(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
04 March 2021 (04.03.2021)

WIPO | PCT

(10) International Publication Number
WO 2021/037053 A1

(51) International Patent Classification:
H04N 19/593 (2014.01) *H04N 19/119* (2014.01)
H04N 19/11 (2014.01) *H04N 19/122* (2014.01)
H04N 19/176 (2014.01)

(21) International Application Number:
PCT/CN2020/111338

(22) International Filing Date:
26 August 2020 (26.08.2020)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
PCT/EP2019/072797
27 August 2019 (27.08.2019) EP

(71) Applicant: **HUAWEI TECHNOLOGIES CO., LTD.** [CN/CN]; Huawei Administration Building, Bantian, Longgang District, Shenzhen, Guangdong 518129 (CN).

(72) Inventors: **GAO, Han**; c/o Huawei Technologies Duesseldorf GmbH, Riesstr. 25, 80992 Munich (DE). **ESENLIK, Semih**; c/o Huawei Technologies Duesseldorf GmbH, Riesstr. 25, 80992 Munich (DE). **ALSHINA, Elena Alexandrovna**; c/o Huawei Technologies Duesseldorf GmbH, Riesstr. 25, 80992 Munich (DE). **WANG, Biao**; c/o Huawei Technologies Duesseldorf GmbH, Riesstr. 25, 80992 Munich (DE). **KOTRA, Anand Meher**; c/o Huawei Technologies Duesseldorf GmbH, Riesstr. 25, 80992 Munich (DE).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, IT, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW,

(54) Title: AN ENCODER, A DECODER AND CORRESPONDING METHODS OF CABAC CODING FOR THE INDICES OF GEOMETRIC PARTITION FLAG

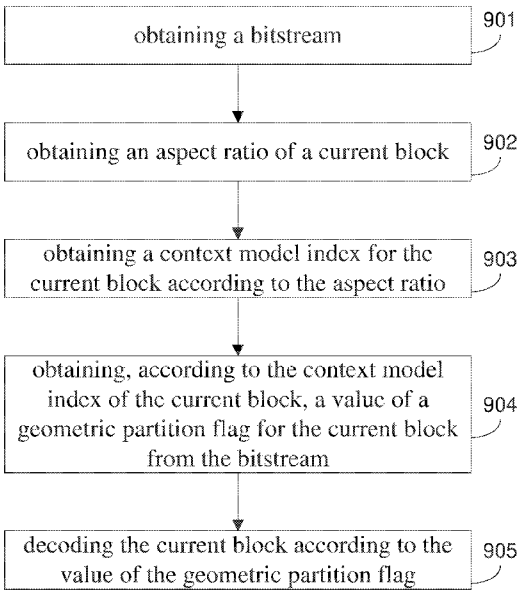

FIG. 9

(57) Abstract: The present disclosure discloses a method of coding implemented by a decoding device, comprising: obtaining a bitstream; obtaining an aspect ratio of a current block; obtaining a context model index for the current block according to the aspect ratio; obtaining, according to the context model index of the current block, a value of a geometric partition flag for the current block from the bitstream, wherein the geometric partition flag of the current block indicates whether a geometric partition mode is used or not for the current block; and decoding the current block according to the value of the geometric partition flag.

WO 2021/037053 A1

[Continued on next page]

SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report (Art. 21(3))*

AN ENCODER, A DECODER AND CORRESPONDING METHODS OF CABAC CODING FOR THE INDICES OF GEOMETRIC PARTITION FLAG

TECHNICAL FIELD

Embodiments of the present disclosure generally relate to the field of picture processing and more particularly to CABAC coding process.

BACKGROUND

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

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

SUMMARY

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

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

A first aspect provides a method of coding implemented by a decoding device, comprising: obtaining a bitstream; obtaining an aspect ratio of a current block; obtaining a context model index for the current block according to the aspect ratio; obtaining, according to the context model index of the current block, a value of a geometric partition flag for the current block from the bitstream, wherein the geometric partition flag of the current block indicates whether a geometric partition mode is used or not for the current block; and decoding the current block according to the value of the geometric partition flag.

Context models can be used to decode the geometric partition flag from the binary bitstream. Since the possibility (probability) of geometric partition occurrence may depend on the aspect ratio of the current block, it is advantageous to derive a context model index based on the aspect ratio. The context model index is used to obtain the geometric partition flag from the bitstream. The geometric partition mode flag is to indicate whether geometric partition mode is used or not for the current block. Each block contains one geometric mode flag with value 0 or value 1 when the geometric partition application condition is satisfied.

With reference to the first aspect, in a first implementation manner, the aspect ratio is the ratio of the width and the height of the current block.

With reference to the first aspect or the first implementation manner of the first aspect, in a second implementation manner, the aspect ratio is obtained according to the following equation: Ratio = 1 << abs(log2(width) – log2(height)); wherein height and width in the equation are the height and width of the current block, abs() is an absolute value operator, log2() it a logarithm scaling based on 2, and << is a left shift operation.

With reference to the first aspect or any implementation manners of the first aspect, in a third implementation manner, the obtaining a context model index for the current block according to the aspect ratio comprises: if the aspect ratio is greater than a pre-defined threshold, obtaining a context model index 3 for the current block. This provides for a specific context model index when the aspect ratio of the current block is large.

With reference to the first aspect or any implementation manners of the first aspect, in a fourth implementation manner, the obtaining a context model index for the current block according to the aspect ratio comprises: if the aspect ratio is equal to or less than a pre-defined threshold, obtaining the context model index for the current block according to at least one pieces of information of a triangle partition mode and a geometric partition mode of neighboring blocks adjacent to the current block, wherein the neighboring blocks comprises a left neighboring block and a above neighboring block.

With reference to the first aspect or the third or fourth implementation manners of the first aspect, in a fifth implementation manner, the pre-defined threshold is 2^n, and n is a positive integer.

With reference to the first aspect or the third to fifth implementation manners of the first aspect, in a sixth implementation manner, the pre-defined threshold is 4.

A second aspect provides a method of coding implemented by a decoding device, comprising: obtaining a bitstream; obtaining a context model index for a current block according to at least one pieces of information of a triangle partition mode and a geometric partition mode of neighboring blocks adjacent to the current block, wherein the neighboring blocks comprises a left neighboring block and a above neighboring block; obtaining, according to the context model index of the current block, a value of a geometric partition flag for the current block from the bitstream, wherein the geometric partition flag of the current block indicates whether a geometric partition mode is used or not for the current block; and decoding the current block according to the value of the geometric partition flag. The geometric partition mode is an extension of the triangular partition mode. The second aspect provides the advantage that the context model index can be derived from information of a triangle partition mode or a geometric partition mode of neighboring blocks adjacent to the current block. Based on the context model index, a value of a geometric partition flag for the current block can be obtained from the bitstream.

With reference to the fourth to sixth implementation manners of the first aspect, in a seventh implementation manner of the first aspect, or the second aspect, in a first implementation manner of the second aspect, the obtaining a context model index for the current block according to at least one pieces of information of a triangle partition mode and a geometric partition mode of neighboring blocks adjacent to the current block, comprises: obtaining a context model index 0 for the current block when none of the left neighboring block and the above neighboring block are using geometric partition mode or triangle partition mode.

With reference to the fourth to sixth implementation manners of the first aspect, in an eighth implementation manner of the first aspect, or the second aspect, in a second implementation manner of the second aspect, the obtaining a context model index for the current block according to at least one pieces of information of a triangle partition mode and a geometric partition mode of neighboring blocks adjacent to the current block, comprises: obtaining a context model index 1 for the current block when one of the left neighboring block and the above neighboring block is using a geometric partition mode or a triangle partition mode.

With reference to the fourth to sixth implementation manners of the first aspect, in a ninth implementation manner of the first aspect, or the second aspect, in a third implementation manner of the second aspect, the obtaining a context model index for the current block according to at least one pieces of information of a triangle partition mode and a geometric partition mode of neighboring blocks adjacent to the current block, comprises: obtaining a context model index 2 for the current block when both of the left neighboring block and the above neighboring block is using a geometric partition mode or a triangle partition mode.

With reference to the second aspect or any implementation manners of the second aspect, in a fourth implementation manner of the second aspect, further comprising: obtaining an aspect ratio of the current block, wherein the aspect ratio is the ratio of the width and the height of current block; wherein the obtaining a context model index for the current block according to at least one pieces of information of a triangle partition mode and a geometric partition mode of neighboring blocks adjacent to the current block, comprises: obtaining the context model index for the current block according to the aspect ratio and at least one pieces of information of a triangle partition mode and a geometric partition mode of neighboring blocks adjacent to the current block.

With reference to the fourth implementation manner of the second aspect, in a fifth implementation manner of the second aspect, wherein the obtaining the context model index for the current block according to the aspect ratio and at least one of a triangle partition mode and a geometric partition mode of neighboring blocks adjacent to the current block, comprises: obtaining a context model index 3 for the current block when the aspect ratio of the current block is larger than a pre-defined threshold.

With reference to the fifth implementation manner of the second aspect, in a sixth implementation manner of the second aspect, wherein the pre-defined threshold is 4.

With reference to the fifth or sixth implementation manner of the second aspect, in a seventh implementation manner of the second aspect, wherein the aspect ratio of the current block is obtained by following equation: Ratio = 1 << abs(log2(width) – log2(height)); wherein height and width in the equation are the height and width of the current block, abs() is an absolute value operator, log2() it a logarithm scaling based on 2, and << is a left shift operation.

With reference to the fourth to sixth implementation manners of the first aspect, in a tenth implementation manner of the first aspect, or the second aspect or any implementation manners of the second aspect, in an eighth implementation manner of the second aspect, wherein the piece of information of the geometric partition mode of the left neighboring

block indicates whether a geometric partition mode is used for the left neighboring block or not, and the piece of information of the geometric partition mode of the above neighboring block indicates whether a geometric partition mode is used for the above neighboring block or not.

