TITLE

AN ENCODER, A DECODER AND CORRESPONDING METHODS FOR INTRA PREDICTION

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims priority to U.S. Provisional Application No. 62/822,865, filed on March 23, 2019, U.S. Provisional Application No. 62/824,282, filed on March 26, 2019, and U.S. Provisional Application No. 62/824,360, filed on March 27, 2019, all of which applications are incorporated herein by references.

TECHNICAL FIELD

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

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.

In particular, there is an ongoing demand for reducing the processor load in the context of intra prediction processing. In the art, it is known to employ Most Probable Mode (MPM) list for intra prediction mode coding. The MPM list reduces bits required for coding the current block's intra prediction mode. When the intra prediction mode of the current block corresponds to an entry of an MPM list, an index rather than the actual mode is coded whereby less bits are necessary. However, selecting the entries for the generation of an MPM list in view of the most efficient intra prediction coding is still a task. The proper selection of the entries heavily influences the efficiency of the intra prediction mode coding based on MPM list.

In view of this, it is an object underlying the present invention to provide for a technique for obtaining intra prediction mode based on MPM list that results in more efficient coding as compared to the art.

SUMMARY

Embodiments of the present application provide devices 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 performed by a decoding device. The method includes: obtaining a value of a first indication information of a current block, the value of the first indication information indicating whether an intra prediction mode of the current block is comprised in a set of most probable modes; and obtaining a value of a reference index line of the current block. The set of most probable modes comprises 5 candidate intra prediction modes and Planar mode. When the value of the first indication information indicates the intra prediction mode of the current block is comprised in the set of most probable modes, and when the value of the reference index line indicates the closest neighboring reference line to the current block is used, the method further includes obtaining a value of a second indication information of the current block, the value of the second indication information of the current block indicating whether the intra prediction mode of

the current block is Planar mode or not.

In the conventional method, the decoding complexity for the MPM processing is high. Depending on the usage of Multiple Reference Line (MRL) and intra sub-partition coding mode (ISP) tools, three different MPM lists (6 entries) are constructed. In the aspects or any possible embodiments of the aspects of this invention, the MPM list (5 entries) construction is unified with reduced decoding complexity, regardless of whether using MRL or ISP coding tool.

Furthermore, planar mode is considered as a special mode due to its high frequency and a special flag is assigned to check whether the current mode is planar mode or not. Although it is not in the MPM list (with invented 5 entries), It is always checked when mpm flag is true. In other word, the mpm flag indicates whether the intra prediction mode of the current block is in a list of 6 entries, comprising of planar mode and the 5 MPM list entries. The 5 MPM list entries are constructed based on the intra prediction modes of left and above block).

As an implementation of the first aspect, the first indication information is indicated by a first flag, for example, intra_luma_mpm_flag. The value of the reference index line of the current block may be indicated by intra_luma_ref_idx. The second indication information may be indicated by a second flag, for example, intra_luma_not_planar_flag.

When MRL coding is not enabled, i.e., when the value of the reference index line indicates the closest neighboring reference line to the current block is used, the devices parses a bitstream to obtain the value of the second indication information. When MRL coding is enabled, the value of the second indication information is obtained by a default value. In this situation, Planar mode is excluded and its corresponding flag (i.e. intra_luma_not_planar_flag) is not parsed from the bitstream, but is inferred as a default value (e.g. a value of 1). In this case, a bit is saved to indicate the value of the planar flag (i.e, intra_luma_not_planar) when MRL is enabled).

According to a second aspect the invention relates to a method performed by an encoding apparatus. The method includes: determining whether an intra prediction mode of a current block is comprised in a set of most probable modes; and determining whether the closest neighboring reference line to the current block is used for intra prediction. The set of most probable modes comprises 5 candidate intra prediction modes and Planar mode. The method further includes encoding a bitstream, wherein the bitstream includes information to indicate whether the intra prediction mode of the current block is Planar mode when the intra

prediction mode of the current block is comprised in the set of most probable modes, and the closest neighboring reference line to the current block is used for intra prediction.

As an implantation of the second aspect, the bitstream includes a first flag indicating whether the intra prediction mode of the current block is comprised in the set of most probable modes, and a second flag indicating whether the intra prediction mode of the current block is Planar mode. As an example, the first flag is intra_luma_mpm_flag, and the second flag is intra_luma_not_planar_flag.

The method according to the first aspect of the invention can be performed by a decoding device according to the third aspect of the invention. The decoding device includes a first obtaining unit and a second obtaining unit. The first obtaining unit, configured to obtain a value of a reference index line of a current block, and a value of a first indication information of the current block, the value of the first indication information indicating whether an intra prediction mode of the current block is comprised in a set of most probable modes. The set of most probable modes comprises 5 candidate intra prediction modes and Planar mode. The second obtaining unit, configure to obtain a value of a second indication information of the current block when the value of the first indication information indicates the intra prediction mode of the current block is comprised in the set of most probable modes, and when the value of the reference index line indicates the closest neighboring reference line to the current block is used, wherein the value of the second indication information of the current block indicates whether the intra prediction mode of the current block is Planar mode.

Further features and implementation forms of the method according to the third aspect of the invention correspond to the features and implementation forms of the apparatus according to the first aspect of the invention.

The method according to the second aspect of the invention can be performed by an encoding device according to the fourth aspect of the invention. The encoding device includes a determining unit and an encoding unit. The determining unit, configured to determine whether an intra prediction mode of a current block is comprised in a set of most probable modes, and to determine whether the closest neighboring reference line to the current block is used for intra prediction. The set of most probable modes comprises 5 candidate intra prediction modes and Planar mode. The encoding unit, configured to encode a bitstream,

wherein the bitstream includes information to indicate whether the intra prediction mode of the current block is Planar mode when the intra prediction mode of the current block is comprised in the set of most probable modes, and the closest neighboring reference line to the current block is used for intra prediction.

Further features and implementation forms of the method according to the fourth aspect of the invention correspond to the features and implementation forms of the apparatus according to the second aspect of the invention.

According to a fifth aspect 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.

According to a sixth aspect 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 second aspect.

According to a seventh 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 or second aspect or any possible embodiment of the first or second aspect.

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

Planar mode has a high frequency used as an intra prediction mode. The coding efficiency of the coding process can be enhanced by employment of the information to indicate whether the intra prediction mode of the current block is Planar mode as compared to the art, when the intra prediction mode of the current block is comprised in the set of most probable modes, and the closest neighboring reference line to the current block is used for intra prediction.

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 block diagram illustrating an example of Multiple Reference Line (MRL) coding tool;

FIG. 7 is a block diagram illustrating an example of division of 4×8 and 8×4 blocks;

FIG. 8 is a block diagram illustrating an example of division of all blocks except 4×8, 8×4 and 4×4;

FIG. 9 is a block diagram illustrating an example of CABAC context modeling/coding;

FIG. 10 is a block diagram illustrating an example of Left (L) and above (A) neighboring blocks of the current block;

FIG. 11 is a block diagram illustrating intra prediction modes;

FIG. 12A illustrates embodiments of a method 1200 of intra prediction of a current block implemented by a decoding device according to the present invention;

FIG. 12B illustrates embodiments of a method 1210 of intra prediction of a current block implemented by a decoding device according to the present invention;

FIG. 13 illustrates embodiments of a method 1300 of intra prediction of a current block implemented by an encoding device according to the present invention;

FIG. 14 illustrates embodiments of a decoding device 1400 for use in an image decoder according to the present invention;

FIG. 15 illustrates embodiments of an encoding device 1500 for use in an image encoder according to the present invention;

FIG. 16 is a block diagram showing an example structure of a content supply system 3100 which realizes a content delivery service; and

FIG. 17 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 to encode the picture 17 block by block, e.g. the encoding and prediction is performed per block 203.

Embodiments of the video encoder 20 as shown in Fig. 2 may be further configured to partition and/or encode the picture by using slices (also referred to as video slices), wherein a

picture may be partitioned into or encoded using one or more slices (typically non-overlapping), and each slice may comprise one or more blocks (e.g. CTUs).

Embodiments of the video encoder 20 as shown in Fig. 2 may be further configured to partition and/or encode the picture by using tile groups (also referred to as video tile groups) and/or tiles (also referred to as video tiles), wherein a picture may be partitioned into or encoded using one or more tile groups (typically non-overlapping), and each tile group may comprise, e.g. one or more blocks (e.g. CTUs) or one or more tiles, wherein each tile, e.g. may be of rectangular shape and may comprise one or more blocks (e.g. CTUs), e.g. complete or fractional blocks.

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), a combined Quad-tree and binary tree (QTBT) partitioning is for example 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 may 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 (e.g. 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.

The motion compensation unit may also generate syntax elements associated with the blocks and video slices for use by video decoder 30 in decoding the picture blocks of the video slice. In addition or as an alternative to slices and respective syntax elements, tile groups and/or tiles and respective syntax elements may be generated or used.

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/or tile groups or tiles) 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, a mode application unit 360, 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 application 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. In addition or as an alternative to slices and respective syntax elements, tile groups and/or tiles and respective syntax elements may be received and/or used.

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 (or tile or tile group) 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 application 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 application 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 application 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. The same or similar may be applied for or by embodiments using tile groups (e.g. video tile groups) and/or tiles (e.g. video tiles) in addition or alternatively to slices (e.g. video slices), e.g. a video may be coded using I, P or B tile groups and /or tiles.

Mode application unit 360 is configured to determine the prediction information for a video block of the current video slice by parsing the motion vectors or related information 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 application 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. The same or similar may be applied for or by embodiments using tile groups (e.g. video tile groups) and/or tiles (e.g. video tiles) in addition or alternatively to slices (e.g. video slices), e.g. a video may be coded using I, P or B tile groups and/or tiles.

Embodiments of the video decoder 30 as shown in Fig. 3 may be configured to partition and/or decode the picture by using slices (also referred to as video slices), wherein a picture may be partitioned into or decoded using one or more slices (typically non-overlapping), and each slice may comprise one or more blocks (e.g. CTUs).

Embodiments of the video decoder 30 as shown in Fig. 3 may be configured to partition and/or decode the picture by using tile groups (also referred to as video tile groups) and/or tiles (also referred to as video tiles), wherein a picture may be partitioned into or decoded using one or more tile groups (typically non-overlapping), and each tile group may comprise, e.g. one or more blocks (e.g. CTUs) or one or more tiles, wherein each tile, e.g. may be of rectangular shape and may comprise one or more blocks (e.g. CTUs), e.g. complete or fractional blocks.

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.

