g. 4:2:0 YCbCr format), luma samples are down-sampled before modelling and prediction.

The method has been adopted for usage in VTM2.0. Specifically, parameter derivation is performed as follows:

$$\alpha = \frac{N \cdot \sum(L(n) \cdot C(n)) - \sum L(n) \cdot \sum C(n)}{N \cdot \sum(L(n) \cdot L(n)) - \sum L(n) \cdot \sum L(n)},$$
$$\beta = \frac{\sum C(n) - \alpha \cdot \sum L(n)}{N},$$

where *L(n)* represents the down-sampled top and left neighbouring reconstructed luma samples, *C(n)* represents the top and left neighbouring reconstructed chroma samples.

In [G. Laroche, J. Taquet, C. Gisquet, P. Onno (Canon), "CE3: Cross-component linear model simplification (Test 5.1)", Input document to 12th JVET Meeting in Macao, China, Oct. 2018] a different method of deriving α and β was proposed (see Fig. 7). In particular, the linear model parameters α and β are obtained according to the following equations:

$$\alpha = \frac{C(B) - C(A)}{L(B) - L(A)}$$

$$\beta = L(A) - \alpha C(A), \text{ where}$$

where $B = \text{argmax}(L(n))$ and $A = \text{argmin}(L(n))$ are positions of maximum and minimum values in luma samples.

It was also proposed to implement division operation using multiplication by a number that is stored in a look-up table (LUT), which is specified in Table 1. This substitution is possible by using the following method:

$$\frac{v_0}{v_1} \approx v_0 \cdot \text{LUT}[v_1] \gg S,$$

where S is a shift parameter that specifies the precision.

Table 1 provides matching of the LUT indices range (given in the first line of the table) with the lists of values stored in LUT, each list corresponding to its indices range. It could be noticed, that $\text{LUT}[v_1]$ values could be calculated as follows:

$$\text{LUT}[v_1] = \frac{1 \ll S}{v_1}.$$

Using the LUT defined in Table 1 (or equivalently calculated using the equation above), calculation of α is performed as follows:

$$\alpha = \frac{C(B) - C(A)}{L(B) - L(A)} \approx (C(B) - C(A)) \cdot \text{LUT}[L(B) - L(A)] \gg S_0$$

Shift parameter S may be decomposed onto several parts, i.e. $S = S_0 + S_1$, because the value of α is used in a further calculation. This decomposition provides flexibility of calculation precision at different stages and thus it is possible to redistribute bit depth of the multiplication operations over the steps of obtaining a value of a chroma predicted sample. In the particular implementation:

$$S_0 = 1,$$

$$S_1 = 15,$$

$$S = S_0 + S_1 = 16.$$

Table 1. Exemplary LUT table to implement division operation using multiplication by a number