With reference to the tenth implementation manner of the first aspect, in an eleventh implementation manner of the first aspect, or the eighth implementation manner of the second aspect, in a ninth implementation manner of the second aspect, wherein whether the geometric partition mode is used for the left neighboring block or not is determined according to a value of a geometric partition flag of the left neighboring block, or whether the geometric partition mode is used for the above neighboring block or not is determined according to a value of a geometric partition flag of the above neighboring block. In this implementation manner, the piece of information of the left neighboring block is the geometric partition flag of the left neighboring block, and the piece of information of the above neighboring block is the geometric partition flag of the above neighboring block.

With reference to the tenth implementation manner of the first aspect, in a twelfth implementation manner of the first aspect, or the ninth implementation manner of the second aspect, in a tenth implementation manner of the second aspect, wherein whether the geometric partition mode is used for the left neighboring block or not is determined according to whether the left neighboring block is allowed to use a geometric partition mode or not, or whether the geometric partition mode is used for the above neighboring block or not is determined according to whether the above neighboring block is allowed to use a geometric partition mode or not. In this aspect, the piece of information of the left neighboring block is the information indicates whether left neighboring block is allowed to use a geometric partition mode or not, e.g., may be the block size of the left neighboring block, and the piece of information of the above neighboring block is the information indicates whether above neighboring block is allowed to use a geometric partition mode or not, e.g., may be the block size of the above neighboring block.

With reference to the twelfth implementation manner of the first aspect, in a thirteenth implementation manner of the first aspect, or the tenth implementation manner of the second aspect, in an eleventh implementation manner of the second aspect, wherein if a block size of the left neighboring block is less than 8x8, the left neighboring block is not allowed to use a geometric partition mode, or if a block size of the above neighboring block is less than 8x8, the above neighboring block is not allowed to use a geometric partition mode. In this aspect,

the piece of information of the left neighboring block is the block size of the left neighboring block, and the piece of information of the above neighboring block is the block size of the above neighboring block.

With reference to the fourth to sixth implementation manners of the first aspect, in a fourteenth implementation manner of the first aspect, or the second aspect or any one of the implementation manners of the second aspect, in a twelfth implementation manner of the second aspect, wherein the piece of information of the triangle partition mode of the left neighboring block indicates whether a triangle partition mode is used for the left neighboring block or not, and the piece of information of the triangle partition mode of the above neighboring block indicates whether a triangle partition mode is used for the above neighboring block or not.

With reference to the fourteenth implementation manner of the first aspect, in a fifteenth implementation manner of the first aspect, or the twelfth implementation manner of the second aspect, in a thirteenth implementation manner of the second aspect, wherein whether the triangle partition mode is used for the left neighboring block or not is determined according to a value of a triangle partition flag of the left neighboring block, or whether the triangle partition mode is used for the above neighboring block or not is determined according to a value of a triangle partition flag of the above neighboring block.

With reference to the fourteenth implementation manner of the first aspect, in a sixteenth implementation manner of the first aspect, or the twelfth implementation manner of the second aspect, in a fourteenth implementation manner of the second aspect, wherein whether the triangle partition mode is used for the left neighboring block or not is determined according to whether the left neighboring block is allowed to use a triangle partition mode or not, or whether the triangle partition mode is used for the above neighboring block or not is determined according to whether the above neighboring block is allowed to use a triangle partition mode or not.

With reference to the sixteenth implementation manner of the first aspect, in a seventeenth implementation manner of the first aspect, or the fourteenth implementation manner of the second aspect, in a fifteenth implementation manner of the second aspect, wherein if a block size of the left neighboring block is less than 8x8, the left neighboring block is not allowed to use a triangle partition mode, or if a block size of the above neighboring block is less than 8x8, the above neighboring block is not allowed to use a triangle partition mode.

A third aspect of the present disclosure provides a decoder comprising processing circuitry

for carrying out the method according to the first aspect or any implementation manners of the first aspect or the method according to the second aspect or any implementation manners of the second aspect.

A fourth aspect of the present disclosure provides computer program product comprising a program code for performing the method according to the first aspect or any implementation manners of the first aspect or the method according to the second aspect or any implementation manners of the second aspect.

A fifth aspect of the present disclosure provides a decoder, comprising: one or more processors; and a non-transitory computer-readable storage medium coupled to the processors and storing programming for execution by the processors, wherein the programming, when executed by the processors, configures the decoder to carry out the method according to the first aspect or any implementation manners of the first aspect or the method according to the second aspect or any implementation manners of the second aspect.

A sixth aspect of the present disclosure provides a decoder, comprising: obtaining means configured for: obtaining a bitstream; obtaining an aspect ratio of a current block; obtaining a context model index for the current block according to the aspect ratio; obtaining, according to the context model index of the current block, a value of a geometric partition flag for the current block from the bitstream, wherein the geometric partition flag of the current block indicates whether a geometric partition mode is used or not for the current block; and decoding means configured for decoding the current block according to the value of the geometric partition flag.

With reference to the sixth aspect, in a first implementation manner of the sixth aspect, the aspect ratio is the ratio of the width and the height of the current block.

With reference to the sixth aspect, in a second implementation manner of the sixth aspect or the first implementation manner of the sixth aspect, the obtaining means being further configured for obtaining the aspect ratio according to the following equation: Ratio = 1 << abs(log2(width) – log2(height)); wherein height and width in the equation are the height and width of the current block, abs() is an absolute value operator, log2() it a logarithm scaling based on 2, and << is a left shift operation.

With reference to the sixth aspect or any one of the implementation manners of the sixth aspect, in a third implementation manner of the sixth aspect, the obtaining means being further configured for obtaining a context model index 3 for the current block if the aspect ratio is greater than a pre-defined threshold.

With reference to the sixth aspect or any one of the implementation manners of the sixth aspect, in a fourth implementation manner of the sixth aspect, the obtaining means being further configured for obtaining the context model index for the current block according to at least one pieces of information of a triangle partition mode and a geometric partition mode of neighboring blocks adjacent to the current block, wherein the neighboring blocks comprises a left neighboring block and a above neighboring block, if the aspect ratio is equal to or less than a pre-defined threshold.

With reference to the sixth aspect or any one of the implementation manners of the sixth aspect, in a fifth implementation manner of the sixth aspect, the pre-defined threshold is 2^n, and n is a positive integer.

With reference to the sixth aspect or any one of the implementation manners of the sixth aspect, in a sixth implementation manner of the sixth aspect, the pre-defined threshold is 4.

A seventh aspect of the present disclosure provides a decoder, comprising: obtaining means configured for: obtaining a bitstream; obtaining a context model index for a current block according to at least one pieces of information of a triangle partition mode and a geometric partition mode of neighboring blocks adjacent to the current block, wherein the neighboring blocks comprises a left neighboring block and a above neighboring block; obtaining, according to the context model index of the current block, a value of a geometric partition flag for the current block from the bitstream, wherein the geometric partition flag of the current block indicates whether a geometric partition mode is used or not for the current block; and decoding means configured for decoding the current block according to the value of the geometric partition flag.

With reference to the seventh aspect, in a first implementation manner of the seventh aspect, the obtaining means being further configured for obtaining a context model index 0 for the current block when none of the left neighboring block and the above neighboring block are using geometric partition mode or triangle partition mode.

With reference to the seventh aspect or the first implementation manner of the seventh aspect, in a second implementation manner of the seventh aspect, the obtaining means being further configured for obtaining a context model index 1 for the current block when one of the left neighboring block and the above neighboring block is using a geometric partition mode or a triangle partition mode.

With reference to the seventh aspect or any one of the implementation manners of the seventh aspect, in a third implementation manner of the seventh aspect, the obtaining means being

further configured for obtaining a context model index 2 for the current block when both of the left neighboring block and the above neighboring block is using a geometric partition mode or a triangle partition mode.

With reference to the seventh aspect or any one of the implementation manners of the seventh aspect, in a fourth implementation manner of the seventh aspect, the piece of information of the geometric partition mode of the left neighboring block indicates whether a geometric partition mode is used for the left neighboring block or not, and the piece of information of the geometric partition mode of the above neighboring block indicates whether a geometric partition mode is used for the above neighboring block or not.

With reference to the fourth implementation manner of the seventh aspect, in a fifth implementation manner of the seventh aspect, whether the geometric partition mode is used for the left neighboring block or not is determined according to a value of a geometric partition flag of the left neighboring block, or whether the geometric partition mode is used for the above neighboring block or not is determined according to a value of a geometric partition flag of the above neighboring block.

With reference to the fourth implementation manner of the seventh aspect, in a sixth implementation manner of the seventh aspect, whether the geometric partition mode is used for the left neighboring block or not is determined according to whether the left neighboring block is allowed to use a geometric partition mode or not, or whether the geometric partition mode is used for the above neighboring block or not is determined according to whether the above neighboring block is allowed to use a geometric partition mode or not.

With reference to the sixth implementation manner of the seventh aspect, in a seventh implementation manner of the seventh aspect, if a block size of the left neighboring block is less than 8x8, the left neighboring block is not allowed to use a geometric partition mode, or if a block size of the above neighboring block is less than 8x8, the above neighboring block is not allowed to use a geometric partition mode.

With reference to the seventh aspect or any one of the implementation manners of the seventh aspect, in an eighth implementation manner of the seventh aspect, the piece of information of the triangle partition mode of the left neighboring block indicates whether a triangle partition mode is used for the left neighboring block or not, and the piece of information of the triangle partition mode of the above neighboring block indicates whether a triangle partition mode is used for the above neighboring block or not.

With reference to the eighth implementation manner of the seventh aspect, in a ninth implementation manner of the seventh aspect, whether the triangle partition mode is used for the left neighboring block or not is determined according to a value of a triangle partition flag of the left neighboring block, or whether the triangle partition mode is used for the above neighboring block or not is determined according to a value of a triangle partition flag of the above neighboring block.

With reference to the eighth implementation manner of the seventh aspect, in a tenth implementation manner of the seventh aspect, whether the triangle partition mode is used for the left neighboring block or not is determined according to whether the left neighboring block is allowed to use a triangle partition mode or not, or whether the triangle partition mode is used for the above neighboring block or not is determined according to whether the above neighboring block is allowed to use a triangle partition mode or not.