Background Intra prediction mode coding
In current VTM 4.0, a 6-MPM (Most Probable Modes) list is constructed for intra prediction mode coding. The MPM list is introduced to reduce bits required for coding the current block's intra prediction mode. The 6-MPM list is constructed based on intra prediction modes of neighboring blocks of the current block. When intra prediction mode of the current block falls into the MPM list, an index rather than actual mode is coded, which can use less bits. When the intra prediction mode of the current block doesn't fall into the MPM list, truncated binarization is used for coding the intra prediction mode of the current block. MPM list provides a good prediction for intra prediction mode of the current block.

MPM list for multiple reference line (MRL).
In VTM 4.0, Multiple Reference Line (MRL) coding tool can use one of the multiple neighboring reference lines to predict the samples of the current block. When a value of MRL line index equal to 0 (i.e. the closest neighboring reference line is used, i.e. reference line 0 as shown in Fig. 6), a normal 6 MPM list containing planar and DC is used. When a value of MRL line index is not equal to 0, a 6-MPM list excluding planar (i.e. a value of 0) and DC (i.e. a value of 1) is used.

MPM list for intra sub-partition coding mode (ISP)
Intra sub-partition (ISP) coding mode is a tool newly adopted in VTM4.0 (JVET-M0102). Luma intra-predicted blocks are divided vertically or horizontally into 2 or 4 sub-partitions,

depending on the block size dimensions, as shown in Table 1. Figure 7 and Figure 8 show examples of the two possibilities. All sub-partitions fulfill the condition of having at least 16 samples.

Table1: Number of sub-partitions depending on the block size

Block Size	Number of Sub-Partitions
4×4	Not divided
4×8 and 8×4	2
All other cases	4

When intra sub-partition coding mode is enabled, another MPM list excluding DC mode is used in VTM 4.0. The intra sub-partition mode can be applied when a value of multiple reference line index equals to 0 (i.e. MRL is not applied to the current intra prediction block). All sub-partition shares a single intra prediction mode, therefore, the MPM list is constructed once for the intra block and shared by all sub-partitions.

The construction of MPM might depends on the splitting mode of ISP. Two splitting modes are determined: horizontal or vertical. They are shown as in the figure 7 and Figure 8, where horizontal splitting/partition is shown at the top, and vertical splitting/partition is shown at the bottom.

The following Table 2 summarizes the characteristics of the 3 MPM list used in VTM 4.0.

Table 2: different 6-MPM used in VTM 4.0 under different circumstances

MRL config	ISP config	MPM list characteristic
$MRL! = 0$	-	6 MPM but excluding planar mode and DC mode
$MRL == 0$	1	6 MPM but excluding DC mode
$MRL == 0$	0	6 MPM including planar mode and DC mode

Background about CABAC context modeling/coding.

As shown in Figure 9, CABAC coding comprised of binarization, context modeling, and binary arithmetic coding. Binarization maps the syntax elements to binary symbols (bins). Context modeling estimates the probability of each non-bypassed (i.e., regular coded) bin based on some specific context. Finally, binary arithmetic coding compresses the bins to bits according to the estimated probability.

By decomposing each non-binary syntax element value into a sequence of bins, further processing of each bin value in CABAC depends on the associated coding mode decision, which can be either chosen as the regular or the bypass mode. The latter is chosen for bins, which are assumed to be uniformly distributed and for which, consequently, the whole regular binary arithmetic encoding (and decoding) process is simply bypassed. In the regular

coding mode, each bin value is encoded by using the regular binary arithmetic coding engine, where the associated probability model is either determined by a fixed choice, based on the type of syntax element and the bin position or bin index (binIdx) in the binarized representation of the syntax element, or adaptively chosen from two or more probability models depending on the related side information.

Selection of the probability model is referred to as context modeling. Each probability model in CABAC is addressed using a unique context index (ctxIdx), either determined by a fixed assignment or computed by the context derivation logic by which, in turn, the given context model is specified.

Background about how to determine left and above neighbors.
The intra prediction mode coding of VVC depends on the intra prediction modes of its neighboring blocks. Namely The left and above blocks of the current block. Their position is shown in Figure 10.

Signaling of MPM index
A flag (i.e. MPM flag) is used to indicate whether the current block's intra prediction mode is in the MPM list or not. When MPM flag is true (i.e. a value of 1), then the intra prediction mode can be determined using a MPM index. The MPM index is coded using truncated unary code, as shown in the following Table 3 when the length of MPM list is 6. When MPM flag is not true, the intra prediction mode of current block is not in the MPM list, and the mode is coded using truncated binary code. Another example of MPM index when the MPM list is comprised of 5 modes is shown in Table 4. The bin string for coding index is called truncated unary code (not truncated binary code for coding non-MPM modes). Note that for truncated unary code the bin 0 and 1 are interchangeable without loss of generality. In one example, the MPM index when the MPM list is comprised of 5 modes can also be coded as shown in Table 5.

Table 3: MPM flag and MPM index coding when 6 entries in a MPM list

Intra prediction modes	MPM flag	MPM index	Bin string of MPM index
MPM modes (6)	1	0	0

		1	10
		2	110
		3	1110
		4	11110
		5	11111
	0		Truncated binary (TB) code

Table 4: MPM flag and MPM index coding, when 5 entries in a MPM list

Intra prediction modes	MPM flag	MPM index	Bin string of MPM index
MPM modes (5)	1	0	0
		1	10
		2	110
		3	1110
		4	1111
	0		Truncated binary (TB) code

Table 5: MPM flag and MPM index coding, when 5 entries in a MPM list, and alternative codeword for MPM index

Intra prediction modes	MPM flag	MPM index	Bin string of MPM index
MPM modes (5)	1	0	1
		1	01
		2	001
		3	0001
		4	0000
	0		Truncated binary

			(TB) code

Background about planar mode, DC, horizontal, vertical modes and angular modes: Intra prediction modes planar (a value of 0) and DC (a value of 1). Other intra prediction modes are called angular prediction modes.

In VTM4.0, the following syntax design, as shown in Table 6 is used for signaling intra prediction mode information.

Table 6: Syntax parsing in VTM4.0

if(intra_luma_ref_idx[x0][y0] = = 0 && intra_subpartitions_mode_flag[x0][y0] = = 0)	
intra_luma_mpm_flag[x0][y0]	ae(v)
if(intra_luma_mpm_flag[x0][y0]){	
intra_luma_mpm_idx[x0][y0]	ae(v)
}	
else	
intra_luma_mpm_remainder[x0][y0]	ae(v)

intra_luma_ref_idx[x0][y0] can have three values, 0, 1, or 2, indicates which reference line is used. When **intra_luma_ref_idx[x0][y0]** is not present it is inferred to be equal to 0.

intra_subpartitions_mode_flag[x0][y0] equal to 1 specifies that the current intra coding unit is further partitioned (i.e ISP mode is enabled). intra_subpartitions_mode_flag[x0][y0] equal to 0 specifies that the current intra coding unit is not partitioned into rectangular transform block subpartitions. When intra_subpartitions_mode_flag[x0][y0] is not present, it is inferred to be equal to 0.

The syntax elements **intra_luma_mpm_flag**[x0][y0], **intra_luma_mpm_idx**[x0][y0] and **intra_luma_mpm_remainder**[x0][y0] specifies the intra prediction mode for luma samples.

intra_luma_mpm_flag[x0][y0] indicates whether the intra prediction mode of the current block is derived from the MPM list or not. If the value of the **intra_luma_mpm_flag**[x0][y0] is 1, it indicates the intra prediction mode of the current block is derived from the MPM list; If the value of the **intra_luma_mpm_flag**[x0][y0] is 0, it indicates the intra prediction mode of the current block is not derived from the MPM list. When intra_luma_mpm_flag[x0][y0] is not present, it is inferred to be equal to 1. intra_luma_mpm_flag[x0][y0] corresponds to **MPM flag** in Table 3.

intra_luma_mpm_idx corresponds to **MPM index** in Table 3 and use truncated unary code as shown in Table 3.

intra_luma_mpm_remainder represents truncated binary code of mode that is not in the MPM list.

In an embodiment 1, a syntax element intra_luma_planar_flag and ctx are disclosed.

According to one embodiment of the invention, a flag intra_luma_planar_flag is introduced. If a value of the intra_luma_mpm_flag is true (for examples, the value is 1), the decoder parses an intra_luma_planar_flag to determine whether an intra prediction mode of a current block is a Planar mode or not. When a value of the intra_luma_planar_flag is 1, the intra prediction mode of current block is a Planar mode. When a value of the intra_luma_planar_flag is 0, the intra prediction mode of current block is NOT a Planar mode. If the intra_luma_planar_flag is not present (for example, this flag can't be parsed from a bitstream in case **intra_luma_mpm_flag is equal to 0**), a value of the intra_luma_planar_flag is inferred as 0.

If written in a style of VVC working draft specification, the intra prediction mode of the current block

IntraPredModeY[xCb][yCb] is derived by applying the following procedure:

- If intra_luma_mpm_flag[xCb][yCb] is equal to 1,
  - If intra_luma_planar_flag[xCb][yCb] is equal to 1, intraPredModeY[xCb][yCb] is set equal to INTRA_PLANAR.
  - Otherwise, the IntraPredModeY[xCb][yCb] is set equal to candModeList[intra_luma_mpm_idx[xCb][yCb]].

If the intra prediction mode is not a planar mode (i.e a value of the intra_luma_planar_flag is 0), the decoder further parses a syntax element intra_luma_mpm_idx.

In one example, a value of the intra_luma_planar_flag is CABAC contexted coded (i.e. not CABAC by-pass coded), and decoder parses the flag using context based on whether current block is a normal intra block or not.

In one example, a value of the intra_luma_planar_flag is CABAC contexted coded (i.e. not CABAC by-pass coded), and decoder parse the flag using context based on whether current block is an intra block with multiple reference line (MRL) enabled or not.

In one example, a value of the intra_luma_planar_flag is CABAC contexted coded (i.e. not CABAC by-pass coded), and decoder parse the flag using context based on whether current block is an intra block with intra sub-partition (ISP) coding mode enabled or not.

In an example, the position of intra_luma_planar_flag is placed as in Table 7.

Table 7: syntax with intra_luma_planar_flag

if(intra_luma_ref_idx[x0][y0] = = 0 && intra_subpartitions_mode_flag[x0][y0] = = 0)	
intra_luma_mpm_flag[x0][y0]	ae(v)
if(intra_luma_mpm_flag[x0][y0]){	
intra_luma_planar_flag [x0][y0]	ae(v)
if (!**intra_luma_planar_flag** [x0][y0])	
intra_luma_mpm_idx[x0][y0]	ae(v)
}	
else	
intra_luma_mpm_remainder[x0][y0]	ae(v)

Embodiment 2: intra_luma_not_planar_flag and ctx

According to one embodiment of the invention, a syntax element intra_luma_not_planar_flag is introduced.