Indices ranges of the LUT:															
0	**32**	**64**	**96**	**128**	**160**	**192**	**224**	**256**	**288**	**320**	**352**	**384**	**416**	**448**	**480**
..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..
31	**63**	**95**	**127**	**159**	**191**	**223**	**255**	**287**	**319**	**351**	**383**	**415**	**447**	**479**	**511**
Values stored in the LUT:															
65536	1985	1008	675	508	407	339	291	255	226	204	185	170	157	145	136
32768	1927	992	668	504	404	337	289	254	225	203	185	169	156	145	135
21845	1872	978	661	500	402	336	288	253	225	202	184	169	156	145	135
16384	1820	963	655	496	399	334	287	252	224	202	184	168	156	144	135
13107	1771	949	648	492	397	332	286	251	223	201	183	168	155	144	135
10922	1724	936	642	489	394	330	284	250	222	201	183	168	155	144	134
9362	1680	923	636	485	392	329	283	249	222	200	182	167	154	144	134
8192	1638	910	630	481	390	327	282	248	221	199	182	167	154	143	134
7281	1598	897	624	478	387	326	281	247	220	199	181	166	154	143	134
6553	1560	885	618	474	385	324	280	246	219	198	181	166	153	143	133
5957	1524	873	612	471	383	322	278	245	219	197	180	165	153	142	133
5461	1489	862	606	468	381	321	277	244	218	197	180	165	153	142	133
5041	1456	851	601	464	378	319	276	243	217	196	179	165	152	142	132
4681	1424	840	595	461	376	318	275	242	217	196	179	164	152	141	132
4369	1394	829	590	458	374	316	274	241	216	195	178	164	152	141	132
4096	1365	819	585	455	372	315	273	240	215	195	178	163	151	141	132
3855	1337	809	579	451	370	313	271	240	214	194	177	163	151	140	131
3640	1310	799	574	448	368	312	270	239	214	193	177	163	151	140	131
3449	1285	789	569	445	366	310	269	238	213	193	176	162	150	140	131
3276	1260	780	564	442	364	309	268	237	212	192	176	162	150	140	131
3120	1236	771	560	439	362	307	267	236	212	192	175	161	149	139	130
2978	1213	762	555	436	360	306	266	235	211	191	175	161	149	139	130
2849	1191	753	550	434	358	304	265	234	210	191	174	161	149	139	130
2730	1170	744	546	431	356	303	264	234	210	190	174	160	148	138	130
2621	1149	736	541	428	354	302	263	233	209	189	173	160	148	138	129
2520	1129	728	537	425	352	300	262	232	208	189	173	159	148	138	129
2427	1110	720	532	422	350	299	261	231	208	188	172	159	147	137	129
2340	1092	712	528	420	348	297	260	230	207	188	172	159	147	137	129
2259	1074	704	524	417	346	296	259	229	206	187	172	158	147	137	128
2184	1057	697	520	414	344	295	258	229	206	187	171	158	146	137	128
2114	1040	689	516	412	343	293	257	228	205	186	171	157	146	136	128
2048	1024	682	512	409	341	292	256	227	204	186	170	157	146	136	128

In this case linear model coefficient α has a fixed-point integer representation of the fractional value, precision of α is determined by the value of $S_1 = S - S_0$, which is used in obtaining a value of a chroma predicted sample:

$$\boldsymbol{pred_C(i,j)} = \left(\alpha \cdot rec_L^{\prime(i,j)}\right) \gg S_1 + \beta \ ,$$

The size of the LUT is rather critical in hardware implementation of an encoder or a decoder. The most straightforward way to solve this problem is to subsample a LUT regularly by keeping just each N^{th} element (here N is a subsampling ratio) of the initial LUT specified, i.e. in Table 1.

After a regular subsampling with a power-of-two subsampling ratio of N, a fetch from the LUT is defined differently:

$$m = \text{LUT}[(L(B) - L(A) \gg \log_2(N))]$$

instead of

$$m = \text{LUT}[L(B) - L(A)]$$

It could be noticed, that for natural pictures the probability of $L(B) - L(A)$ will have a small value is larger than the probability that this difference will be large. In other words, the probability of occurrence of the value from Table 1 decreases from left columns to right columns, and inside each column this probability decreases from the first to the last element belonging to that column. Exemplary occurrence probability dependency on the value of $L(B) - L(A)$ is given in Fig. 8.

Therefore a keeping just each N^{th} element of the initial LUT is far from being optimal, since it corresponds with equal probability distribution of its argument which is not true.

By considering this distribution it is possible to achieve a better tradeoff between the size of the LUT and precision of calculations than regular subsampling could provide.

It is proposed to define a LUT using non-linear indexing, specifically in such a way that two neighboring LUT entries will correspond to different steps of $L(B) - L(A)$, and the value of this step increases with the index of the entry.

One of the computationally efficient solutions is to determine the index within a LUT using several most significant bits of $L(B) - L(A)$. Position of the most significant bit of $L(B) -$

$L(A)$ defines the precision (i.e. the step between the two neighboring entries of the LUT) based on the position of the most significant bits. The greater values of the position of the most significant bit correspond to lower precision and the greater step value.