With reference to the tenth implementation manner of the seventh aspect, in an eleventh implementation manner of the seventh aspect, if a block size of the left neighboring block is less than 8x8, the left neighboring block is not allowed to use a triangle partition mode, or if a block size of the above neighboring block is less than 8x8, the above neighboring block is not allowed to use a triangle partition mode.

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 shows an example about neighbor blocks of a current block;

FIG. 7 illustrates triangle partition mode and geometric partition mode;

FIG. 8 is a block diagram illustrating an example of Context-Adaptive Binary Arithmetic Coding;

FIG. 9 is a flowchart of a method of coding implemented by a decoding device.

FIG. 10 is a flowchart of another method of coding implemented by a decoder.

FIG. 11 is a block diagram illustrating a decoder.

FIG. 12 is a block diagram illustrating another embodiment of the decoder.

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.

Context-Adaptive Binary Arithmetic Coding (CABAC).

Context-Based Adaptive Binary Arithmetic Coding (CABAC) is a form of entropy coding used in H.264/AVC and also in HEVC. Entropy coding is a lossless compression scheme that uses the statistical properties to compress data such that the number of bits used to represent the data is logarithmically proportional to the probability of the data. For instance, when compressing a string of characters, frequently used characters are each represented by a few bits, while infrequently used characters are each represented by many bits. From Shannon's information theory, when the compressed data is represented in bits {0, 1}, the optimal average code length for a character with probability p is - log2 (p).

Figure 8 is a schematic diagram of a CABAC entropy coding process. The CABAC coding operation can be described according to the following steps:

Step 1: Binarization

Binarization (corresponding to the Binarizer component in Figure 8) process converts a syntax element into a series of binary digits (b_1, b_2, …b_n), for example, using truncated unary code. The "binary" part of the context adaptive binary arithmetic coding refers to the binarization step.

Step 2: Probability model selection

For each binary digit, either Regular Arithmetic Encoding or Bypass Arithmetic Encoding coding method is selected (the selection process is represent with regular/bypass mode switch in the figure). The Bypass arithmetic encoding mode indicates that the said binary digit has a 50% chance of assuming a value of "1", and 50% chance of assuming a value of "0". In other words the probability model that is used for coding of a binary digit is equal probability model in the bypass arithmetic coding mode.

If the second branch (Regular arithmetic encoding mode) is selected, then first a probability model (context model, or probability estimation model) is selected corresponding to the said binary digit. The probability model is a number that represents the probability of observing a "1" for the said binary digit. For example the probability estimation model can be 20%, indicating that the said syntax element has a 20% chance of assuming a value of 1 and 80% probability of assuming a zero.

For each binary digit that is to be coded in the regular arithmetic encoding mode, there is at least one associated context model (probability model). Selection of the probability model is referred to as context modeling. In some cases corresponding to a binary digit there can be more than N>1 associated context models. Out of the set of N context models one of them is selected each time according to the previously coded syntax elements. This process is known as context switching. For example for coding of an intra_luma_not_planar_flag syntax element, 1 out of an N=2 context models might be selected according to the rule:

- If the current block is applied with intra sub-partition (ISP) coding mode, a first context model is selected.
- If the current block is NOT applied with intra sub-partition coding mode, a second context model is selected.

intra_luma_not_planar_flag equal to 0 specifies that the current coding block is coded in intra planar prediction mode. intra_luma_not_planar_flag equal to 1 specifies that the current coding block is coded not in intra planar prediction mode.

As Figure 8 shows, there might be multiple probability models that are stored in context memory. Each of the context models are used during the coding of a binary digit. Sometimes more than one context models are required because in each context model, the probability of a digit become 0 (or 1) might be different. Exploiting such difference can lead to less entropy to encode the digit, i.e. more compression can be achieved. For example, suppose the probability of intra_luma_not_planar_flag become 0 and 1 are 50% each in total. Furthermore, there are 50% probability the current block is applied with ISP and 50% not. However, in blocks applied with ISP, the probability for the flag intra_luma_not_planar_flag become 1 is 75%, while in blocks NOT applied with ISP, the probability for the flag intra_luma_not_planar_flag become 1 is 25%.

In such case, if only one context model is used, the entropy is 1

$$\text{Entropy} = -(0.5*\log 0.5 + 0.5*\log 0.5) = 1$$

However, if two context models are used, the entropy is 0.811

$$\text{Entropy} = 0.5*\text{Entropy}^{ISP} + 0.5*\text{Entropy}^{non\text{-}ISP}$$

$$\text{Entropy}^{ISP} = -(0.25*\log 0.25 + 0.75*\log 0.75) = 0.811$$

$$\text{Entropy}^{non\text{-}ISP} = -(0.75*\log 0.75 + 0.25*\log 0.25) = 0.811$$

For each digit using regular CABAC coding, it is addressed in CABAC using a unique context index (ctxIdx), corresponding to its first context model it associated. If more than one context is defined, specific context derivation rule is defined. For each its associated context, some initialization parameters estimating its initial probability distribution is required, including initialization parameter for I, P, and B frames, and a parameter called window size to control the speed of adaptation in CABAC. More detailed information please refer to the book “High Efficiency Video Coding (HEVC): Algorithms and Architectures”, Vivienne Sze, Madhukar Budagavi, Gary J. Sullivan, Springer, 2014.

Step 3: Arithmetic coding.

According to the selected encoding mode (bypass coding or regular arithmetic coding) and the selected probability estimation model (context model), the binary digit is coded using arithmetic coding. Arithmetic coding is a lossless entropy coding method, which is explained for example in (https://en.wikipedia.org/wiki/Arithmetic_coding). The arithmetic coding operation requires probabilities of occurrences of the symbols (probability models,

probability estimation models or context models). The probability model to be used to encode a binary digit is obtained according to step 2. The probability model is used in such a way that a low probability symbol is coded using more bits, whereas a high probability symbol is coded using fewer bits. For example if probability model is 50%, then value of 1 and 0 are coded using equal amount of bits. However if the probability modes is 20%, a 1 is coded with more bits than a 0. Using this scheme the number of total bits necessary for coding many binary digits are minimized.

Step 4: Probability model updating

The context models that are used in coding of the binary digits are updated accordingly after the coding of the binary digit is complete. For example if the Nth context model is used during the coding of a binary digit and if the context model indicated 20% possibility of observing a 1, then if said binary digit had a value of 1, probability model is increased (so that probability of observing a 1 is greater than 20%). On the other hand if the said binary digit had a value of 0, then the probability model is decreased (so that probability of observing a 1 is smaller than 20%). The “adaptive” part of the context adaptive binary arithmetic coding refers to this step. It is noted that the Step 4 (Probability model updating) is applied only for the regular arithmetic coding mode of CABAC and not for the bypass coding mode of CABAC. In the bypass coding mode the probability model that is used always indicates 50% (equal probability of observing 1 or 0). On the other hand the probability models in the regular arithmetic coding mode are updated in order to capture the statistical variance in the associated binary digit.

The decoding process of CABAC applies the 4 steps described above in a similar fashion. Also more details about CABAC can be obtained for example in (https://en.wikipedia.org/wiki/Context-adaptive_binary_arithmetic_coding)

It is noted that terms context model, probability model, probability estimation model are used synonymously throughout the invention.

Geometric partition and Geometric partition flag

In JVET-O0489, the geometric partition mode is proposed. Geometric partition mode (GEO) is a split method as extension of the current triangle prediction mode (TPM).

Triangle Partition Mode (TPM) coding was adopted to VVC draft at JVET-L meeting in Macao. TPM is allowed for uni-predicted blocks not smaller than 8×8 in order to equalize complexity with bi-predicted inter blocks coding. TPM divides a rectangular coding block into two triangular prediction blocks with diagonal or inverse diagonal direction as shown in Figure 7 on the left side. Whole block residual is encoded for TPM; subblock transform for inter-predicted CUs (SBT) is not used for TPM.

Geometric partition mode is an extension of the exsisting TPM mode. Proposed solution further extends flexibility of non-rectangular partitioning of inter-blocks utilizing Geometrical partitioning (GEO) idea and introduces specific version of sub-block transform for non-rectangular inter CUs.

For example in Figure 7 shows some examples of geometric partition mode on the right side. The dividing line may be characterized by an angle variable and a distance variable.

A geometric partition mode flag is to indicate whether geometric partition mode is used or not for the current coding block (or current block). Each coding block (or block) contains one geometric mode flag with value 0 or 1 when the geometric partition application condition is satisfied. In JVET-O0489, when coding block is larger than 8x8 luma samples, geometric partition mode can be applied.

The parsing process of geometric partition flag from a binarized bitstream is using the mentioned CABAC method. As mentioned in the CABAC background, context models are used to decode the binary bitstream to a flag. In JVET-O0489, 3 context models are used to to decode the geometric partition flag from the binary bitsteam. The contex model index (ctxIdx) are derived as following:

Contex model index 0: When none of the left and above neighboring block of current coding block are using geometric partition mode. The probability that the current block uses geometric partitioning is low. The position of left and above neighboring blocks are shown in Figure 6.

Contex model index 1: When one of the left and above neighboring block of current coding block is using geometric partition mode. The probability that the current block uses geometric partitioning is medium.

Contex model index 2: When both of the left and above neighboring block of current coding block are using geometric partition mode. The probability that the current block uses geometric partitioning is high.

As mentioned in the background of CABAC, the context models are designed based on the possibility for the occurrence of a flag. The occurrence of geometric partition flag can be different under certain conditions. The following embodiment designed new context model index derivation method for decoding the geometric partition flag.

In one embodiment:

According to the embodiment, still 3 context models are used to decode the geometric partition flag from the binary bitstream during the CABAC decoding process.

As mentioned the Geometric partition mode is extension of the existed triangle prediction mode, is reasonable to add TPM mode to the context modeling derivation.

In one example,

The context model indices of the geometric partition flag is derived as following:

Context model index 0: When none of the left and above neighboring block of current coding block (or may be called as current block) are using geometric partition mode or triangle partition mode. The position of left and above neighboring blocks are shown in Figure 6.

Context model index 1: When one of the left and above neighboring block of current coding block is using geometric partition mode or triangle partition mode.

Context model index 2: When both of the left and above neighboring block of current coding block are using geometric partition mode or triangle partition mode.