If a value of the intra_luma_mpm_flag is true (for example, the value is 1), the decoder parses intra_luma_not_planar_flag to determine whether the intra prediction mode of current block is a Planar mode or not. When a value of the intra_luma_not_planar_flag is 1, the intra prediction mode of current block is NOT a Planar mode; When a value of the intra_luma_not_planar_flag is 0, the intra prediction mode of current block is a Planar mode. If intra_luma_not_planar_flag is not present in a bitstream, a value of the intra_luma_not_planar_flag is inferred as 1. In an example, the position of intra_luma_not_planar_flag is placed as in Table 8.

If an intra prediction mode of a current block is not a planar mode (i.e a value of the intra_luma_not_planar_flag is 1), the decoder further parses a syntax element intra_luma_mpm_idx.

Written in a style of VVC working draft specification, the intra prediction mode of the current block

IntraPredModeY[xCb][yCb] is derived by applying the following procedure:

- If intra_luma_mpm_flag[xCb][yCb] is equal to 1,
  - If intra_luma_not_planar_flag[xCb][yCb] is equal to 0, intraPredModeY[xCb][yCb] is set equal to INTRA_PLANAR.
  - Otherwise, the IntraPredModeY[xCb][yCb] is set equal to candModeList[intra_luma_mpm_idx[xCb][yCb]].

In one example, a value of the intra_luma_ not_planar_flag is CABAC contexted coded (i.e. not CABAC by-pass coded), and decoder parse the flag using context based on whether current block is a normal intra block or not.

In one example, a value of the intra_luma_ not_planar_flag is CABAC contexted coded (i.e. not CABAC by-pass coded), and decoder parse the flag using context based on whether current block is an intra block with multiple reference line (MRL) enabled or not.
In one example, a value of the intra_luma_ not_planar_flag is CABAC contexted coded (i.e. not CABAC by-pass coded), and decoder parse the flag using context based on whether current block is an intra block with intra sub-partition (ISP) coding mode enabled or not.

Table 8: syntax with intra_luma_not_planar_flag

if(intra_luma_ref_idx[x0][y0] = = 0 && intra_subpartitions_mode_flag[x0][y0] = = 0)	
intra_luma_mpm_flag[x0][y0]	ae(v)
if(intra_luma_mpm_flag[x0][y0]){	
intra_luma_not_planar_flag [x0][y0]	ae(v)
if (intra_luma_not_planar_flag [x0][y0])	
intra_luma_mpm_idx[x0][y0]	ae(v)
}	
else	
intra_luma_mpm_remainder[x0][y0]	ae(v)

Embodiment 3: **intra_luma_planar_flag and MRL, and by pass decoding**

According to one embodiment of the invention, a flag **intra_luma_planar_flag** is introduced.
If a value of the intra_luma_mpm_flag is true (for example, the value is 1), the decoder parses whether a value of the reference index line ((intra_luma_ref_idx) is 0 or not. If a value of the reference index line is 0, a value of the **intra_luma_planar_flag** is parsed to determine whether the intra prediction mode of current block is a Planar mode or not. When a value of the **intra_luma_planar_flag** is **1,** the intra prediction mode of current block is a Planar mode. When a value of the **intra_luma_planar_flag** is **0,** the intra prediction mode of current block is NOT a Planar mode. If **intra_luma_planar_flag** is not present, a value of the **intra_luma_planar_flag** is inferred as 0.
If an intra prediction mode of a current block is not a planar mode (i.e a value of the **intra_luma_planar_flag is 0**), the decoder further parses **intra_luma_mpm_idx.**

In one example, a value of the **intra_luma_planar_flag is CABAC by-pass** coded (i.e. not CABAC contexted coded), and decoder parse the flag without using any CABAC context. The position of **intra_luma_planar_flag** is placed as in Table 9.

Table 9: syntax with intra_luma_planar_flag and it is by-pass coded

if(intra_luma_ref_idx[x0][y0] = = 0 && intra_subpartitions_mode_flag[x0][y0] = = 0)	
intra_luma_mpm_flag[x0][y0]	ae(v)
if(intra_luma_mpm_flag[x0][y0]){	
if(intra_luma_ref_idx[x0][y0] = = 0)	
intra_luma_planar_flag[x0][y0]	ae(v)
if (!**intra_luma_planar_flag** [x0][y0])	
intra_luma_mpm_idx[x0][y0]	ae(v)
}	
else	
intra_luma_mpm_remainder[x0][y0]	ae(v)

Embodiment 4: **intra_luma_not_planar_flag and MRL, and by pass decoding**

According to one embodiment of the invention, a flag **intra_luma_not_planar_flag** is introduced.

If a value of the intra_luma_mpm_flag is true (for example, the value is 1), the decoder parses whether a value of the reference index line is 0 or not. If a value of the reference index line is 0, **intra_luma_not_planar_flag** is parsed to determine whether the intra prediction mode of current block is a Planar mode or not. When a value of the **intra_luma_not_planar_flag** is **1,** the intra prediction mode of current block is NOT a Planar mode. When a value of the **intra_luma_planar_flag** is **0,** the intra prediction mode of current block is a Planar mode. If **intra_luma_not_planar_flag** is not present, a value of the **intra_luma_not_planar_flag** is inferred as 1.

If an intra prediction mode of a current block is not a planar mode (i.e a value of the **intra_luma_not_planar_flag is 1**), the decoder further parses **intra_luma_mpm_idx.**

In one example, a value of the **intra_luma_not_planar_flag is CABAC by-pass** coded (i.e. not CABAC context coded), and decoder parse the flag without using any CABAC context. The position of **intra_not_luma_planar_flag** is placed as in Table 10.

Table 10: syntax with intra_luma_not_planar_flag and it is by-pass coded

if(intra_luma_ref_idx[x0][y0] = = 0 && intra_subpartitions_mode_flag[x0][y0] = = 0)	
intra_luma_mpm_flag[x0][y0]	ae(v)
if(intra_luma_mpm_flag[x0][y0]){	
if(intra_luma_ref_idx[x0][y0] = = 0)	
intra_luma_not_planar_flag[x0][y0]	ae(v)
if (**intra_luma_not_planar_flag** [x0][y0])	
intra_luma_mpm_idx[x0][y0]	ae(v)
}	
else	
intra_luma_mpm_remainder[x0][y0]	ae(v)

Embodiment 5: **intra_luma_not_planar_flag and MRL, and cabac contexted coded (the difference to embodiment 4 is that intra_luma_not_planar_flag is cabac context coded instead of by-pass coded).**

According to one embodiment of the invention, a flag **intra_luma_not_planar_flag** is introduced.

The decoder parses a value indicating the used reference line index. When the value of the reference index line is 0, the decoder parses a value of the intra_luma_mpm_flag.

If a value of the intra_luma_mpm_flag is true (for example, the value is 1), the decoder checks whether a value of the reference index line is 0 or not. If a value of the reference index line is 0, **intra_luma_not_planar_flag** is parsed to determine whether the intra prediction mode of current block is a Planar mode or not. When a value of the **intra_luma_not_planar_flag** is **1,** the intra prediction mode of current block is NOT a Planar mode. When a value of the **intra_luma_not_planar_flag** is **0,** the intra prediction mode of current block is a Planar mode. If **intra_luma_not_planar_flag** is not present, a value of the **intra_luma_not_planar_flag** is inferred as 1.

If an intra prediction mode of a current block is not a planar mode (i.e., a value of the **intra_luma_not_planar_flag is 1**), the decoder further parses **intra_luma_mpm_idx.**

In one example, a value of the intra_luma_ not_planar_flag is CABAC contexted coded (i.e. not CABAC by-pass coded), and decoder parse the flag using context based on whether current block is a normal intra block or not.

In one example, a value of the intra_luma_ not_planar_flag is CABAC contexted coded (i.e. not CABAC by-pass coded), and decoder parse the flag using context based on whether current block is an intra block with multiple reference line (MRL) enabled or not.
In one example, a value of the intra_luma_ not_planar_flag is CABAC contexted coded (i.e. not CABAC by-pass coded), and decoder parse the flag using context based on whether current block is an intra block with intra sub-partition (ISP) coding mode enabled or not.
The position of **intra_not_luma_planar_flag** is placed as in Table 11.

Table 11: syntax with intra_luma_not_planar_flag and it is **cabac contexted** coded

if(intra_luma_ref_idx[x0][y0] = = 0 && intra_subpartitions_mode_flag[x0][y0] = = 0)	
intra_luma_mpm_flag[x0][y0]	ae(v)
if(intra_luma_mpm_flag[x0][y0]){	
if(intra_luma_ref_idx[x0][y0] = = 0)	
intra_luma_not_planar_flag[x0][y0]	ae(v)
if (**intra_luma_not_planar_flag** [x0][y0])	
intra_luma_mpm_idx[x0][y0]	ae(v)
}	
else	
intra_luma_mpm_remainder[x0][y0]	ae(v)

In an example,

Add intra_luma_not_planar_flag

if(intra_luma_mpm_flag[x0][y0]) {	
if(intra_luma_ref_idx[x0][y0] = = 0)	
intra_luma_not_planar_flag[x0][y0]	ae(v)
if(intra_luma_not_planar_flag[x0][y0])	
intra_luma_mpm_idx[x0][y0]	ae(v)
} else	
intra_luma_mpm_remainder[x0][y0]	ae(v)
}	

intra_luma_mpm_flag[ ][ ]	FL	cMax = 1
intra_luma_not_planar_flag[ ][ ]	FL	cMax = 1
intra_luma_mpm_idx[ ][ ]	TR	cMax = 4, cRiceParam = 0

When intra_luma_not_planar_flag[x0][y0] is not present, it is inferred to be equal to 1.

The below Table 12 shows an example of assignment of ctxInc to syntax elements with context coded bins.

Table 12 – Assignment of ctxInc to syntax elements with context coded bins

Syntax element	binIdx					
	0	1	2	3	4	>= 5
intra_subpartition_split_flag	0	na	na	na	na	na
intra_luma_mpm_flag[ ][ ]	0	na	na	na	na	na
intra_luma_not_planar_flag[ ][ ]	!intra_subpartitions_mode_flag	na	na	na	na	na
intra_luma_mpm_idx[ ][ ]	bypass	bypass	bypass	bypass	na	na
intra_luma_mpm_remainder[ ][ ]	bypass	bypass	bypass	bypass	bypass	bypass

In an embodiment, a MPM list is constructed according to the following process.