A particular embodiment is shown in Fig. 9 and Fig. 10. Fig. 9 demonstrates a flowchart of LUT values calculation, and Fig. 10 shows how to obtain an index "idx" within a LUT corresponding to an input value of $L(B) - L(A)$.

By using the steps shown in Fig. 9, it is possible to obtain values that will be stored in a LUT for their further usage in CCLM modelling. In Fig. 9, "ctr" variable is iterates through all the possible values of $L(B) - L(A) - 1$. It could be noticed that in fact the LUT has subranges, each subrange comprises "col_{max}" entries. Within a subrange "ctr" value increases with the equal "step" value. A subrange that has index of "lut_shift+1" that is greater than 1 has a corresponding "step" value increased twice as compared with "lut_shift" subrange. The first two subranges would have a step equal to one due to a threshold in "step=1<<max(0,lut_shift)" operation.

As shown in Fig. 9, the flowchart illustrates exemplary lookup table generation process in accordance with an embodiment of the invention. At step 902, the video coding device starts the lookup table generation process. The video coding device may be a decoder such as video decoder 30 of Fig. 1A-1B, Fig. 3, or an encoder such as video encoder 20 of FIG. 1A-1B, Fig. 2, or Video Coding Device 400 of Fig. 4, or the apparatus 500 of Fig. 5.

At step 904, let ctr=1 and lut_shift=1. At step 906, whether index lut_shift< lut_shiftmax is determined. If lut_shift< lut_shiftmax, the step is calculated as 1<<max(0, lut_shift), and col=0 at step 908; otherwise, the generation process ends at step 922. At step 910, starting offset is provided by the "ctr=ctr+(step>>1)" step. Then whether col<colmax is determined at step 912. If col<colmax, LUT[ctr]=(1<<S)/ctr at step 914, here, the pre-calculated LUT values which are defined at steps 912-918 generate one row of the LUT; otherwise, let index lut_shift= lut_shift+1 at step 920. At step 916, the values of ctr corresponding to the start of each subrange is set as ctrl+step. At step 918, the generation process moves to next column, then the process goes back to step 912.

Exemplary LUT generated using a flowchart shown in Fig. 9 is given in Table 2. A row of this table corresponds to a subrange having "lut_shift" index. This table was obtained with

"lut_shift_{max}" equal to 6 and "col_{max}" equal to 3 thus resulting in 48 entries within a LUT. The values of ctr corresponding to the start of each subrange (zero "col" value) are as follows:

0, 8, 17, 35, 71, 143, 287.

These values are not always a power of two because subsampling within each subrange is performed with respect to the middle of subrange. Corresponding starting offset is provided by the "ctr=ctr+(step>>1)" step shown in Fig. 9. The value of "step" for "lut_shift" values not greater than 0 is set equal to 1.

Table 2. Exemplary LUT generated using a flowchart shown in Fig. 9

lut_shift	col							
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
-1	65536	32768	21845	16384	13107	10922	9362	8192
0	7281	6553	5957	5461	5041	4681	4369	4096
1	3640	3276	2978	2730	2520	2340	2184	2048
2	1820	1638	1489	1365	1260	1170	1092	1024
3	910	819	744	682	630	585	546	512
4	455	409	372	341	315	292	273	256
5	227	204	186	170	157	146	136	128

In Fig. 10A, a binary representation 1001 of input value (e.g., a difference $L(B) - L(A)$) is being processed in order to determine position of corresponding entry in the LUT shown in Table 2. The most significant non-zero bit of the input value is marked as 1002. Position of this bit determines the "msb" value. In fact, msb value is a log2() of the input value 1001. Subtracting 1 from "msb" gives the "lut_shift" value that selects a row in Table 2. Otherwise, step is calculated as "1<<lut_shift".

Column selection is performed by taking "col_{max}" bits next to 1002. The value of "col" within Table 2 is obtained as follows:

- value of "high_bits" is obtained by selecting "col_{max}+1" bits that follows the most significant bit 1020;
- col is set equal to the value stored in "high_bits" decremented by 1.