In one implementation manner, the context model indices of the geometric partition flag of the current block can be derived according to the following equation:

ctxInc = condL + condA

ctxInc is the context model index; condL indicates whether a geometric partition mode or a triangle partition mode is used for the left neighboring block, if either a geometric partition mode or a triangle partition mode is used for the left neighboring block, condL is equal to 1, otherwise, if neither the geometric partition mode nor the triangle partition mode is used for the left neighboring block, condL is equal to 0; condA indicates whether a geometric partition

mode or a triangle partition mode is used for the above neighboring block, if either a geometric partition mode or a triangle partition mode is used for the above neighboring block, condA is equal to 1, otherwise, if neither the geometric partition mode nor the triangle partition mode is used for the above neighboring block, condA is equal to 0.

In another implementation manner, the availabilities of the left neighboring block and the above neighboring block are used in the derivation of the context model indices, and the context model indices of the geometric partition flag of the current block can be derived according to the following equation:

ctxInc = (condL && availableL) + (condA && availableA)

availableL indicates whether the left neighboring block is available or not, if the left neighboring block is available, availableL is equal to 1, otherwise, if the left neighboring block is not available, availableL is equal to 0; availableA indicates whether the above neighboring block is available or not, if the left neighboring block is available, availableA is equal to 1, otherwise, if the above neighboring block is not available, availableA is equal to 0.

According to the example, whether a neighboring block is using geometric partition mode can be determined by checking whether the geometric partition flag of the neighboring block is 1.

Alternatively whether a neighboring block is using geometric partition mode can be determined by checking whether the neighboring block is allowed using geometric partition mode, in one example a block smaller than 8x8 is not allowed using geometric partition mode.

Whether a neighboring block is using triangle prediction mode can be determined by checking whether the triangle prediction flag of the neighboring block is 1.

Alternatively whether a neighboring block is using triangle prediction mode be determined by checking whether the neighboring block is allowed using triangle prediction mode, in one example a block smaller than 8x8 is not allowed using geometric partition mode.

A corresponding general method of coding implemented by a decoding device is shown in Figure 10, and comprises:

1001. Obtaining a bitstream.

1002. Obtaining a context model index for a current block according to at least one pieces of information of a triangle partition mode and a geometric partition mode of neighboring blocks adjacent to the current block, wherein the neighboring blocks comprises a left neighboring block and a above neighboring block. This information indicates whether a geometric partition mode or a triangle partition mode is used for neighboring blocks, specifically whether a geometric partition mode or a triangle partition mode is used for at least one of the left neighboring block and the above neighboring block or not, and if it is used, whether it is used for just one or for both of the neighboring blocks. The context model index is obtained according to this information.

1003. Obtaining, according to the context model index of the current block, a value of a geometric partition flag for the current block from the bitstream, wherein the geometric partition flag of the current block indicates whether a geometric partition mode is used or not for the current block.

1004. Decoding the current block according to the value of the geometric partition flag.

In a second embodiment

According to second embodiment, 4 context models are used to decode the geometric partition flag from the binary bitstream during the CABAC decoding process.

Since the possibility of geometric partition occurrence depends on the aspect ratio the current block, according this embodiment, the other 3 context model indices derivation are based on neighboring block, the 4th context model index derivation are based on the aspect ratio of the current block.

In one example:

Context model index 0: When none of the left and above neighboring block of current coding block are using geometric partition mode .The position of left and above neighboring blocks are shown in Figure 6.

Context model index 1: When one of the left and above neighboring block of current coding block is using geometric partition mode.

Context model index 2: When both of the left and above neighboring block of current coding block are using geometric partition mode.

Context model index 3: when the aspect ratio of the current coding block is larger than a

pre-defined threshold (2, 4, 8, etc for example) (not depends on neighboring blocks). In one implementation manner, the ratio may be set as 2^n, and n is a positive integer.

It should be noted that, the context model indices 0-2 are obtained when the aspect ratio of the current coding block is equal to or less than the pre-defined threshold. That is, during the decoding process, whether the aspect ratio of the current coding block is greater than the pre-defined threshold is first determined, and the context model indices is obtained based on the result of the determination.

The aspect ratio is the ratio of width and height of current block, and it can be obtained for example by following equation:

Ratio = 1 << abs(log2(width) – log2(height))

Where height and width are the height and width of the current coding block. Abs() is the absolute value operator, log2() it the logarithm scaling based on 2, and << is the left shift operation.

In one implementation manner, the context model indices of the geometric partition flag of the current block can be derived according to the following equation:

ctxInc = Ratio > 4 ? 3:(condL + condA)

ctxInc is the context model index; condL indicates whether a geometric partition mode is used for the left neighboring block, if a geometric partition mode is used for the left neighboring block, condL is equal to 1, otherwise, if the geometric partition mode is used for the left neighboring block, condL is equal to 0; condA indicates whether a geometric partition mode is used for the above neighboring block, if either a geometric partition mode is used for the above neighboring block, condA is equal to 1, otherwise, if the geometric partition mode is used for the above neighboring block, condA is equal to 0.

In another implementation manner, the availabilities of the left neighboring block and the above neighboring block are used in the derivation of the context model indices, and the context model indices of the geometric partition flag of the current block can be derived according to the following equation:

ctxInc = Ratio > 4 ? 3: (condL && availableL) + (condA && availableA)

availableL indicates whether the left neighboring block is available or not, if the left neighboring block is available, availableL is equal to 1, otherwise, if the left neighboring block is not available, availableL is equal to 0; availableA indicates whether the above

neighboring block is available or not, if the left neighboring block is available, availableA is equal to 1, otherwise, if the above neighboring block is not available, availableA is equal to 0.

In a third embodiment

The context model indices derivation method of geometric partition mode in the first and second embodiments can be combined.

In other words, 4 context models are used to decode the geometric partition flag from the binary bitstream during the CABAC decoding process.

The context model indices are derived as:

Context model index 0: When none of the left and above neighboring block of current coding block are using geometric partition mode or triangle partition mode. The position of left and above neighboring blocks are shown in Figure 6.

Context model index 1: When one of the left and above neighboring block of current coding block is using geometric partition mode or triangle partition mode

Context model index 2: When both of the left and above neighboring block of current coding block are using geometric partition mode or triangle partition mode.

Context model index 3: when the aspect ratio of the current coding block is larger than a pre-defined threshold (4 for example) (not depends on neighboring blocks).

It should be noted that, the context model indices 0-2 are obtained when the aspect ratio of the current coding block is equal to or less than the pre-defined threshold. That is, during the decoding process, whether the aspect ratio of the current coding block is greater than the pre-defined threshold is first determined, and the context model indices is obtained based on the result of the determination.

The aspect ratio is the ratio of width and height of current block, and it can be obtained for example by following equation:

Ratio = 1 << abs(log2(width) – log2(height))

Where height and width are the height and width of the current coding block. Abs() is the absolute value operator, log2() it the logarithm scaling based on 2, and << is the left shift operation. For example, if the width of the current block is 16 pixels and the height is 4 pixels, this gives Ratio = 1 << abs(4-2) such that the result for Ratio is 4.

In one implementation manner, the context model indices of the geometric partition flag of the current block can be derived according to the following equation:

ctxInc = Ratio > 4 ? 3:(condL + condA)

ctxInc is the context model index; condL indicates whether a geometric partition mode or a triangle partition mode is used for the left neighboring block, if either a geometric partition mode or a triangle partition mode is used for the left neighboring block, condL is equal to 1, otherwise, if neither the geometric partition mode nor the triangle partition mode is used for the left neighboring block, condL is equal to 0; condA indicates whether a geometric partition mode or a triangle partition mode is used for the above neighboring block, if either a geometric partition mode or a triangle partition mode is used for the above neighboring block, condA is equal to 1, otherwise, if neither the geometric partition mode nor the triangle partition mode is used for the above neighboring block, condA is equal to 0. The context model index ctxInc evaluates to be equal to 3, if the aspect ratio larger than 4 (in this specific example), independent of the values of condL and condA.

In another implementation manner, the availabilities of the left neighboring block and the above neighboring block are used in the derivation of the context model indices, and the context model indices of the geometric partition flag of the current block can be derived according to the following equation:

ctxInc = Ratio > 4 ? 3: (condL && availableL) + (condA && availableA)

availableL indicates whether the left neighboring block is available or not, if the left neighboring block is available, availableL is equal to 1, otherwise, if the left neighboring block is not available, availableL is equal to 0; availableA indicates whether the above neighboring block is available or not, if the left neighboring block is available, availableA is equal to 1, otherwise, if the above neighboring block is not available, availableA is equal to 0.

In a fourth embodiment

A method of coding implemented by a decoding device as shown in fig. 9, includes:

901. obtaining a bitstream.

902. obtaining an aspect ratio of a current block.

The aspect ratio is the ratio of the width and the height of the current block, and in a implementation manner, the aspect ratio may be obtained according to the following equation:

Ratio = 1 << abs(log2(width) – log2(height));

The height and width in the equation are the height and width of the current block, abs() is an absolute value operator, log2() it a logarithm scaling based on 2, and << is a left shift operation.

903. obtaining a context model index for the current block according to the aspect ratio.

If the aspect ratio is greater than a pre-defined threshold, obtaining a context model index 3 for the current block.

If the aspect ratio is equal to or less than a pre-defined threshold, obtaining the context model index for the current block according to at least one pieces of information of a triangle partition mode and a geometric partition mode of neighboring blocks adjacent to the current block, wherein the neighboring blocks comprises a left neighboring block and a above neighboring block.

The detail for obtaining the context model index is similar to the embodiments 2 and 3.

904. obtaining, according to the context model index of the current block, a value of a geometric partition flag for the current block from the bitstream, wherein the geometric partition flag of the current block indicates whether a geometric partition mode is used or not for the current block.

905. decoding the current block according to the value of the geometric partition flag.

Figure 11 illustrates a decoder 1100 according to the present disclosure. The decoder 1100 comprises one or more processors 1101 and a non-transitory computer-readable storage medium 1102. The non-transitory computer-readable storage medium 1102 is coupled to the one or more processors 1101 and stores programming for execution by the processors, wherein the programming, when executed by the one or more processors 1101, configures the decoder 1100 to carry out the method according to one of the aspects or implementations of the present disclosure. During startup, the one or more processors 1101 receive the programming from the non-transitory computer-readable storage medium 1102.