8.4.2 Derivation process for luma intra prediction mode

Input to this process are:

– a luma location (xCb , yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,

– a variable cbWidth specifying the width of the current coding block in luma samples,

– a variable cbHeight specifying the height of the current coding block in luma samples.

In this process, the luma intra prediction mode IntraPredModeY[xCb][yCb] is derived.

Table 13 specifies the value for the intra prediction mode IntraPredModeY[xCb][yCb] and the associated names.

Table 13 – Specification of intra prediction mode and associated names

Intra prediction mode	Associated name
0	INTRA_PLANAR
1	INTRA_DC
2..66	INTRA_ANGULAR2..INTRA_ANGULAR66
81..83	INTRA_LT_CCLM, INTRA_L_CCLM, INTRA_T_CCLM

NOTE – : The intra prediction modes INTRA_LT_CCLM, INTRA_L_CCLM and INTRA_T_CCLM are only applicable to chroma components.

IntraPredModeY[xCb][yCb] is derived as follows:

– If intra_luma_not_planar_flag[xCb][yCb] is equal to 1, the following ordered steps apply:

1. The neighbouring locations (xNbA, yNbA) and (xNbB, yNbB) are set equal to (xCb − 1, yCb + cbHeight − 1) and (xCb + cbWidth − 1, yCb − 1), respectively.
2. For X being replaced by either A or B, the variables candIntraPredModeX are derived as follows:
   - The availability derivation process for a block is invoked: Neighbouring blocks availability checking process is invoked with the location (xCurr, yCurr) set equal to (xCb, yCb) and the neighbouring location (xNbY, yNbY) set equal to (xNbX, yNbX) as inputs, and the output is assigned to availableX.
   - The candidate intra prediction mode candIntraPredModeX is derived as follows:
     - If one or more of the following conditions are true, candIntraPredModeX is set equal to INTRA_PLANAR.
       - The variable availableX is equal to FALSE.
       - CuPredMode[xNbX][yNbX] is not equal to MODE_INTRA and ciip_flag[xNbX][yNbX] is not equal to 1.
       - pcm_flag[xNbX][yNbX] is equal to 1.
       - X is equal to B and yCb − 1 is less than ((yCb >> CtbLog2SizeY) << CtbLog2SizeY).
     - Otherwise, candIntraPredModeX is set equal to IntraPredModeY[xNbX][yNbX].

where X being replaced by either neighbor block A or B, a luma location (xCb , yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture, the location (xNbA, yNbA) of the neighbor A is set equal to (xCb − 1, yCb + cbHeight − 1), the location (xNbB, yNbB) of the neighbor B is set equal to (xCb + cbWidth − 1, yCb − 1), a variable cbWidth specifying the width of the current coding block in luma samples, a variable cbHeight specifying the height of the current coding block in luma samples, CuPredMode representing a currect prediction mode, SizeY representing the size of Y components of a coding tree block.

3. The candModeList[x] with x = 0..4 is derived as follows:

– If candIntraPredModeB is equal to candIntraPredModeA and candIntraPredModeA is greater than INTRA_DC, candModeList[x] with x = 0..4 is derived as follows:

candModeList[0] = candIntraPredModeA (8-10)

candModeList[1] = 2 + ((candIntraPredModeA + 61) % 64) (8-12)

candModeList[2] = 2 + ((candIntraPredModeA − 1) % 64) (8-13)

candModeList[3] = INTRA_DC (8-11)

candModeList[4] = 2 + ((candIntraPredModeA + 60) % 64) (8-14)

– Otherwise if candIntraPredModeB is not equal to candIntraPredModeA and candIntraPredModeA or candIntraPredModeB is greater than INTRA_DC, the following applies:

– The variables minAB and maxAB are derived as follows:

minAB = Min(candIntraPredModeA, candIntraPredModeB) (8-24)

maxAB = Max(candIntraPredModeA, candIntraPredModeB) (8-25)

– If candIntraPredModeA and candIntraPredModeB are both greater than INTRA_DC, candModeList[x] with x = 0..4 is derived as follows:

candModeList[0] = candIntraPredModeA (8-27)

candModeList[1] = candIntraPredModeB (8-29)

candModeList[2]= INTRA_DC (8-29)

– If maxAB − minAB is in the range of 2 to 62, inclusive, the following applies:

candModeList[3] = 2 + ((maxAB + 61) % 64) (8-30)

candModeList[4] = 2 + ((maxAB − 1) % 64) (8-31)

– Otherwise, the following applies:

candModeList[3] = 2 + ((maxAB + 60) % 64) (8-32)

candModeList[4] = 2 + ((maxAB) % 64) (8-33)

– Otherwise (candIntraPredModeA or candIntraPredModeB is greater than INTRA_DC), candModeList[x] with x = 0..4 is derived as follows:

candModeList[0] = maxAB (8-65)

candModeList[1] = INTRA_D (8-66)

candModeList[2] = 2 + ((maxAB + 61) % 64) (8-66)

candModeList[3] = 2 + ((maxAB − 1) % 64) (8-67)

candModeList[4] = 2 + ((maxAB + 60) % 64) (8-68)

– Otherwise, the following applies:

candModeList[0] = INTRA_DC (8-71)

candModeList[1] = INTRA_ANGULAR50 (8-72)

candModeList[2] = INTRA_ANGULAR18 (8-73)

candModeList[3] = INTRA_ANGULAR46 (8-74)

candModeList[4] = INTRA_ANGULAR54 (8-75)

4. IntraPredModeY[xCb][yCb] is derived by applying the following procedure:

– If intra_luma_mpm_flag[xCb][yCb] is equal to 1, the IntraPredModeY[xCb][yCb] is set equal to candModeList[intra_luma_mpm_idx[xCb][yCb]].

– Otherwise, IntraPredModeY[xCb][yCb] is derived by applying the following ordered steps:

1. When candModeList[i] is greater than candModeList[j] for i = 0..3 and for each i, j = (i + 1)..4, both values are swapped as follows:

(candModeList[i], candModeList[j]) = Swap(candModeList[i], candModeList[j]) (8-94)

2. IntraPredModeY[xCb][yCb] is derived by the following ordered steps:
   i. IntraPredModeY[xCb][yCb] is set equal to intra_luma_mpm_remainder[xCb][yCb].
   ii. The value of IntraPredModeY[xCb][yCb] is incremented by one.
   iii. For i equal to 0 to 4, inclusive, when IntraPredModeY[xCb][yCb] is greater than or equal to candModeList[i], the value of IntraPredModeY[xCb][yCb] is incremented by one.

– Otherwise (intra_luma_not_planar_flag[xCb][yCb] is equal to 0), IntraPredModeY[xCb][yCb] is set equal to INTRA_PLANAR.

The variable IntraPredModeY[x][y] with x = xCb..xCb + cbWidth − 1 and y = yCb..yCb + cbHeight − 1 is set to be equal to IntraPredModeY[xCb][yCb].

In one example, the intra prediction mode of the current block (e.g. variable name is. luma_intra_mode) is neither a planar mode nor an intra prediction mode in a 5-MPM list, its value is derived and the prediction of this intra prediction mode is performed as follows:

1. Derive the value of intra_luma_mpm_remainder from the bitstream, and luma_intra_mode is set to the value of intra_luma_mpm_remainder.
2. The value of IntraPredModeY[xCb][yCb] is incremented by one
3. For i equal to 0 to 4 (compare with every entry in the 5-MPM list), when luma_intra_mode is greater than or equal to MPM[i] (here MPM[0] corresponds to the first entry in MPM list, MPM[1] corresponds to the second entry in MPM list, and so on), the value of luma_intra_mode is incremented by one.
4. Using the derived intra prediction mode (luma_intra_mode) and corresponding referenece samples to performance intra prediction.

In one implementation of the above embodiments, Planar mode might be implicitly considered as always the first entry in an implicit 6-MPM list. This implicit 6-MPM list is comprised of planar (always the first entry) and 5 other modes (which can be explicit described as 5-MPM list).

This is because if the Planar mode is always in the first position of the implicit 6-MPM list, it can be split out from the implicit MPM list, and a special flag (for example a flag

intra_luma _planar_flag or a flag **intra_luma_not_planar_flag) is assigned to indicate whehter the intra prediction mode of current block is planar or not.**

In one example, considering introducing **intra_luma_not_planar_flag[x0][y0] as a special flag to indicate if planar mode is equal to the intra prediction mode of the current block. When it is planar mode, intra_luma_not_planar_flag** is set to 0, otherwise set to 1. In this case, the first bin of mpm_idx is equivalent to the special flag **intra_luma_not_planar_flag.**

When the intra prediction mode of the current block is planar, MPM index equal to 0, and the coded bin is 0. When the intra prediction mode of the current block is Not planar, MPM index equal is not equal to 0, and the fisrt bin of MPM index is always equal to 1.

In an example, intra_luma_mpm_idx is coded by using truncated unary (TR) code,

	Value	TR
Mpm_idx	0	0
Mpm_idx	1	1 0
Mpm_idx	2	1 1 0
Mpm_idx	3	1 1 1 0
Mpm_idx	4	1 1 1 1.

According to one embodiment of the invention (**non-angular branch**):

- Step 1: obtain an intra prediction mode of a left neighbor block and an intra prediction mode of an above neighbor block according to Figure 10. If an intra prediction mode of the neighbor block is not available (for example, neighbor block is out of picture boundaries or neighbor block is inter coded), the intra prediction mode of the neighbor block is set as planar. The intra prediction mode of left neighbor is denoted as left_mode and the intra prediction mode of above neighbor block is denoted as above_mode.
- Step 2: determining whether the intra prediction modes of the neighbor blocks are angular modes or not.
- Step 3: if all of the following conditions (bullet items) are evaluated to be true,
    - Left_mode is not angular mode and above_mode is not angular mode

then, a list of intra prediction modes are constructed which comprises N entries of intra prediction modes as follows (for example, N equals to 5, where 1st entry corresponds to MPM[1], 2nd entry corresponds to MPM[2], and so on) :

MPM[1]: DC

MPM[2]: vertical mode (i.e. 50)

MPM[3]: horizontal mode (i.e. 18)

MPM[4]: vertical-4 (i.e. 46)

MPM[5]: vertical+4 (i.e. 54)

- Step 4: predicting the current block according to an index (denoted as MPM index) and the entries of the constructed list.