For the case when aligning step "ctr=ctr+(step>>1)" is not performed in Fig.9, derivation of "lut_shift" value is the same and derivation of "col" value is even simpler (Fig. 10B): value of "col" is obtained by selecting "col_{max}" bits that follows the most significant bit 1020. This index derivation method corresponds to Table 3. The values of "ctr" (Fig. 9) corresponding to the start of each subrange (zero "col" value) are as follows:

0, 8, 16, 32, 64, 128, 256

Table 3. Another Exemplary LUT generated using a flowchart shown in Fig. 9 in case when "ctr=ctr+(step>>1)" is skipped

lut_shift	col							
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
-1	65536	32768	21845	16384	13107	10922	9362	8192
0	7281	6553	5957	5461	5041	4681	4369	4096
1	3855	3449	3120	2849	2621	2427	2259	2114
2	1985	1771	1598	1456	1337	1236	1149	1074
3	1008	897	809	736	675	624	579	541
4	508	451	407	370	339	313	291	271
5	255	226	204	185	170	157	145	136

When deriving the value of "col", msb may be less than or equal to "col_{max}". In this case, the value of "col" is set equal to the "col_{max}" least significant bits of the input difference 1001.

In practical implementations the LUT index is a one dimensional one. It's typically understood that the LUTs shown in Table 2 and Table 3 could be addressed linearly using index set equal to:

$$\max(\text{lut}_{\text{shift}}, 0) \cdot (1 \ll \text{col}_{\text{max}}) + \text{col}.$$

Both LUTs shown in Table 2 and Table 3 store values that are very different in their magnitude. Hence it is reasonable to have a similar number of bits for all the values stored in the LUT. The value being fetched from the LUT could further be left-shifted in accordance with the value of lut_shift. The only exception from this rule are the first four values that have different precision with the last for values within this row. However, this problem could be solved by and additional LUT that stores this additional shifts for the first four values. In this embodiment, the value of multiplier is restored from the value fetched from the LUT as follows:

$$m = \text{LUT}[\text{idx}] \ll m_s$$

where $m_s = 6 - (\text{lut}_{\text{shift}} + 1) + \delta$. The value of δ is set equal to 3, 2, 1, 1, respectively for "idx" values less or equal to 4. Look up table for this embodiment is given in Table 4.

Table 4. Another Exemplary LUT generated using a flowchart shown in Fig. 9 in case of equal precision within ranges.

lut_shift	col							
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
-1	128	128	171	128	205	171	146	128

0	228	205	186	171	158	146	137	128
1	241	216	195	178	164	152	141	132
2	248	221	200	182	167	155	144	134
3	252	224	202	184	169	156	145	135
4	254	226	204	185	170	157	146	136
5	128	128	171	128	205	171	146	128

It could be noticed that last rows of Table 4 are very similar to each other. Hence it is possible to reduce the size of the LUT by storing just one row for some sets of subbranges. In a particular embodiments, when a value of "lut_shift" is greater than a certain threshold, the value of lut_shift is set equal to the threshold and the value of δ is decreased by the difference between initial value of "lut_shift" and the value of threshold.

FIG. 11 is a flowchart illustrating exemplary intra prediction of a chroma sample of a block by applying cross-component linear model. At step 1102, a video coding device obtains reconstructed luma samples. The video coding device may be a decoder such as video decoder 30 of Fig. 1A-1B, Fig. 3, or an encoder such as video encoder 20 of FIG. 1A-1B, Fig. 2, or Video Coding Device 400 of Fig. 4, or the apparatus 500 of Fig. 5.

At step 1104, the video coding device determines positions of maximum and minimum reconstructed sample values within the reconstructed luma samples. For example, the reconstructed luma samples are neighboring reconstructed luma samples corresponding to the chroma sample.

At step 1106, the video coding device obtains the value of difference of the maximum and minimum of the reconstructed luma samples.