Figure 12 illustrates another decoder 1200 according to the present disclosure. The decoder 1200 comprises obtaining means 1201 and decoding mean 1202.

According to one aspect, the obtaining means 1201 are be configured configured for: obtaining a bitstream; obtaining an aspect ratio of a current block; obtaining a context model index for the current block according to the aspect ratio; obtaining, according to the context

model index of the current block, a value of a geometric partition flag for the current block from the bitstream, wherein the geometric partition flag of the current block indicates whether a geometric partition mode is used or not for the current block; and the decoding means 1202 are be configured for decoding the current block according to the value of the geometric partition flag.

Alternatively, according to another aspect, the obtaining means 1201 are be configured for: obtaining a bitstream; obtaining a context model index for a current block according to at least one pieces of information of a triangle partition mode and a geometric partition mode of neighboring blocks adjacent to the current block, wherein the neighboring blocks comprises a left neighboring block and a above neighboring block; obtaining, according to the context model index of the current block, a value of a geometric partition flag for the current block from the bitstream, wherein the geometric partition flag of the current block indicates whether a geometric partition mode is used or not for the current block; and the decoding means 1202 are be configured for decoding the current block according to the value of the geometric partition flag.

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:

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

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

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

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

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

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

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

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

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

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

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

Text description of logical operations

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

```
if( condition 0 )
   statement 0
else if( condition 1 )
   statement 1
...
else /* informative remark on remaining condition */
   statement n
```

may be described in the following manner:

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

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

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

the following applies" with the ending "Otherwise, ...".
In the text, a statement of logical operations as would be described mathematically in the following form:

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

may be described in the following manner:

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

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

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

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

may be described in the following manner:

When condition 0, statement 0

When condition 1, statement 1.

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. A method of coding implemented by a decoding device, comprising:
obtaining a bitstream;
obtaining an aspect ratio of a current block;
obtaining a context model index for the current block according to the aspect ratio;
obtaining, according to the context model index of the current block, a value of a geometric partition flag for the current block from the bitstream, wherein the geometric partition flag of the current block indicates whether a geometric partition mode is used or not for the current block; and
decoding the current block according to the value of the geometric partition flag.

2. The method according to claim 1, wherein the aspect ratio is the ratio of the width and the height of the current block.

3. The method according to claim 1 or 2, wherein the aspect ratio is obtained according to the following equation:
Ratio = 1 << abs(log2(width) – log2(height));
wherein height and width in the equation are the height and width of the current block, abs() is an absolute value operator, log2() it a logarithm scaling based on 2, and << is a left shift operation.

4. The method according to any one of claims 1 to 3, wherein the obtaining a context model index for the current block according to the aspect ratio, comprises:
if the aspect ratio is greater than a pre-defined threshold, obtaining a context model index 3 for the current block.

5. The method according to any one of claims 1 to 4, wherein the obtaining a context model index for the current block according to the aspect ratio, comprises:
if the aspect ratio is equal to or less than a pre-defined threshold, obtaining the context model index for the current block according to at least one pieces of information of a triangle partition mode and a geometric partition mode of neighboring blocks adjacent to the current block, wherein the neighboring blocks comprises a left neighboring block and a above neighboring block.

6. The method according to claim 4 or 5, wherein the pre-defined threshold is 2^n, and n is a positive integer.

7. The method according to any one of claims 4 to 6, wherein the pre-defined threshold is 4.

8. A method of coding implemented by a decoding device, comprising:
obtaining a bitstream;
obtaining a context model index for a current block according to at least one pieces of information of a triangle partition mode and a geometric partition mode of neighboring blocks adjacent to the current block, wherein the neighboring blocks comprises a left neighboring block and a above neighboring block;
obtaining, according to the context model index of the current block, a value of a geometric partition flag for the current block from the bitstream, wherein the geometric partition flag of the current block indicates whether a geometric partition mode is used or not for the current block; and
decoding the current block according to the value of the geometric partition flag.

9. The method of any one of claims 5 to 8, wherein obtaining the context model index for the current block according to at least one pieces of information of a triangle partition mode and a geometric partition mode of neighboring blocks adjacent to the current block, comprises:
obtaining a context model index 0 for the current block when none of the left neighboring block and the above neighboring block are using geometric partition mode or triangle partition mode.

10. The method of any one of claims 5 to 9, wherein the obtaining the context model index for the current block according to at least one pieces of information of a triangle partition mode and a geometric partition mode of neighboring blocks adjacent to the current block, comprises:
obtaining a context model index 1 for the current block when one of the left neighboring block and the above neighboring block is using a geometric partition mode or a triangle partition mode.

11. The method of any one of claims 5 to 10, wherein the obtaining a context model index for the current block according to at least one pieces of information of a triangle partition mode

and a geometric partition mode of neighboring blocks adjacent to the current block, comprises:

obtaining a context model index 2 for the current block when both of the left neighboring block and the above neighboring block is using a geometric partition mode or a triangle partition mode.

12. The method of any one of claims 5 to 11, wherein the piece of information of the geometric partition mode of the left neighboring block indicates whether a geometric partition mode is used for the left neighboring block or not, and the piece of information of the geometric partition mode of the above neighboring block indicates whether a geometric partition mode is used for the above neighboring block or not.

13. The method of claim 12, wherein whether the geometric partition mode is used for the left neighboring block or not is determined according to a value of a geometric partition flag of the left neighboring block, or whether the geometric partition mode is used for the above neighboring block or not is determined according to a value of a geometric partition flag of the above neighboring block.

14. The method of claim 12, wherein whether the geometric partition mode is used for the left neighboring block or not is determined according to whether the left neighboring block is allowed to use a geometric partition mode or not, or whether the geometric partition mode is used for the above neighboring block or not is determined according to whether the above neighboring block is allowed to use a geometric partition mode or not.

15. The method according to claim 14, wherein if a block size of the left neighboring block is less than 8x8, the left neighboring block is not allowed to use a geometric partition mode, or if a block size of the above neighboring block is less than 8x8, the above neighboring block is not allowed to use a geometric partition mode.

16. The method of any one of claims 5 to 15, wherein the piece of information of the triangle partition mode of the left neighboring block indicates whether a triangle partition mode is used for the left neighboring block or not, and the piece of information of the triangle partition mode of the above neighboring block indicates whether a triangle partition mode is used for the above neighboring block or not.

17. The method of claim 16, wherein whether the triangle partition mode is used for the left neighboring block or not is determined according to a value of a triangle partition flag of the left neighboring block, or whether the triangle partition mode is used for the above neighboring block or not is determined according to a value of a triangle partition flag of the above neighboring block.

18. The method of claim 16, wherein whether the triangle partition mode is used for the left neighboring block or not is determined according to whether the left neighboring block is allowed to use a triangle partition mode or not, or whether the triangle partition mode is used for the above neighboring block or not is determined according to whether the above neighboring block is allowed to use a triangle partition mode or not.

19. The method according to claim 18, wherein if a block size of the left neighboring block is less than 8x8, the left neighboring block is not allowed to use a triangle partition mode, or if a block size of the above neighboring block is less than 8x8, the above neighboring block is not allowed to use a triangle partition mode.

20. A decoder (30) comprising processing circuitry for carrying out the method according to any one of claims 1 to 19.

21. A computer program product comprising a program code for performing the method according to any one of claims 1 to 19.

22. A decoder (1100), comprising:
one or more processors (1101); and
a non-transitory computer-readable storage medium (1102) coupled to the one or more processors (1101) and storing programming for execution by the processors, wherein the programming, when executed by the one or more processors, configures the decoder to carry out the method according to any one of claims 1 to 19.

23. A decoder (1200) comprising:
obtaining means (1201) configured for:
obtaining a bitstream;

obtaining an aspect ratio of a current block;

obtaining a context model index for the current block according to the aspect ratio;

obtaining, according to the context model index of the current block, a value of a geometric partition flag for the current block from the bitstream, wherein the geometric partition flag of the current block indicates whether a geometric partition mode is used or not for the current block; and

decoding means (1202) configured for decoding the current block according to the value of the geometric partition flag.

24. The decoder according to claim 23, the obtaining means being further configured for obtaining the aspect ratio according to the following equation:

Ratio = 1 << abs(log2(width) – log2(height));

wherein height and width in the equation are the height and width of the current block, abs() is an absolute value operator, log2() it a logarithm scaling based on 2, and << is a left shift operation.

25. The decoder according to claim 23 or 24, the obtaining means being further configured for obtaining a context model index 3 for the current block if the aspect ratio is greater than a pre-defined threshold.

26. The decoder according to any one of claims 23 to 25, the obtaining means being further configured for obtaining the context model index for the current block according to at least one pieces of information of a triangle partition mode and a geometric partition mode of neighboring blocks adjacent to the current block, wherein the neighboring blocks comprises a left neighboring block and a above neighboring block, if the aspect ratio is equal to or less than a pre-defined threshold.

27. A decoder (1200) comprising:

obtaining means (1201) configured for:

obtaining a bitstream;

obtaining a context model index for a current block according to at least one pieces of information of a triangle partition mode and a geometric partition mode of neighboring blocks adjacent to the current block, wherein the neighboring blocks comprises a left neighboring block and a above neighboring block;

obtaining, according to the context model index of the current block, a value of a geometric partition flag for the current block from the bitstream, wherein the geometric partition flag of the current block indicates whether a geometric partition mode is used or not for the current block; and

decoding means (1202) configured for decoding the current block according to the value of the geometric partition flag.

28. The decoder according to claim 27, the obtaining means being further configured for obtaining a context model index 0 for the current block when none of the left neighboring block and the above neighboring block are using geometric partition mode or triangle partition mode.

29. The decoder according to claim 27 or 28, the obtaining means being further configured for obtaining a context model index 1 for the current block when one of the left neighboring block and the above neighboring block is using a geometric partition mode or a triangle partition mode.