According to one embodiment of the invention (**non-angular branch**):

- Step 1: obtain an intra prediction mode of a left neighbor block and an intra prediction mode of an above neighbor block according to Figure 10. If an intra prediction mode of the neighbor block is not available (for example, neighbor block is out of picture boundaries or neighbor block is inter coded), the intra prediction mode of the neighbor block is set as planar. The intra prediction mode of left neighbor is denoted as left_mode and the intra prediction mode of above neighbor block is denoted as above_mode.
- Step 2: determining whether the intra prediction modes of the neighbor blocks are angular modes or not.
- Step 3: if all of the following conditions (bullet items) are evaluated to be true,
    - Left_mode is not angular mode and above_mode is not angular mode

then, a list of intra prediction modes are constructed which comprises N entries of intra prediction modes as follows (for example, N equals to 5, where 1st entry corresponds to MPM[1], 2nd entry corresponds to MPM[2], and so on) :

MPM[1]: DC

MPM[2]: vertical mode (i.e. 50)

MPM[3]: horizontal mode (i.e. 18)

MPM[4]: 66

MPM[5]: 2

- Step 4: predicting the current block according to an index (denoted as MPM index) and the entries of the constructed list.

According to one embodiment of the invention (**non-angular branch**):

- Step 1: obtain an intra prediction mode of a left neighbor block and an intra prediction mode of an above neighbor block according to Figure 10. If an intra prediction mode of the neighbor block is not available (for example, neighbor block is out of picture boundaries or neighbor block is inter coded), the intra prediction mode of the neighbor block is set as planar. The intra prediction mode of left neighbor is denoted as left_mode and the intra prediction mode of above neighbor block is denoted as above_mode.
- Step 2: determining whether the intra prediction modes of the neighbor blocks are angular modes or not.
- Step 3: if all of the following conditions (bullet items) are evaluated to be true,
    - Left_mode is not angular mode and above_mode is not angular mode

then, a list of intra prediction modes are constructed which comprises N entries of intra prediction modes as follows (for example, N equals to 5, where 1st entry corresponds to MPM[1], 2nd entry corresponds to MPM[2], and so on) :

MPM[1]: DC

MPM[2]: vertical mode (i.e. 50)

MPM[3]: horizontal mode (i.e. 18)

MPM[4]: 2

MPM[5]: 34

- Step 4: predicting the current block according to an index (denoted as MPM index) and the entries of the constructed list.

According to one embodiment of the invention (**non-angular branch**):

- Step 1: obtain an intra prediction mode of a left neighbor block and an intra prediction mode of an above neighbor block according to Figure 10. If an intra prediction mode of the neighbor block is not available (for example, neighbor block is out of picture boundaries or neighbor block is inter coded), the intra prediction mode of the neighbor block is set as planar. The intra prediction mode of left neighbor is denoted as left_mode and the intra prediction mode of above neighbor block is denoted as above_mode.

- Step 2: determining whether the intra prediction modes of the neighbor blocks are angular modes or not.
- Step 3: if all of the following conditions (bullet items) are evaluated to be true,
    - Left_mode is not angular mode and above_mode is not angular mode

then, a list of intra prediction modes are constructed which comprises N entries of intra prediction modes as follows (for example, N equals to 5, where 1st entry corresponds to MPM[1], 2nd entry corresponds to MPM[2] and so on) :

MPM[1]: DC

MPM[2]: vertical mode (i.e. 50)

MPM[3]: horizontal mode (i.e. 18)

MPM[4]: 66

MPM[5]: 34

- Step 4: predicting the current block according to an index (denoted as MPM index) and the entries of the constructed list.

According to one embodiment of the invention (**one-angular branch, one angular the other non-angular**):

- Step 1: obtain an intra prediction mode of a left neighbor block and an intra prediction mode of an above neighbor block according to Figure 10. If an intra prediction mode of the neighbor block is not available (for example, neighbor block is out of picture boundaries or neighbor block is inter coded), the intra prediction mode of the neighbor block is set as planar. The intra prediction mode of left neighbor is denoted as left_mode and the intra prediction mode of above neighbor block is denoted as above_mode.
- Step 2: determining whether the intra prediction modes of the neighbor blocks are angular modes or not.
- Step 3: if all of the following conditions (bullet items) are evaluated to be true,
    - One of the left_mode and above_mode is angular mode, the other one is non-angular mode.

then, a list of intra prediction modes are constructed which comprises N entries of intra prediction modes as follows (for example, N equals to 5, where 1st entry corresponds to MPM[1], 2nd entry corresponds to MPM[2], and so on) :

define an angular mode as ang and obtain as follow:

if left_mode is angular mode, ang=left mode

otherwise (above mode is angular mode) ang=above mode

MPM[1]: ang

MPM[2]: DC

MPM[3]: 2 + ((ang + 61) % 64)

MPM[4]: 2 + ((ang – 1) % 64)

MPM[5]: 2 + ((ang + 60) % 64)

- Step 4: predicting the current block according to an index (denoted as MPM index) and the entries of the constructed list.

According to one embodiment of the invention (**one-angular branch, one angular the other non-angular**):

- Step 1: obtain an intra prediction mode of a left neighbor block and an intra prediction mode of an above neighbor block according to Figure 10. If an intra prediction mode of the neighbor block is not available (for example, neighbor block is out of picture boundaries or neighbor block is inter coded), the intra prediction mode of the neighbor block is set as planar. The intra prediction mode of left neighbor is denoted as left_mode and the intra prediction mode of above neighbor block is denoted as above_mode.
- Step 2: determining whether the intra prediction modes of the neighbor blocks are angular modes or not.
- Step 3: if all of the following conditions (bullet items) are evaluated to be true,
    - One of the left_mode and above_mode is angular mode, the other one is non-angular mode.

then, a list of intra prediction modes are constructed which comprises N entries of intra prediction modes as follows (for example, N equals to 5, where 1st entry corresponds to MPM[1], 2nd entry corresponds to MPM[2], and so on) :

define an angular mode as ang and obtain as follow:

if left_mode is angular mode, ang=left mode

otherwise (above mode is angular mode) ang=above mode

MPM[1]: DC

MPM[2]: ang

MPM[3]: 2 + ((ang + 61) % 64)

MPM[4]: 2 + ((ang – 1) % 64)

MPM[5]: 2 + ((ang + 60) % 64)

- Step 4: predicting the current block according to an index (denoted as MPM index) and the entries of the constructed list.

According to one embodiment of the invention (**one-angular branch, one angular the other non-angular**):

- Step 1: obtain an intra prediction mode of a left neighbor block and an intra prediction mode of an above neighbor block according to Figure 10. If an intra prediction mode of the neighbor block is not available (for example, neighbor block is out of picture boundaries or neighbor block is inter coded), the intra prediction mode of the neighbor block is set as planar. The intra prediction mode of left neighbor is denoted as left_mode and the intra prediction mode of above neighbor block is denoted as above_mode.
- Step 2: determining whether the intra prediction modes of the neighbor blocks are angular modes or not.
- Step 3: if all of the following conditions (bullet items) are evaluated to be true,
    - One of the left_mode and above_mode is angular mode, the other one is non-angular mode.

then, a list of intra prediction modes are constructed which comprises N entries of intra prediction modes as follows (for example, N equals to 5, where 1st entry corresponds to MPM[1], 2nd entry corresponds to MPM[2], and so on) :

define an angular mode as ang and obtain as follow:

if left_mode is angular mode, ang=left mode

otherwise (above mode is angular mode) ang=above mode

MPM[1]: ang

MPM[2]: 2 + ((ang + 61) % 64)

MPM[3]: DC

MPM[4]: 2 + ((ang – 1) % 64)

MPM[5]: 2 + ((ang + 60) % 64)

- Step 4: predicting the current block according to an index (denoted as MPM index) and the entries of the constructed list.

According to one embodiment of the invention (**one-angular branch, one angular the other non-angular**):

- Step 1: obtain an intra prediction mode of a left neighbor block and an intra prediction mode of an above neighbor block according to Figure 10. If an intra prediction mode of the neighbor block is not available (for example, neighbor block is out of picture boundaries or neighbor block is inter coded), the intra prediction mode of the neighbor block is set as planar. The intra prediction mode of left neighbor is denoted as left_mode and the intra prediction mode of above neighbor block is denoted as above_mode.
- Step 2: determining whether the intra prediction modes of the neighbor blocks are angular modes or not.
- Step 3: if all of the following conditions (bullet items) are evaluated to be true,
    - One of the left_mode and above_mode is angular mode, the other one is non-angular mode.

then, a list of intra prediction modes are constructed which comprises N entries of intra prediction modes as follows (for example, N equals to 5, where 1st entry corresponds to MPM[1], 2nd entry corresponds to MPM[2], and so on) :

define an angular mode as ang and obtain as follow:

if left_mode is angular mode, ang=left mode

otherwise (above mode is angular mode) ang=above mode

MPM[0]: Planar

MPM[1]: ang

MPM[2]: 2 + ((ang + 61) % 64)

MPM[3]: 2 + ((ang – 1) % 64)

MPM[4]: DC

MPM[5]: 2 + ((ang + 60) % 64)

- Step 4: predicting the current block according to an index (denoted as MPM index) and the entries of the constructed list.

According to one embodiment of the invention (**one-angular branch, one angular the other non-angular**):

- Step 1: obtain an intra prediction mode of a left neighbor block and an intra prediction mode of an above neighbor block according to Figure 10. If an intra prediction mode of the neighbor block is not available (for example, neighbor block is out of picture boundaries or neighbor block is inter coded), the intra prediction mode of the neighbor block is set as planar. The intra prediction mode of left neighbor is denoted as left_mode and the intra prediction mode of above neighbor block is denoted as above_mode.
- Step 2: determining whether the intra prediction modes of the neighbor blocks are angular modes or not.
- Step 3: if all of the following conditions (bullet items) are evaluated to be true,
    - One of the left_mode and above_mode is angular mode, the other one is non-angular mode.

then, a list of intra prediction modes are constructed which comprises N entries of intra prediction modes as follows (for example, N equals to 5, where 1st entry corresponds to MPM[1], 2nd entry corresponds to MPM[2], and so on) :

define an angular mode as ang and obtain as follow:

if left_mode is angular mode, ang=left mode

otherwise (above mode is angular mode) ang=above mode

MPM[1]: ang

MPM[2]: 2 + ((ang + 61) % 64)

MPM[3]: 2 + ((ang – 1) % 64)

MPM[4]: 2 + ((ang + 60) % 64)

MPM[5]: DC

- Step 4: predicting the current block according to an index (denoted as MPM index) and the entries of the constructed list.