At step 1108, the video coding device calculates an index for a LUT to fetch a value of a multiplier that corresponds to the value of difference the maximum and minimum of the reconstructed luma samples.

For example, the video coding device determines the position of the most-significant bit of the difference of the maximum and minimum luma sample values, and uses a set of bits following the position of the most-significant bit of the difference of the maximum and minimum luma sample values as the index for the LUT to fetch the value. The position of the most-significant bit of the difference of the maximum and minimum luma sample values may be obtained as a logarithm of two of the difference. The video coding device determines the set of bits following the position of the most-significant bit of the difference. As a possible result, the set of bits includes four bits.

The LUT is generated with or without an aligning step. The LUT may include at least two neighboring values stored in LUT that correspond to different steps of the obtained difference, and the value of this step increases with the value of difference or is a constant.

As disclosed in exemplary tables 1-4, the LUT may include subranges of values. A step of the value of difference of the maximum and minimum of the reconstructed luma samples is constant within a subrange; and a step for different subranges is different. As an example, the step of the value of difference of the maximum and minimum of the reconstructed luma samples increases with the increase of the subrange index. For example, the step of the value of difference of the maximum and minimum of the reconstructed luma samples may be a power of two of the subrange index.

Let the LUT includes at least three values: a first value, a second value, and a third value. Among the three values, the first value and the second value are two neighboring values, and the second value and the third value are two neighboring values. A step (i.e., precision, or difference) between the first value and the second value may equal to, or may be different from, a step between the second value and the third value. If the first value is indexed by a first set of bits, and the second value is indexed by a second set of bits, the first value is smaller than the second value when a value of the first set of bits is greater than a value of the second set of bits; or the first value is greater than the second value when a value of the first set of bits is smaller than a value of the second set of bits.

The LUT is divided into subranges. A subrange index is determined using the position of the most significant non-zero bit of the difference of the maximum and minimum of the reconstructed luma samples. As an example, subrange size is set to 8 and the number of subranges is 6. As another example, different neighboring subranges have different value increases with the same step.

The LUT may include non-linear indexes. Two neighboring LUT entries correspond to different steps of $L(B)-L(A)$, where $L(B)$ represents the maximum value of the reconstructed luma samples, $L(A)$ represents the minimum value of the reconstructed luma samples. A value of a step of the entry may increase with the index of the entry.

When the LUT uses several most significant bits of $L(B)-L(A)$, position of the most significant bit of $L(B)-L(A)$ defines the precision (i.e. the step between the two neighboring entries of the LUT) based on the position of the most significant bits. The greater values of the position of the most significant bit may correspond to lower precision and the greater step value.

At step 1110, the video coding device obtains linear model parameters α and β, by multiplying the fetched value by the difference of the maximum and minimum values of the reconstructed chroma samples.

At step 1112, the video coding device calculates predicted chroma sample value using the obtained linear model parameters α and β.

FIG. 12 is a block diagram showing an example structure of an apparatus 1200 for intra prediction of a chroma sample of a block by applying cross-component linear model. The apparatus 1200 is configured to carry out the above methods, and may include:

an obtaining unit 1210, configured to obtain reconstructed luma samples;

a determining unit 1220, configured to determine maximum and minimum luma sample values based on the reconstructed luma samples;

the obtaining unit 1210, further configured to obtain a difference of the maximum and minimum luma sample values.

the determining unit 1220, further configured to determine the position of the most-significant bit of the difference of the maximum and minimum luma sample values. As an example, the position of the most-significant bit of the difference of the maximum and minimum luma sample values is a logarithm of two of the difference. As an implementation, the most-significant bit is a first non-zero bit.

The apparatus 1200 further includes a calculating unit 1230, configured to fetch a value out of a lookup table (LUT) by using a set of bits as an index, the set of bits following the position of the most-significant bit of the difference of the maximum and minimum luma sample values, obtain linear model parameters α and β based on the fetched value; and calculate a predicted chroma sample value by using the obtained linear model parameters α and β. For example, the set of bits comprises four bits

The calculating unit may obtain the linear model parameters α and β, based on the fetched value and a difference of maximum and minimum values of reconstructed chroma samples. For example, the calculating unit obtains the linear model parameters α and β by multiplying the fetched value by the difference of the maximum and minimum values of the reconstructed chroma samples.