30. The decoder according to any one of claims 27 to 29, the obtaining means being further configured for obtaining a context model index 2 for the current block when both of the left neighboring block and the above neighboring block is using a geometric partition mode or a triangle partition mode.

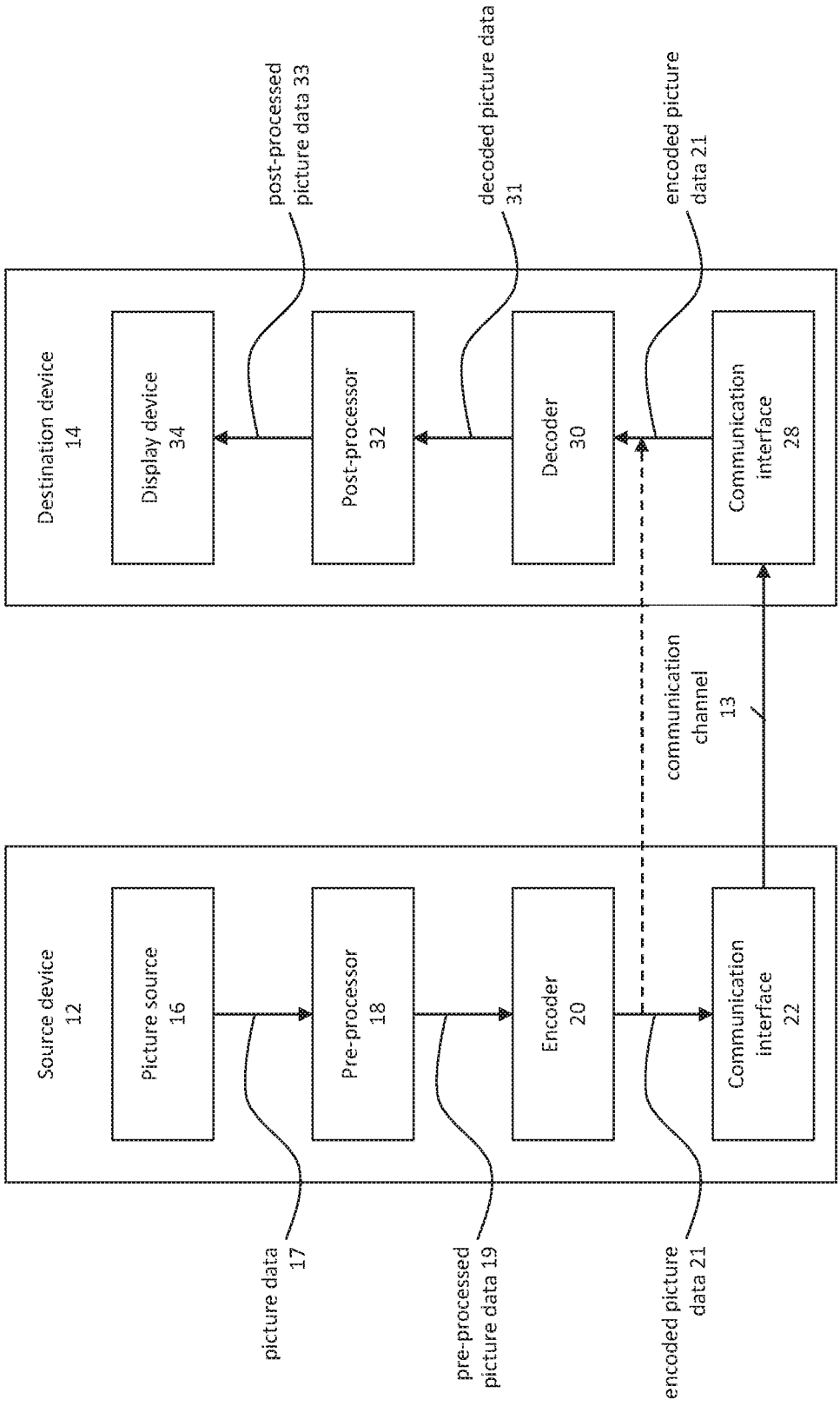
Destination device
14
Display device
34
post-processed
picture data 33
Post-processor
32
decoded picture data
31
Decoder
30
encoded picture
data 21
Communication
interface
28
communication
channel
13
Source device
12
Picture source
16
picture data
17
Pre-processor
18
pre-processed
picture data 19
Encoder
20
encoded picture
data 21
Communication
interface
22
10

Fig. 1A

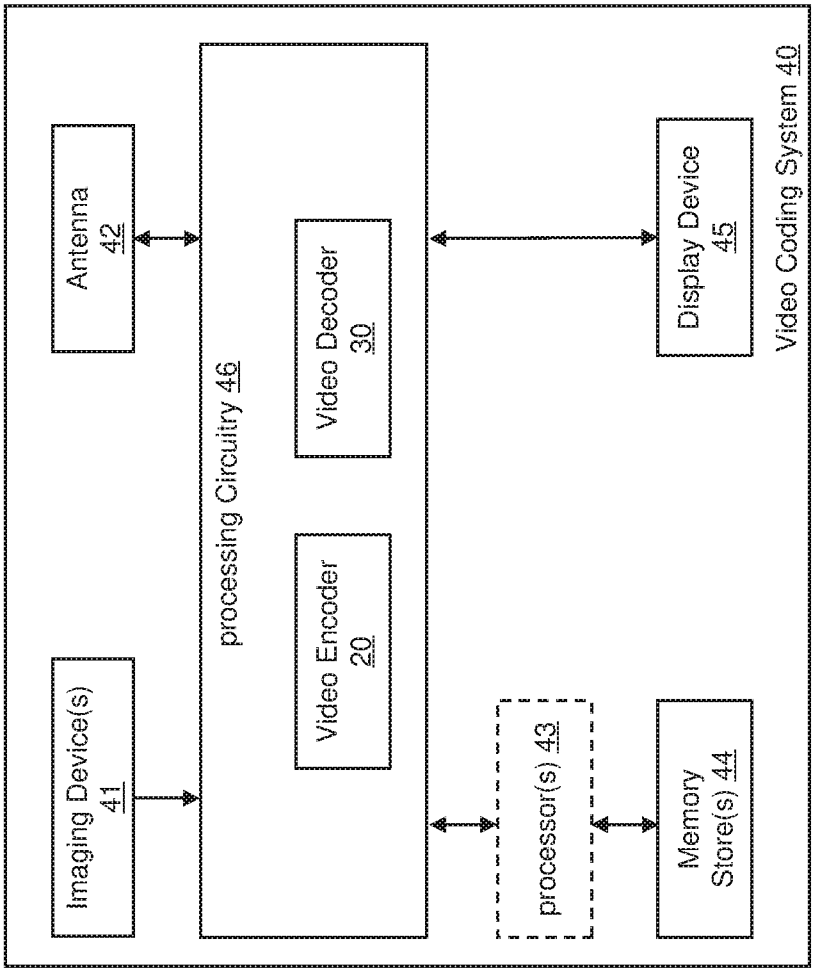
Imaging Device(s) 41
Antenna 42
processing Circuitry 46
Video Encoder 20
Video Decoder 30
processor(s) 43
Memory Store(s) 44
Display Device 45
Video Coding System 40

Fig. 1B

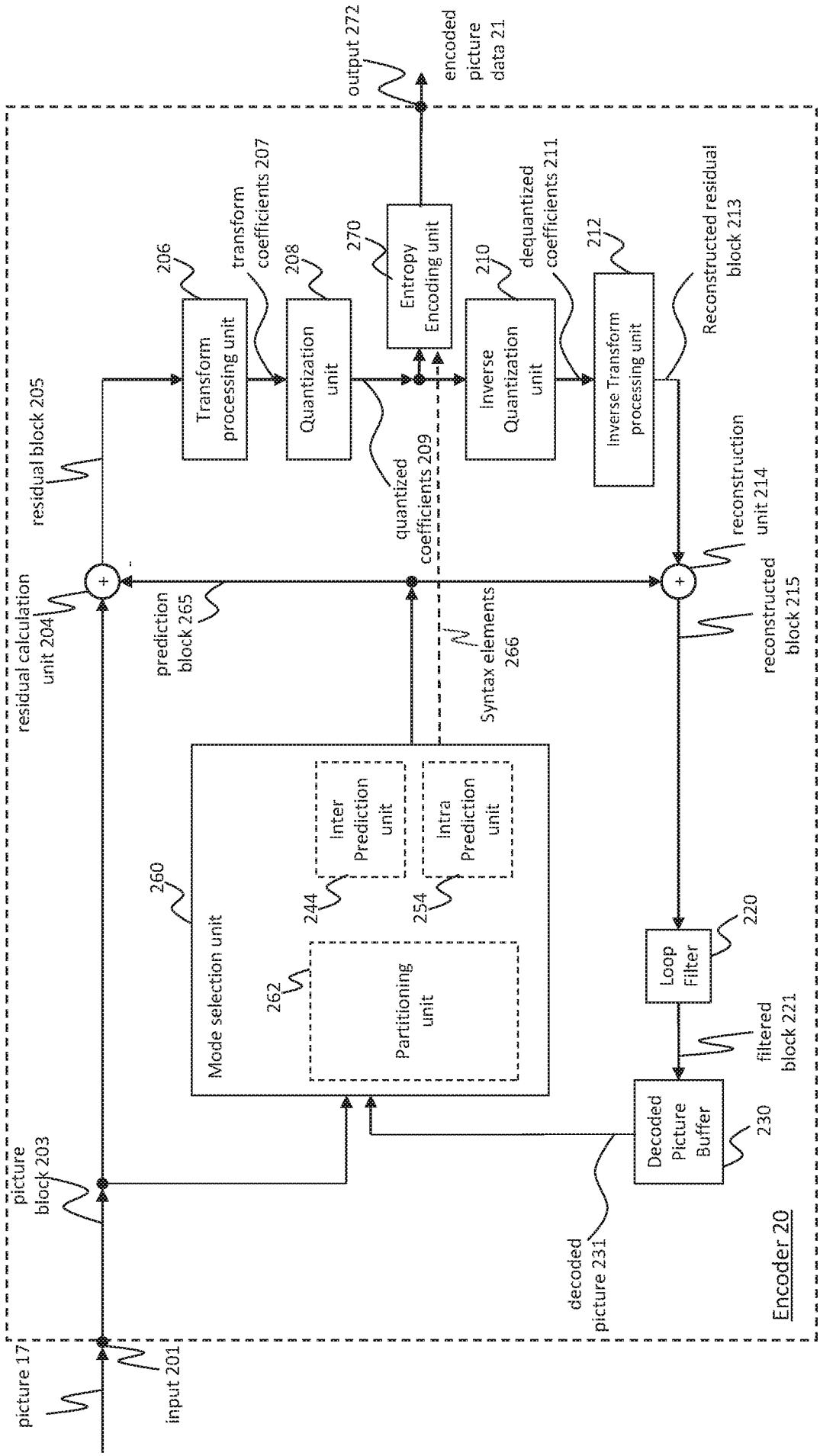
output 272
encoded picture data 21
transform coefficients 207
206
Transform processing unit
208
Quantization unit
270
Entropy Encoding unit
210
dequantized coefficients 211
212
Inverse Transform processing unit
Reconstructed residual block 213
Inverse Quantization unit
residual block 205
quantized coefficients 209
reconstruction unit 214
residual calculation unit 204
prediction block 265
Syntax elements 266
reconstructed block 215
Inter Prediction unit
Intra Prediction unit
260
Mode selection unit
244
254
262
Partitioning unit
Loop Filter
220
filtered block 221
Decoded Picture Buffer
230
picture block 203
decoded picture 231
Encoder 20
picture 17
input 201