According to one embodiment of the invention (**one-angular branch, two are same angular**):

- Step 1: obtain an intra prediction mode of a left neighbor block and an intra prediction mode of an above neighbor block according to Figure 10. If an intra prediction mode of the neighbor block is not available (for example, neighbor block is out of picture

boundaries or neighbor block is inter coded), the intra prediction mode of the neighbor block is set as planar. The intra prediction mode of left neighbor is denoted as left_mode and the intra prediction mode of above neighbor block is denoted as above_mode.

- Step 2: determining whether the intra prediction modes of the neighbor blocks are angular modes or not.
- Step 3: if all of the following conditions (bullet items) are evaluated to be true,
  - left_mode and above_mode are both angular modes and they are the same.

then, a list of intra prediction modes are constructed which comprises N entries of intra prediction modes as follows (for example, N equals to 5, where 1st entry corresponds to MPM[1], 2nd entry corresponds to MPM[2], and so on) :

MPM[1]: left mode

MPM[2]: 2 + ((left mode + 61) % 64)

MPM[3]: 2 + ((left mode – 1) % 64)

MPM[4]: DC

MPM[5]: 2 + ((left mode + 60) % 64)

- Step 4: predicting the current block according to an index (denoted as MPM index) and the entries of the constructed list.

According to one embodiment of the invention (**one-angular branch, two are same angular**):

- Step 1: obtain an intra prediction mode of a left neighbor block and an intra prediction mode of an above neighbor block according to Figure 10. If an intra prediction mode of the neighbor block is not available (for example, neighbor block is out of picture boundaries or neighbor block is inter coded), the intra prediction mode of the neighbor block is set as planar. The intra prediction mode of left neighbor is denoted as left_mode and the intra prediction mode of above neighbor block is denoted as above_mode.
- Step 2: determining whether the intra prediction modes of the neighbor blocks are angular modes or not.
- Step 3: if all of the following conditions (bullet items) are evaluated to be true,
  - left_mode and above_mode are both angular modes and they are the same.

then, a list of intra prediction modes are constructed which comprises N entries of intra prediction modes as follows (for example, N equals to 5, where 1st entry corresponds to MPM[1], 2nd entry corresponds to MPM[2], and so on) :

MPM[1]: DC

MPM[2]: left mode

MPM[3]: 2 + ((left mode + 61) % 64)

MPM[4]: 2 + ((left mode – 1) % 64)

MPM[5]: 2 + ((left mode + 60) % 64)

- Step 4: predicting the current block according to an index (denoted as MPM index) and the entries of the constructed list.

According to one embodiment of the invention (**one-angular branch, two are same angular**):

- Step 1: obtain an intra prediction mode of a left neighbor block and an intra prediction mode of an above neighbor block according to Figure 10. If an intra prediction mode of the neighbor block is not available (for example, neighbor block is out of picture boundaries or neighbor block is inter coded), the intra prediction mode of the neighbor block is set as planar. The intra prediction mode of left neighbor is denoted as left_mode and the intra prediction mode of above neighbor block is denoted as above_mode.
- Step 2: determining whether the intra prediction modes of the neighbor blocks are angular modes or not.
- Step 3: if all of the following conditions (bullet items) are evaluated to be true,
  - left_mode and above_mode are both angular modes and they are the same.

then, a list of intra prediction modes are constructed which comprises N entries of intra prediction modes as follows (for example, N equals to 5, where 1st entry corresponds to MPM[1], 2nd entry corresponds to MPM[2], and so on) :

MPM[1]: left mode

MPM[2]: DC

MPM[3]: 2 + ((left mode + 61) % 64)

MPM[4]: 2 + ((left mode – 1) % 64)

MPM[5]: 2 + ((left mode + 60) % 64)

- Step 4: predicting the current block according to an index (denoted as MPM index) and the entries of the constructed list.

According to one embodiment of the invention (**one-angular branch, two are same angular**):

- Step 1: obtain an intra prediction mode of a left neighbor block and an intra prediction mode of an above neighbor block according to Figure 10. If an intra prediction mode of the neighbor block is not available (for example, neighbor block is out of picture boundaries or neighbor block is inter coded), the intra prediction mode of the neighbor block is set as planar. The intra prediction mode of left neighbor is denoted as left_mode and the intra prediction mode of above neighbor block is denoted as above_mode.
- Step 2: determining whether the intra prediction modes of the neighbor blocks are angular modes or not.
- Step 3: if all of the following conditions (bullet items) are evaluated to be true,
    - left_mode and above_mode are both angular modes and they are the same.

then, a list of intra prediction modes are constructed which comprises N entries of intra prediction modes as follows (for example, N equals to 5, where 1st entry corresponds to MPM[1], 2nd entry corresponds to MPM[2], and so on) :

MPM[1]: left mode
MPM[2]: 2 + ((left mode + 61) % 64)
MPM[3]: DC
MPM[4]: 2 + ((left mode – 1) % 64)
MPM[5]: 2 + ((left mode + 60) % 64)

- Step 4: predicting the current block according to an index (denoted as MPM index) and the entries of the constructed list.

According to one embodiment of the invention (**one-angular branch, two are same angular**):

- Step 1: obtain an intra prediction mode of a left neighbor block and an intra prediction mode of an above neighbor block according to Figure 10. If an intra prediction mode of the neighbor block is not available (for example, neighbor block is out of picture

boundaries or neighbor block is inter coded), the intra prediction mode of the neighbor block is set as planar. The intra prediction mode of left neighbor is denoted as left_mode and the intra prediction mode of above neighbor block is denoted as above_mode.

- Step 2: determining whether the intra prediction modes of the neighbor blocks are angular modes or not.
- Step 3: if all of the following conditions (bullet items) are evaluated to be true,
  - left_mode and above_mode are both angular modes and they are the same.

then, a list of intra prediction modes are constructed which comprises N entries of intra prediction modes as follows (for example, N equals to 5, where 1st entry corresponds to MPM[1], 2nd entry corresponds to MPM[2], and so on) :

MPM[1]: left mode

MPM[2]: 2 + ((left mode + 61) % 64)

MPM[3]: 2 + ((left mode – 1) % 64)

MPM[4]: 2 + ((left mode + 60) % 64)

MPM[5]: DC

- Step 4: predicting the current block according to an index (denoted as MPM index) and the entries of the constructed list.

According to one embodiment of the invention (**two-angular branch**):

- Step 1: obtain an intra prediction mode of a left neighbor block and an intra prediction mode of an above neighbor block according to Figure 10. If an intra prediction mode of the neighbor block is not available (for example, neighbor block is out of picture boundaries or neighbor block is inter coded), the intra prediction mode of the neighbor block is set as planar. The intra prediction mode of left neighbor is denoted as left_mode and the intra prediction mode of above neighbor block is denoted as above_mode.
- Step 2: determining whether the intra prediction mode of the neighbor blocks are angular modes or not.
- Step 3: if all of the following conditions (bullet items) are evaluated to be true,
  - left_mode and above_mode are two different angular modes.

then, a list of intra prediction modes are constructed which comprises N entries of intra prediction modes as follows (for example, N equals to 5, where 1st entry corresponds to MPM[1], 2nd entry corresponds to MPM[2], and so on) :

MPM[1]: left_mode

MPM[2]: above_mode

MPM[3]: DC

– The variables minAB and maxAB are derived as follows:

minAB = Min(above_mode, left_mode)

maxAB = Max(above_mode, left_mode)

– If maxAB − minAB is in the range of 2 to 62, inclusive, the following applies:

MPM [4] = 2 + ((maxAB + 61) % 64) (8-30)

MPM [5] = 2 + ((maxAB − 1) % 64) (8-31)

– Otherwise, the following applies:

MPM [4] = 2 + ((maxAB + 60) % 64) (8-32)

MPM [5] = 2 + ((maxAB) % 64) (8-33)

- Step 4: predicting the current block according to an index (denoted as MPM index) and the entries of the constructed list.

According to one embodiment of the invention (**two-angular branch**):

- Step 1: obtain an intra prediction mode of a left neighbor block and an intra prediction mode of an above neighbor block according to Figure 10. If an intra prediction mode of the neighbor block is not available (for example, neighbor block is out of picture boundaries or neighbor block is inter coded), the intra prediction mode of the neighbor block is set as planar. The intra prediction mode of left neighbor is denoted as left_mode and the intra prediction mode of above neighbor block is denoted as above_mode.
- Step 2: determining whether the intra prediction mode of the neighbor blocks are angular modes or not.
- Step 3: if all of the following conditions (bullet items) are evaluated to be true,
  - left_mode and above_mode are two different angular modes.

then, a list of intra prediction modes are constructed which comprises N entries of intra prediction modes as follows (for example, N equals to 5, where 1st entry corresponds to MPM[1], 2nd entry corresponds to MPM[2], and so on) :

MPM[1]: above_mode

MPM[2]: left_mode

MPM[3]: DC

- The variables minAB and maxAB are derived as follows:

  minAB = Min(above_mode, left_mode)

  maxAB = Max(above_mode, left_mode)

  - If maxAB − minAB is in the range of 2 to 62, inclusive, the following applies:

    MPM [4] = 2 + ((maxAB + 61) % 64) (8-30)

    MPM [5] = 2 + ((maxAB − 1) % 64) (8-31)

  - Otherwise, the following applies:

    MPM [4] = 2 + ((maxAB + 60) % 64) (8-32)

    MPM [5] = 2 + ((maxAB) % 64) (8-33)

- Step 6: predicting the current block according to an index (denoted as MPM index) and the entries of the constructed list.

According to one embodiment of the invention (**two-angular branch**):

- Step 1: obtain an intra prediction mode of a left neighbor block and an intra prediction mode of an above neighbor block according to Figure 10. If an intra prediction mode of the neighbor block is not available (for example, neighbor block is out of picture boundaries or neighbor block is inter coded), the intra prediction mode of the neighbor block is set as planar. The intra prediction mode of left neighbor is denoted as left_mode and the intra prediction mode of above neighbor block is denoted as above_mode.
- Step 2: determining whether the intra prediction mode of the neighbor blocks are angular modes or not.
- Step 3: if all of the following conditions (bullet items) are evaluated to be true,
  - left_mode and above_mode are two different angular modes.

then, a list of intra prediction modes are constructed which comprises N entries of intra prediction modes as follows (for example, N equals to 5, where 1st entry corresponds to MPM[1], 2nd entry corresponds to MPM[2], and so on) :

MPM[1]: left_mode

MPM[2]: DC

MPM[3]: above_mode

– The variables minAB and maxAB are derived as follows:

minAB = Min(above_mode, left_mode)

maxAB = Max(above_mode, left_mode)

– If maxAB − minAB is in the range of 2 to 62, inclusive, the following applies:

MPM [4] = 2 + ((maxAB + 61) % 64) (8-30)

MPM [5] = 2 + ((maxAB − 1) % 64) (8-31)

– Otherwise, the following applies:

MPM [4] = 2 + ((maxAB + 60) % 64) (8-32)

MPM [5] = 2 + ((maxAB) % 64) (8-33)

- Step 4: predicting the current block according to an index (denoted as MPM index) and the entries of the constructed list.