Benefit of the embodiments of the invention

1. index for the LUT is calculated on an elegant way which extracts several bits in binary representation. As a result, the efficiency to fetch the value out of the LUT is increased.

2. Since the efficiency to fetch the value out of the LUT is increased, magnitude of multiplier to obtain linear model parameters α and β is minimized.

3. The size of LUT is minimized. Curve of division function (f(x) = 1/x, known as hyperbola) in embodiments of this invention was approximated on a way:

1) LUT table size may be 16
   i. (minimal number of entries for approximation of 1/x curve which has derivative changing from 0 to infinity)
2) Elements of LUT have non-linear dependency on entry index
   i. (to approximate 1/x curve)
3) Multipliers (elements of LUT) are 3 bits unsigned integer numbers (0…7)
   i. (minimal precision for approximation of 1/x curve which has derivative changing from 0 to infinity)

Following is an explanation of the applications of the encoding method as well as the decoding method as shown in the above-mentioned embodiments, and a system using them.

FIG. 13 is a block diagram showing a content supply system 3100 for realizing content distribution service. This content supply system 3100 includes capture device 3102, terminal device 3106, and optionally includes display 3126. The capture device 3102 communicates with the terminal device 3106 over communication link 3104. The communication link may include the communication channel 13 described above. The communication link 3104 includes but not limited to WIFI, Ethernet, Cable, wireless (3G/4G/5G), USB, or any kind of combination thereof, or the like.

The capture device 3102 generates data, and may encode the data by the encoding method as shown in the above embodiments. Alternatively, the capture device 3102 may distribute the data to a streaming server (not shown in the Figures), and the server encodes the data and transmits the encoded data to the terminal device 3106. The capture device 3102 includes but not limited to camera, smart phone or Pad, computer or laptop, video conference system, PDA, vehicle mounted device, or a combination of any of them, or the like. For example, the capture device 3102 may include the source device 12 as described above. When the data includes video, the video encoder 20 included in the capture device 3102 may actually perform video encoding processing. When the data includes audio (i.e., voice), an audio encoder included in the capture device 3102 may actually perform audio encoding processing. For some practical scenarios, the capture device 3102 distributes the encoded video and audio data by multiplexing them together. For other practical scenarios, for example in the video conference system, the encoded audio data and the encoded video data are not multiplexed. Capture device 3102

distributes the encoded audio data and the encoded video data to the terminal device 3106 separately.

In the content supply system 3100, the terminal device 310 receives and reproduces the encoded data. The terminal device 3106 could be a device with data receiving and recovering capability, such as smart phone or Pad 3108, computer or laptop 3110, network video recorder (NVR)/ digital video recorder (DVR) 3112, TV 3114, set top box (STB) 3116, video conference system 3118, video surveillance system 3120, personal digital assistant (PDA) 3122, vehicle mounted device 3124, or a combination of any of them, or the like capable of decoding the above-mentioned encoded data. For example, the terminal device 3106 may include the destination device 14 as described above. When the encoded data includes video, the video decoder 30 included in the terminal device is prioritized to perform video decoding. When the encoded data includes audio, an audio decoder included in the terminal device is prioritized to perform audio decoding processing.

For a terminal device with its display, for example, smart phone or Pad 3108, computer or laptop 3110, network video recorder (NVR)/ digital video recorder (DVR) 3112, TV 3114, personal digital assistant (PDA) 3122, or vehicle mounted device 3124, the terminal device can feed the decoded data to its display. For a terminal device equipped with no display, for example, STB 3116, video conference system 3118, or video surveillance system 3120, an external display 3126 is contacted therein to receive and show the decoded data.