Fig. 2

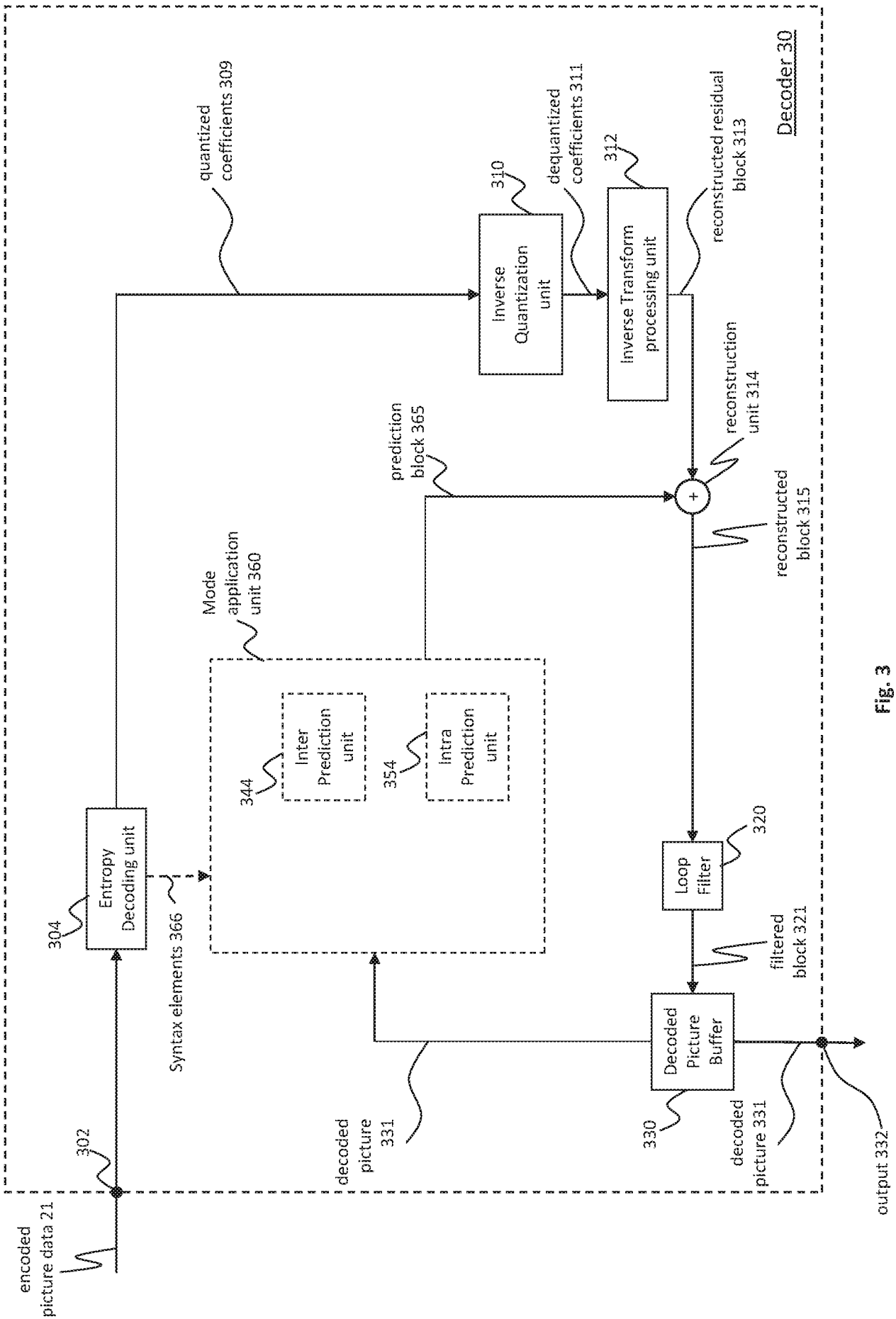
Decoder 30
quantized coefficients 309
310
Inverse Quantization unit
dequantized coefficients 311
312
Inverse Transform processing unit
reconstructed residual block 313
reconstruction unit 314
prediction block 365
+
reconstructed block 315
Mode application unit 360
344
Inter Prediction unit
354
Intra Prediction unit
304
Entropy Decoding unit
Syntax elements 366
320
Loop Filter
filtered block 321
Decoded Picture Buffer
330
decoded picture 331
decoded picture 331
302
output 332
encoded picture data 21

Fig. 3

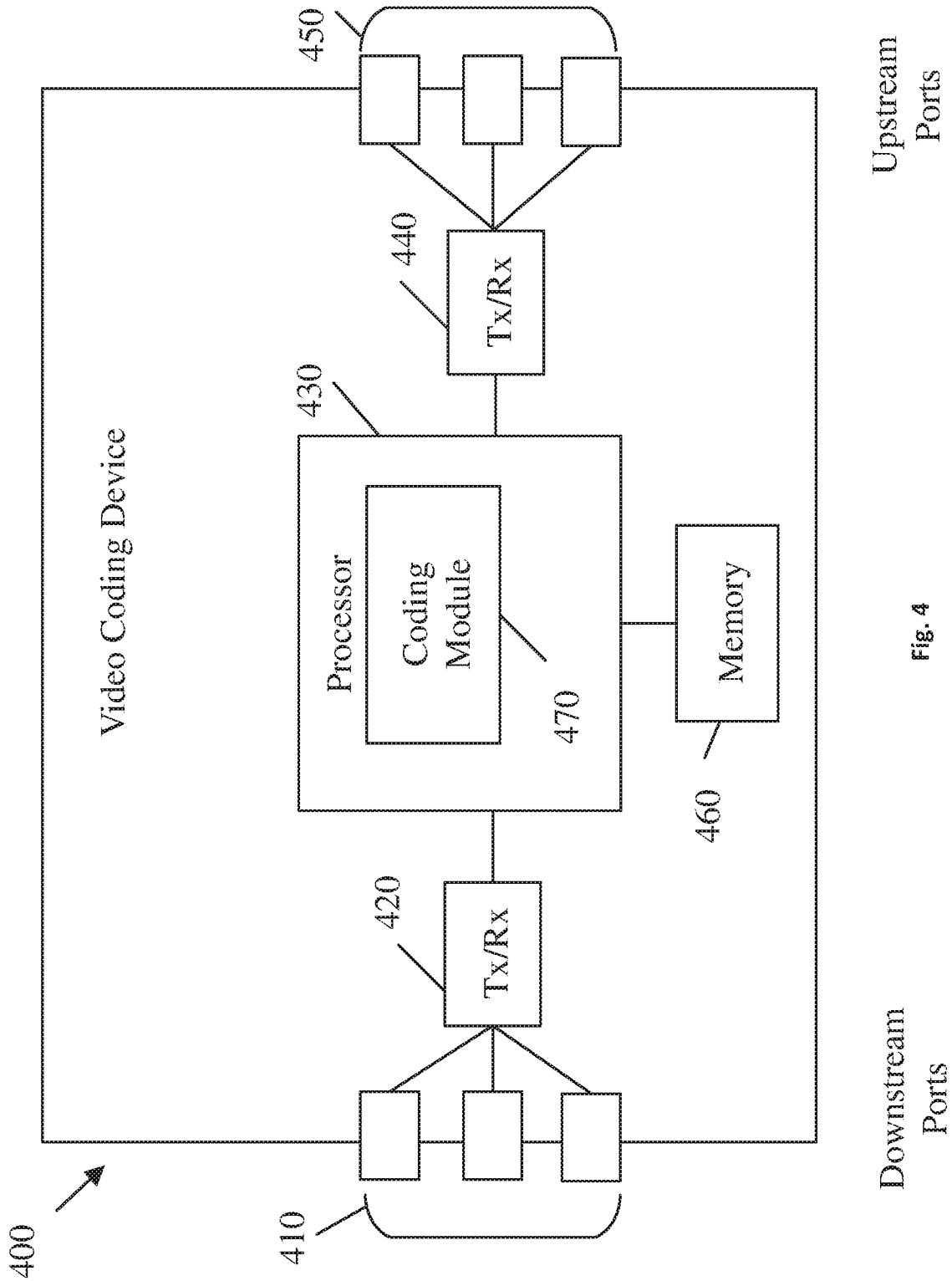
400
Video Coding Device
410
Downstream Ports
420
Tx/Rx
430
Processor
Coding Module
470
460
Memory
440
Tx/Rx
450
Upstream Ports