According to one embodiment of the invention (**two-angular branch**):

- Step 1: obtain an intra prediction mode of a left neighbor block and an intra prediction mode of an above neighbor block according to Figure 10. If an intra prediction mode of the neighbor block is not available (for example, neighbor block is out of picture boundaries or neighbor block is inter coded), the intra prediction mode of the neighbor block is set as planar. The intra prediction mode of left neighbor is denoted as left_mode and the intra prediction mode of above neighbor block is denoted as above_mode.
- Step 2: determining whether the intra prediction mode of the neighbor blocks are angular modes or not.
- Step 3: if all of the following conditions (bullet items) are evaluated to be true,
    - left_mode and above_mode are two different angular modes.

then, a list of intra prediction modes are constructed which comprises N entries of intra prediction modes as follows (for example, N equals to 5, where 1st entry corresponds to MPM[1], 2nd entry corresponds to MPM[2], and so on) :

MPM[1]: above_mode

MPM[2]: DC

MPM[3]: left_mode

– The variables minAB and maxAB are derived as follows:

minAB = Min(above_mode, left_mode)

maxAB = Max(above_mode, left_mode)

– If maxAB − minAB is in the range of 2 to 62, inclusive, the following applies:

MPM [4] = 2 + ((maxAB + 61) % 64) (8-30)

MPM [5] = 2 + ((maxAB − 1) % 64) (8-31)

– Otherwise, the following applies:

MPM [4] = 2 + ((maxAB + 60) % 64) (8-32)

MPM [5] = 2 + ((maxAB) % 64) (8-33)

- Step 4: predicting the current block according to an index (denoted as MPM index) and the entries of the constructed list.

In one implementation of the embodiment, the N is set equal to 6.

In one implementation of the embodiment, the N is set equal to 6. In this case the said list of intra prediction modes includes the first 5 entries as listed above and not the sixth entry.

In one implementation of the embodiment, the N is set equal to 4. In this case the said list of intra prediction modes includes the first 4 entries as listed above and not the fifth entry and the sixth entry.

In one implementation of the embodiment, said list of intra prediction modes is called an MPM list.

In one implementation, said list index (MPM index) is signaled in the bitstream as an indicator. Advantageously the MPM index can be coded with Context Adaptive Entropy Coder (CABAC). The MPM index might be coded according to various number of probability models (in other words context) by CABAC.

For one example, the first bin of the codedword for mpm_index is CABAC context coded.

In one example, its context is decided based on whether the current intra block is applied with multiple reference line, ISP or normal intra blocks. In total three contexts are created for the CABAC context of the first bin of the codedword for mpm_index.

If a MPM list contains both planar and DC mode, then a MPM list excluding planar and DC mode from the constructed MPM list is used by multiple reference line coding tool. In this case, a 4-MPM list is used by multiple reference line's intra prediction mode coding. In one example, a MPM list is {Planar (a value of 0), DC(a value of 1), VER(a value of 50), HOR(a value of 18), VER-4 (a value of 46), VER+4 (a value of 54)} after MPM list construction, then a 4-MPM list of {VER, HOR, VER-4, VER+4} is used by the intra prediction mode coding when multiple reference line is enable.

If a MPM list contains planar and but no DC mode, then a MPM list excluding planar from the constructed MPM list is used by multiple reference line coding tool. In this case, a 5-MPM list is used by multiple reference line's intra prediction mode coding. In one example, a MPM list is {Planar, VER, HOR, 43, 60, 3} after MPM list construction, then a 5-MPM list of {VER, HOR, 43, 60, 3} is used by the intra prediction mode coding when multiple reference line is enable.

In another example, syntax modification for a MPM list without DC mode might be needed. If a MPM list contains no DC, probably DC mode will be used very frequently in the non-MPM branch (i.e. the mpm flag is false). In this case, a new syntax, DC_mode is introduced. DC_mode is equal to 1 indicates the intra prediction mode of current block is DC mode. DC_mode is equal to 0 indicates the intra prediction mode of current block is not DC mode.
Namely, the old syntax of non-MPM is changed from

If (mpm_flag)
…
else
intra_mode (using TB)

to

If (mpm_flag)

...

else

DC_mode

If (DC_mode==0)

intra_mode (using TB)

The syntax DC_mode can be coded with Context Adaptive Entropy Coder (CABAC). The DC_mode might be coded according to various number of probability models (in other words context) by CABAC. With the introduction of DC_Mode, the maximum value for truncated binary is 60, i.e. 67 (number of total modes) – 6 (MPM) -1 (DC). While without the introduction of DC_mode, the maximum value for the truncated binary is 61, i.e, 67 (number of total modes) – 6 (MPM).

In particular, the following methods of prediction coding of a current block implemented by a decoding device or encoding device are provided herein. The decoding device may be video decoder 30 of Figure 1A, or decoder 30 of Figure 3. The encoding device may be video encoder 20 of Figure 1A, or encoder 20 of Figure 2.

According to an embodiment 1200 (see Figure 12A), a method of intra prediction of a current block implemented by a decoding device includes the following steps. Step 1201, the device obtains a value of a first indication information of a current block, the value of the first indication information indicates whether the intra prediction mode of the current block is comprised in a set of most probable modes. The set of most probable modes comprises 5 candidate intra prediction modes and Planar mode. The first indication information is indicated by a first flag, for example, intra_luma_mpm_flag. As described above, when the value of intra_luma_mpm_flag equals to 1, the intra prediction mode of the current block is comprised in the set of most probable modes.

Step 1202, the device obtains a value of a reference index line of the current block. The value of the reference index line of the current block may be indicated by intra_luma_ref_idx. When the value of the reference index line of the current block equals to 0, i.e., intra_luma_ref_idx[x0][y0] = = 0, the closest neighboring reference line to the current block is used. There is no sequence between step 1201 and 1202.

Step 1203, the device determines whether the value of the first indication information indicates the intra prediction mode of the current block is comprised in the set of most probable modes.

Step 1204, the device determines whether the value of the reference index line indicates the closest neighboring reference line to the current block is used. There is no sequence between step 1203 and 1204.

Step 1205, the device obtains a value of a second indication information of the current block when the value of the first indication information indicates the intra prediction mode of the current block is comprised in the set of most probable modes, and the value of the reference index line indicates the closest neighboring reference line to the current block is used, the value of the second indication information of the current block indicating whether the intra prediction mode of the current block is Planar mode or not. The value of the second indication information is context adaptive binary arithmetic coding (CABAC) context decoded. The second indication information may be indicated by a second flag, for example, intra_luma_not_planar_flag. The device parses the second flag using context based on whether the current block is an intra block with intra sub-partition (ISP) coding mode enabled or not.

When MRL coding is not enabled, i.e., when the value of the reference index line indicates the closest neighboring reference line to the current block is used, the devices parse a bitstream to obtain the value of the second indication information.

It is found the planar mode is not efficient when MRL is enabled (i.e. reference line index is not equal to 0) in the conventional method. In the embodiment of the invention, when MRL coding is enabled, the value of the second indication information is obtained by a default value. In this situation, Planar mode is excluded and its corresponding flag (i.e. intra_luma_not_planar_flag) is not parsed from the bitstream, but is inferred as a default value (e.g. a value of 1). In this case, a bit is saved to indicate the value of the planar flag (i.e, intra_luma_not_planar) when MRL is enabled).

Detailed information of this embodiment 1200 is shown in the above-mentioned embodiments.

According to an embodiment 1210 (see Figure 12B), a method of intra prediction of a current block implemented by a decoding device includes the following steps.

Step 1211, the device obtains a value of a reference index line of the current block. The value of the reference index line of the current block may be indicated by intra_luma_ref_idx.

When the value of the reference index line of the current block equals to 0, i.e., intra_luma_ref_idx[x0][y0] = = 0, the closest neighboring reference line to the current block is used.

Step 1212, the device determines whether the value of the reference index line indicates the closest neighboring reference line to the current block is used.

Step 1213, when the value of the reference index line indicates the closest neighboring reference line to the current block is used, the device obtains a value of a first indication information of a current block, the value of the first indication information indicates whether the intra prediction mode of the current block is comprised in a set of most probable modes, The set of most probable modes comprises 5 candidate intra prediction modes and Planar mode. The first indication information is indicated by a first flag, for example, intra_luma_mpm_flag. As described above, when the value of intra_luma_mpm_flag equals to 1, the intra prediction mode of the current block is comprised in the set of most probable modes.

Step 1214, the device determines whether the value of the first indication information indicates the intra prediction mode of the current block is comprised in the set of most probable modes.

Step 1215, the device obtains a value of a second indication information of the current block when the value of the first indication information indicates the intra prediction mode of the current block is comprised in the set of most probable modes, and the value of the reference index line indicates the closest neighboring reference line to the current block is used, the value of the second indication information of the current block indicating whether the intra prediction mode of the current block is Planar mode or not. The value of the second indication information is context adaptive binary arithmetic coding (CABAC) context decoded. The second indication information may be indicated by a second flag, for example, intra_luma_not_planar_flag. The device parses the second flag using context based on whether the current block is an intra block with intra sub-partition (ISP) coding mode enabled or not.

Detailed information of this embodiment 1210 is shown in the above-mentioned embodiments.

According to an embodiment 1300 (see Figure 13), a method of intra prediction of a current block implemented by an encoding device includes the following steps. Step 1301, the device determines whether an intra prediction mode of a current block is comprised in a set of most probable modes. The set of most probable modes comprises 5 candidate intra prediction modes and Planar mode. Step 1302, the device determines whether the closest neighboring reference line to the current block is used for intra prediction. There is no sequence between 1301 and 1302.

Step 1303, the device encodes a bitstream, wherein the bitstream includes information to indicate whether the intra prediction mode of the current block is Planar mode when the intra prediction mode of the current block is comprised in the set of most probable modes, and the closest neighboring reference line to the current block is used for intra prediction, and send the encoded bitstream to a decoding device.

The bitstream includes a first flag indicating whether the intra prediction mode of the current block is comprised in the set of most probable modes, and a second flag indicating whether the intra prediction mode of the current block is Planar mode. As described above, as an example, the first flag is intra_luma_mpm_flag. When the value of intra_luma_mpm_flag equals to 1, the intra prediction mode of the current block is comprised in the set of most probable modes.