When each device in this system performs encoding or decoding, the picture encoding device or the picture decoding device, as shown in the above-mentioned embodiments, can be used.

FIG. 14 is a diagram showing a structure of an example of the terminal device 3106. After the terminal device 3106 receives stream from the capture device 3102, the protocol proceeding unit 3202 analyzes the transmission protocol of the stream. The protocol includes but not limited to Real Time Streaming Protocol (RTSP), Hyper Text Transfer Protocol (HTTP), HTTP Live streaming protocol (HLS), MPEG-DASH, Real-time Transport protocol (RTP), Real Time Messaging Protocol (RTMP), or any kind of combination thereof, or the like.

After the protocol proceeding unit 3202 processes the stream, stream file is generated. The file is outputted to a demultiplexing unit 3204. The demultiplexing unit 3204 can separate the multiplexed data into the encoded audio data and the encoded video data. As described above, for some practical scenarios, for example in the video conference system, the encoded audio data and the encoded video data are not multiplexed. In this situation, the encoded data is transmitted to video decoder 3206 and audio decoder 3208 without through the demultiplexing unit 3204.

Via the demultiplexing processing, video elementary stream (ES), audio ES, and optionally subtitle are generated. The video decoder 3206, which includes the video decoder 30 as explained in the above mentioned embodiments, decodes the video ES by the decoding method as shown in the above-mentioned embodiments to generate video frame, and feeds this data to the synchronous unit 3212. The audio decoder 3208, decodes the audio ES to generate audio frame, and feeds this data to the synchronous unit 3212. Alternatively, the video frame may store in a buffer (not shown in FIG. Y) before feeding it to the synchronous unit 3212. Similarly, the audio frame may store in a buffer (not shown in FIG. Y) before feeding it to the synchronous unit 3212.

The synchronous unit 3212 synchronizes the video frame and the audio frame, and supplies the video/audio to a video/audio display 3214. For example, the synchronous unit 3212 synchronizes the presentation of the video and audio information. Information may code in the syntax using time stamps concerning the presentation of coded audio and visual data and time stamps concerning the delivery of the data stream itself.

If subtitle is included in the stream, the subtitle decoder 3210 decodes the subtitle, and synchronizes it with the video frame and the audio frame, and supplies the video/audio/subtitle to a video/audio/subtitle display 3216.

The present invention is not limited to the above-mentioned system, and either the picture encoding device or the picture decoding device in the above-mentioned embodiments can be incorporated into other system, for example, a car system.

Although embodiments of the invention have been primarily described based on video coding, it should be noted that embodiments of the coding system 10, encoder 20 and decoder 30 (and correspondingly the system 10) and the other embodiments described herein may also be configured for still picture processing or coding, i.e. the processing or coding of an individual picture independent of any preceding or consecutive picture as in video coding. In general only inter-prediction units 244 (encoder) and 344 (decoder) may not be available in case the picture processing coding is limited to a single picture 17. All other functionalities (also referred to as tools or technologies) of the video encoder 20 and video decoder 30 may equally be used for still picture processing, e.g. residual calculation 204/304, transform 206, quantization 208, inverse quantization 210/310, (inverse) transform 212/312, partitioning 262/362, intra-prediction 254/354, and/or loop filtering 220, 320, and entropy coding 270 and entropy decoding 304.

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 limitating, 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 intra predicting a chroma sample of a block by applying a cross-component linear model, comprising:

obtaining reconstructed luma samples;

determining maximum and minimum luma sample values based on the reconstructed luma samples;

obtaining a difference of the maximum and minimum luma sample values;

determining the position of the most-significant bit of the difference of the maximum and minimum luma sample values;

fetching a value out of a lookup table (LUT) by using a set of bits as an index, the set of bits following the position of the most-significant bit of the difference of the maximum and minimum luma sample values;

obtaining linear model parameters α and β based on the fetched value; and

calculating a predicted chroma sample value by using the obtained linear model parameters α and β.