The value of the second indication information is context adaptive binary arithmetic coding (CABAC) context decoded. The second indication information may be indicated by a second flag, for example, intra_luma_not_planar_flag. The device parses the second flag using context based on whether the current block is an intra block with intra sub-partition (ISP) coding mode enabled or not.

The bitstream may also include an indication to indicate the value of the reference index line of the current block. The value of the reference index line of the current block is indicated by intra_luma_ref_idx, for example. When the value of the reference index line of the current block equals to 0, i.e., intra_luma_ref_idx[x0][y0] = = 0, the closest neighboring reference line to the current block is used.

Detailed information of this embodiment 1300 is shown in the above-mentioned embodiments.

FIG. 14 illustrates embodiments of a decoding device 1400. The decoding device 1400 may be video decoder 30 of Figure 1A, or decoder 30 of Figure 3. The decoding device 1400 can be used to implement he embodiment 1200 and the other embodiments described above.

The device includes a first obtaining unit 1401 and a second obtaining unit 1402. The first obtaining unit 1401, configured to obtain a value of a reference index line of a current block, and a value of a first indication information of the current block, the value of the first indication information indicating whether an intra prediction mode of the current block is comprised in a set of most probable modes. The set of most probable modes comprises 5 candidate intra prediction modes and Planar mode. The second obtaining unit 1402, configure to obtain a value of a second indication information of the current block when the value of the first indication information indicates the intra prediction mode of the current block is

comprised in the set of most probable modes, and when the value of the reference index line indicates the closest neighboring reference line to the current block is used, wherein the value of the second indication information of the current block indicates whether the intra prediction mode of the current block is Planar mode.

The device 1400 further includes a parsing unit 1403 (not shown in Fig. 14). The parsing unit 1403 configured to parse the second flag using context based on whether the current block is an intra block with intra sub-partition (ISP) coding mode enabled or not.

FIG. 15 illustrates embodiments of an encoding device 1500 for intra prediction. The encoding device 1500 may be video encoder 20 of Figure 1A, or encoder 20 of Figure 2. The encoding device 1500 can be used to implement the embodiment 1300 and the other embodiments described above.

The device 1500 includes a determining unit 1501 and an encoding unit 1502. The determining unit 1501, configured to determine whether an intra prediction mode of a current block is comprised in a set of most probable modes, and to determine whether the closest neighboring reference line to the current block is used for intra prediction. The set of most probable modes comprises 5 candidate intra prediction modes and Planar mode. The encoding unit 1502, configured to encode a bitstream, wherein the bitstream includes information to indicate whether the intra prediction mode of the current block is Planar mode when the intra prediction mode of the current block is comprised in the set of most probable modes, and the closest neighboring reference line to the current block is used for intra prediction.

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. 16 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. 17 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.

Mathematical Operators

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

Arithmetic operators

The following arithmetic operators are defined as follows:

+	Addition
−	Subtraction (as a two-argument operator) or negation (as a unary prefix operator)
*	Multiplication, including matrix multiplication
x^y	Exponentiation. Specifies x to the power of y. In other contexts, such notation is used for superscripting not intended for interpretation as exponentiation.
/	Integer division with truncation of the result toward zero. For example, 7 / 4 and −7 / −4 are truncated to 1 and −7 / 4 and 7 / −4 are truncated to −1.
÷	Used to denote division in mathematical equations where no truncation or rounding is intended.
$\frac{x}{y}$	Used to denote division in mathematical equations where no truncation or rounding is intended.
$\sum_{i=x}^{y} f(i)$	The summation of f(i) with i taking all integer values from x up to and including y.
x % y	Modulus. Remainder of x divided by y, defined only for integers x and y with x >= 0 and y > 0.

Logical operators

The following logical operators are defined as follows:

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

Relational operators

The following relational operators are defined as follows:

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

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

Bit-wise operators

The following bit-wise operators are defined as follows:

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

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

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

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

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

Assignment operators
The following arithmetic operators are defined as follows:

= Assignment operator

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

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

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

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

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

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

Mathematical functions
The following mathematical functions are defined:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

The table 14 below specifies the precedence of operations from highest to lowest; a higher position in the table 14 indicates a higher precedence.

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

Table 14: Operation precedence from highest (at top of table) to lowest (at bottom of table)

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

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

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

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

may be described in the following manner:

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

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

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

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

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

may be described in the following manner:

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

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

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

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

may be described in the following manner:

When condition 0, statement 0

When condition 1, statement 1

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 limiting, such computer-readable storage media can comprise RAM, ROM, EEPROM, CD-ROM or other optical disk storage, magnetic disk storage, or other magnetic storage devices, flash memory, or any other medium that can be used to store desired program code in the form of instructions or data structures and that can be accessed

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

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

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

CLAIMS

1. An encoding device for implementing intra prediction, the device 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 device to:

determine whether a closest neighboring reference line to a current block is used for intra prediction;

determine an intra prediction mode of the current block is not Planar mode when the closest neighboring reference line to the current block is not used for intra prediction; and

encode a bitstream, wherein the bitstream includes indication information to indicate the closest neighboring reference line to the current block is not used for the intra prediction.

2. The encoding device of claim 1, wherein the programming, when executed by the processors, configures the device to:

set a flag intra_luma_not_planar_flag to a default value when the closest neighboring reference line to the current block is not used for the intra prediction, wherein the intra_luma_not_planar_flag is used to indicate the intra prediction mode of the current block is not Planar mode.

3. The encoding device of claim 1 or 2, wherein the bitstream further includes an index indicating which intra prediction mode in a set of most probable modes is used for the intra prediction mode of the current block when the intra prediction mode of the current block is not Planar mode.

4. The encoding device of claim 3, wherein the index is intra_luma_mpm_idx[x0][y0].

5. The encoding device of any one of claims 1-4, wherein the programming, when executed by the processors, configures the device to:

set a flag intra_luma_mpm_flag to a default value when the closest neighboring reference line to the current block is not used for the intra prediction, wherein the intra_luma_mpm_flag is used to indicate the intra prediction mode of the current block is included in the set of most probable modes.

6. The encoding device of any one of claims 1-5, wherein when an intra prediction mode variable candIntraPredModeA indicating a left neighbor block and an intra prediction mode variable candIntraPredModeB indicating an above neighbor block are both non-angular modes, the set of most probable modes comprise:

candModeList[0] = INTRA_DC

candModeList[1] = INTRA_ANGULAR50

candModeList[2] = INTRA_ANGULAR18

candModeList[3] = INTRA_ANGULAR46

candModeList[4] = INTRA_ANGULAR54.

7. The encoding device of claim 6, wherein the intra prediction mode variable candIntraPredModeX (X can be replaced by A or B) is set equal to INTRA_PLANAR when one or more of the following conditions are true: a variable availableX is equal to FALSE; CuPredMode[xNbX][yNbX] is not equal to MODE_INTRA (or intra prediction mode); or X is equal to B and yCb − 1 is less than ((yCb >> CtbLog2SizeY) << CtbLog2SizeY),

wherein X being replaced by either neighbor block A or B, a luma location (xCb , yCb) specifying a top-left sample of a current luma coding block relative to a top-left luma sample of a current picture, the location (xNbA, yNbA) of the left neighbor A is set equal to (xCb − 1, yCb + cbHeight − 1), the location (xNbB, yNbB) of the above neighbor B is set equal to (xCb + cbWidth − 1, yCb − 1), a variable cbWidth specifying a width of a current coding block in luma samples, a variable cbHeight specifying the height of the current coding block in luma samples, CuPredMode[xNbA][yNbA] and CuPredMode[xNbB][yNbB] represents prediction mode of the left and above neighboring block A and B, respectively, and yCb − 1 is less than ((yCb >> CtbLog2SizeY) << CtbLog2SizeY).

8. The encoding device of any one of claims 1-7, wherein when a value of a reference index line of the current block is not equal to 0, the closest neighboring reference line to the current block is not used for the intra prediction.

9. A method of intra prediction implemented by an encoding device, comprising:

determining whether a closest neighboring reference line to a current block is used for intra prediction;

determining an intra prediction mode of the current block is not Planar mode when the closest neighboring reference line to the current block is not used for intra prediction; and

encoding a bitstream, wherein the bitstream includes indication information to indicate the closest neighboring reference line to the current block is not used for the intra prediction.

10. The method of claim 9, wherein the method comprises:

setting a flag intra_luma_not_planar_flag to a default value when the closest neighboring reference line to the current block is not used for the intra prediction, wherein the intra_luma_not_planar_flag is used to indicate the intra prediction mode of the current block is not Planar mode.

11. The method of claim 9 or 10, wherein the bitstream further includes an index indicating which intra prediction mode in a set of most probable modes is used for the intra prediction mode of the current block when the intra prediction mode of the current block is not Planar mode.

12. The method of claim 11, wherein the index is intra_luma_mpm_idx[x0][y0].

13. The method of any one of claims 9-12, wherein the method comprises:
setting a flag intra_luma_mpm_flag to a default value when the closest neighboring reference line to the current block is not used for the intra prediction, wherein the intra_luma_mpm_flag is used to indicate the intra prediction mode of the current block is included in the set of most probable modes.

14. A bitstream that is generated by a video encoding method, the bitstream comprises:
indication information indicating whether a closest neighboring reference line to a current block is not used for the intra prediction, wherein an intra prediction mode of the current block is not Planar mode when the closest neighboring reference line to the current block is not used for intra prediction; and
an index indicating which intra prediction mode in a set of most probable modes is used for the intra prediction mode of the current block when the intra prediction mode of the current block is not Planar mode.

15. An encoder (20) comprising processing circuitry for carrying out the method according to any one of claims 9 to 13.

16. An encoder, comprising:
one or more processors; and
a non-transitory computer-readable storage medium coupled to the one or more processors and storing programming for execution by the one or more processors to cause the one or more processors to carry out the method according to any one of claims 9 to 13.

17. A computer program product comprising a program code for performing the method according to any one of claims 9 to 13.

18. A non-transitory computer readable storage medium storing computer executable instructions, when the instructions are executed by a processor cause the processor to perform the method of any one of claims 9 to 13.

19. An apparatus for storing a bitstream, comprising one or more storage mediums and a receiver, wherein the receiver is configured to receive one or more bitstreams generated by the method according to any one of claims 9 to 13;
wherein the one or more storage mediums are configured to store the one or more bitstreams.

20. A device for transmitting a bitstream, wherein the device comprises:
at least one storage medium, configured to store at least one bitstream generated by the method according to any one of claims 9 to 13;

at least one processor, configured to obtain one or more bitstreams from one of the at least one storage medium, and transmit the one or more bitstreams to another device.