2. The method of claim 1, wherein the position of the most-significant bit of the difference of the maximum and minimum luma sample values is obtained as a logarithm of two of the difference.

3. The method of claim 1 or 2, wherein determining the set of bits following the position of the most-significant bit of the difference results in the set of bits comprising four bits.

4. The method of any one of claims 1-3, wherein the most-significant bit is the first non-zero bit.

5. The method of any one of claims 1-4, wherein the method comprises:
obtaining the linear model parameters α and β, based on the fetched value and a difference of maximum and minimum values of reconstructed chroma samples.

6. The method of claim 5, wherein the method comprises:
obtaining the linear model parameters α and β by multiplying the fetched value by the difference of the maximum and minimum values of the reconstructed chroma samples.

7. The method of any one of claims 1-6, wherein the LUT comprises at least three values: a first value, a second value, and a third value;
among the three values, the first value and the second value are two neighboring values, and the second value and the third value are two neighboring values.

8. The method of claim 7, wherein a step between the first value and the second value equals to a step between the second value and the third value.

9. The method of claim 7, wherein a step between the first value and the second value is different from a step between the second value and the third value.

10. The method of claims 7-9, wherein the first value is indexed by a first set of bits, and the second value is indexed by a second set of bits;
when a value of the first set of bits is greater than a value of the second set of bits, the first value is smaller than the second value; or
when a value of the first set of bits is smaller than a value of the second set of bits, the first value is greater than the second value.

11. The method of claim 1-10, wherein the LUT comprises subranges of values, and a step of any two neighboring values is constant within one subrange.

12. An apparatus for intra predicting a chroma sample of a block by applying a cross-component linear model, wherein the apparatus is an encoder or a decoder, and the apparatus comprises:

an obtaining unit, configured to obtain reconstructed luma samples;

a determining unit, configured to determine maximum and minimum luma sample values based on the reconstructed luma samples;

the obtaining unit, further configured to obtain a difference of the maximum and minimum luma sample values;

the determining unit, further configured to determine the position of the most-significant bit of the difference of the maximum and minimum luma sample values; and

a calculating unit, configured to fetch a value out of a lookup table (LUT) by using a set of bits as an index, the set of bits following the position of the most-significant bit of the

difference of the maximum and minimum luma sample values, obtain linear model parameters α and β based on the fetched value; and calculate a predicted chroma sample value by using the obtained linear model parameters α and β.

13. The apparatus of claim 12, wherein the position of the most-significant bit of the difference of the maximum and minimum luma sample values is a logarithm of two of the difference.

14. The apparatus of claim 12 or 13, wherein the set of bits comprises four bits

15. The apparatus of any one of claims 12-14, wherein the most-significant bit is the first non-zero bit.

16. The apparatus of any one of claims 12-15, wherein the calculating unit is configured to: obtain the linear model parameters α and β, based on the fetched value and a difference of maximum and minimum values of reconstructed chroma samples.

17. The apparatus of claim 16, wherein the calculating unit is configured to:
obtain the linear model parameters α and β by multiplying the fetched value by the difference of the maximum and minimum values of the reconstructed chroma samples.

18. An encoder comprising processing circuitry for carrying out the method according to any one of claims 1 to 11.

19. A decoder comprising processing circuitry for carrying out the method according to any one of claims 1 to 11.

20. A computer program product comprising a program code for performing the method according to any one of claims 1 to 11.

21. A decoder, comprising:

one or more processors; and

a non-transitory computer-readable storage medium coupled to the processors and storing programming for execution by the processors, wherein the programming, when executed by

the processors, configures the decoder to carry out the method according to any one of claims 1 to 11.

22. An encoder, comprising:

one or more processors; and

a non-transitory computer-readable storage medium coupled to the processors and storing programming for execution by the processors, wherein the programming, when executed by the processors, configures the encoder to carry out the method according to any one of claims 1 to 11.

23. A non-transitory recording medium which includes a predictive-encoded bit stream decoded by an apparatus, the bit stream being generated according to any one of claims 1 to 11.