Fig. 4

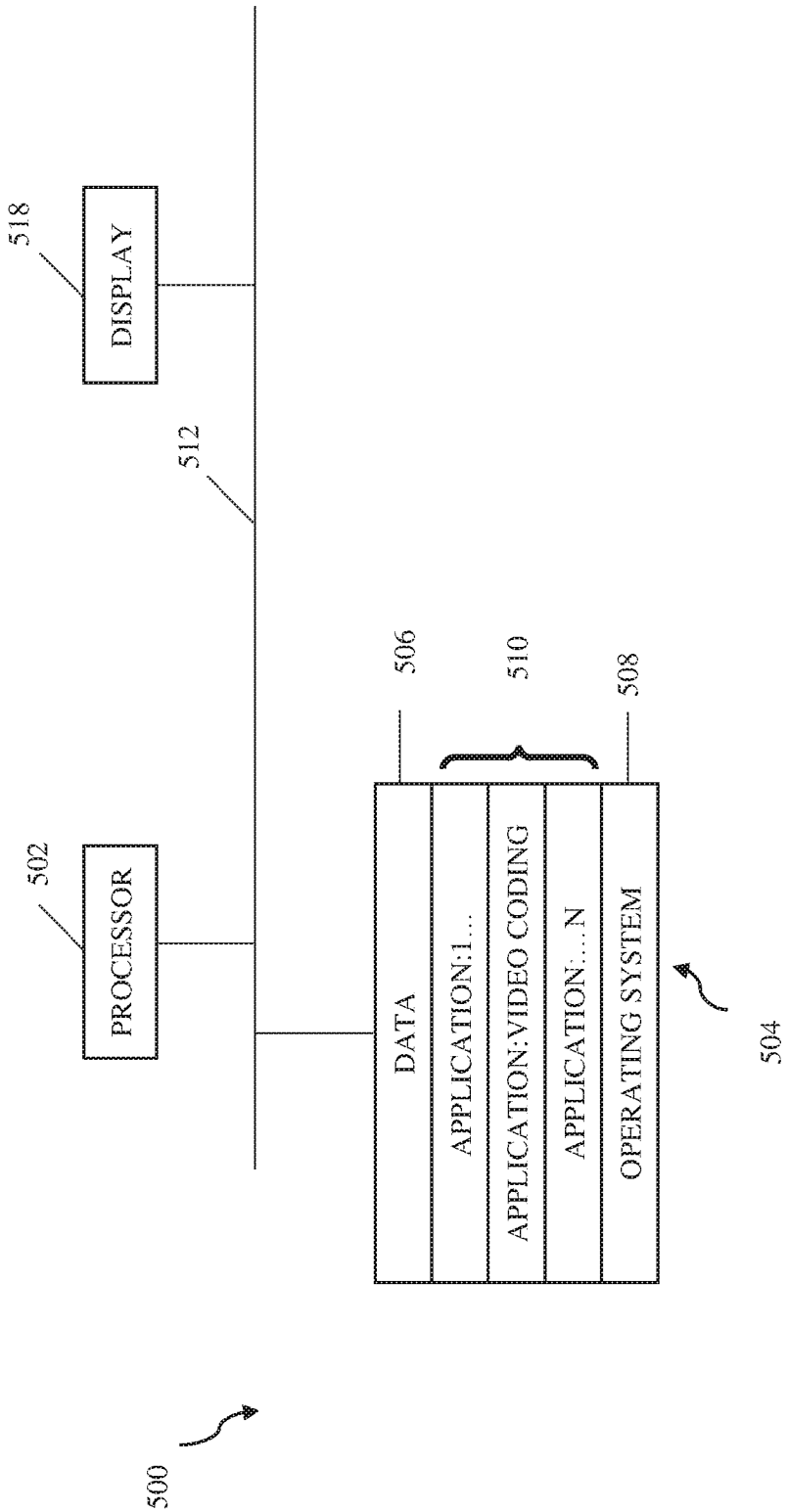
500
502
PROCESSOR
518
DISPLAY
512
506
DATA
510
APPLICATION:1...
APPLICATION:VIDEO CODING
APPLICATION:...N
508
OPERATING SYSTEM
504

Fig. 5

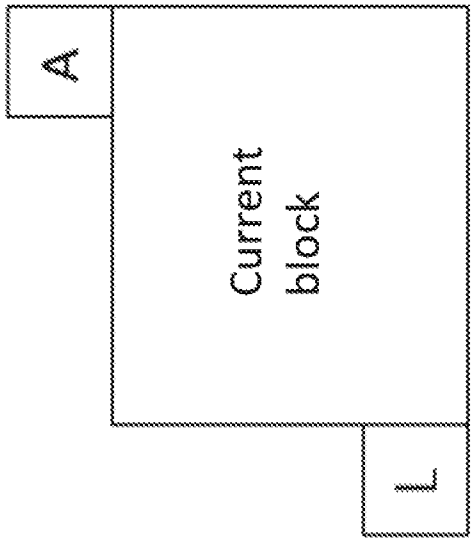

FIG. 6

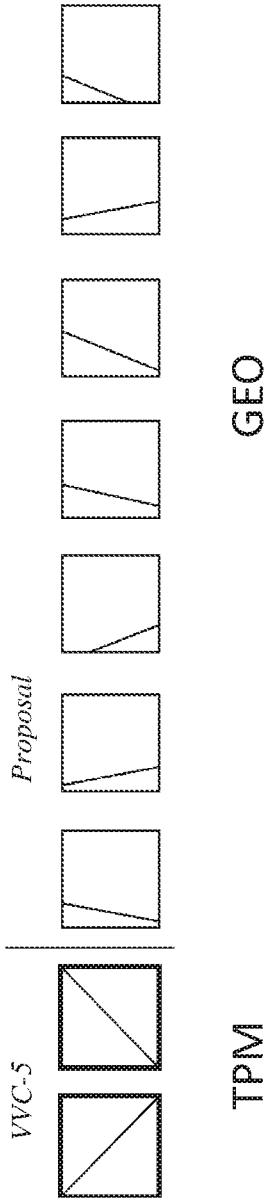

FIG. 7

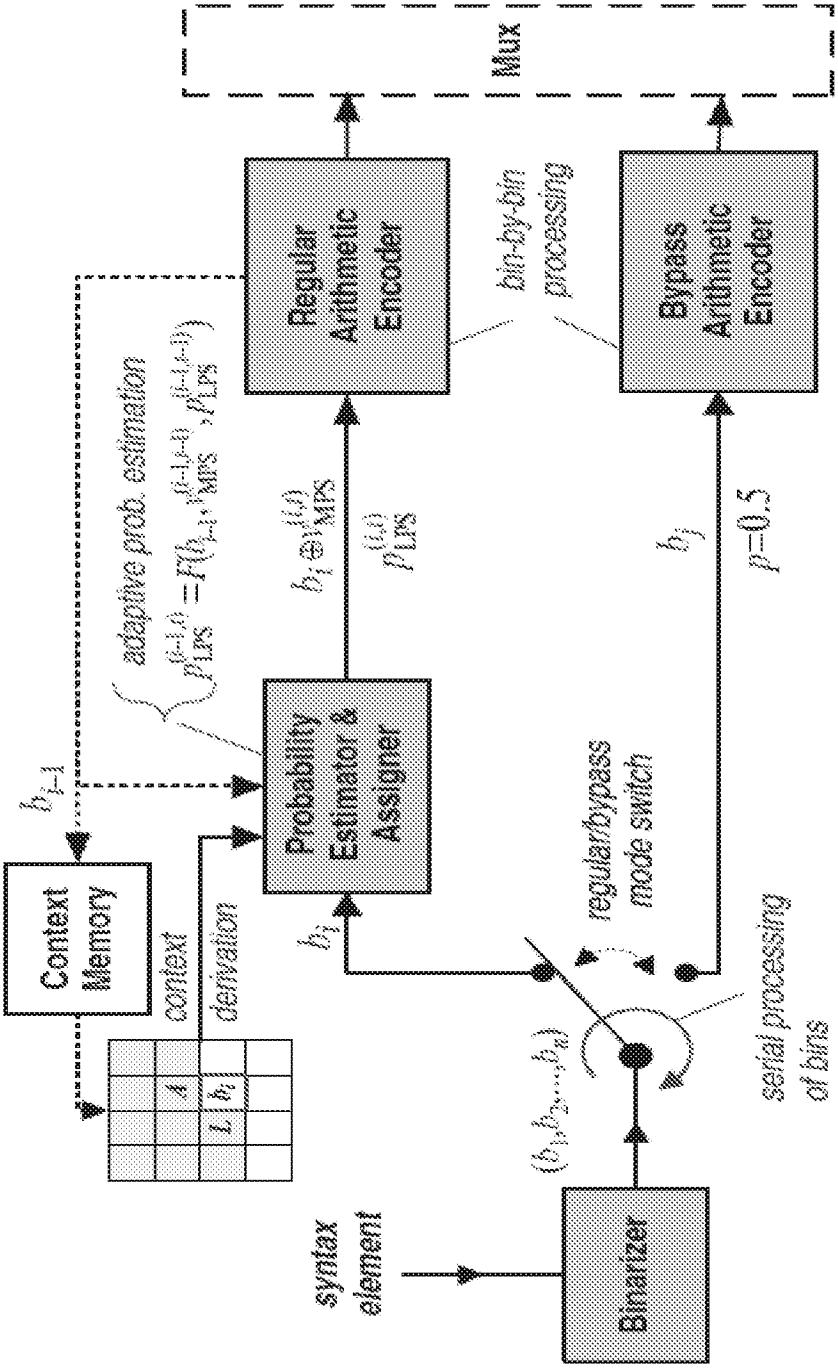
Mux
Regular Arithmetic Encoder
bin-by-bin processing
Bypass Arithmetic Encoder
adaptive prob. estimation
$p_{LPS}^{(i-1,j)} = F(b_{i-1}, p_{MPS}^{(i-1,j-1)}, p_{LPS}^{(i-1,j-1)})$
$b_i \oplus v_{MPS}^{(i,j)}$
$p_{LPS}^{(i,j)}$
b_j
$p=0.5$
b_{i-1}
Probability Estimator & Assigner
Context Memory
context derivation
b_i
regular/bypass mode switch
serial processing of bins
$(b_1, b_2, ..., b_n)$
syntax element
Binarizer

FIG. 8

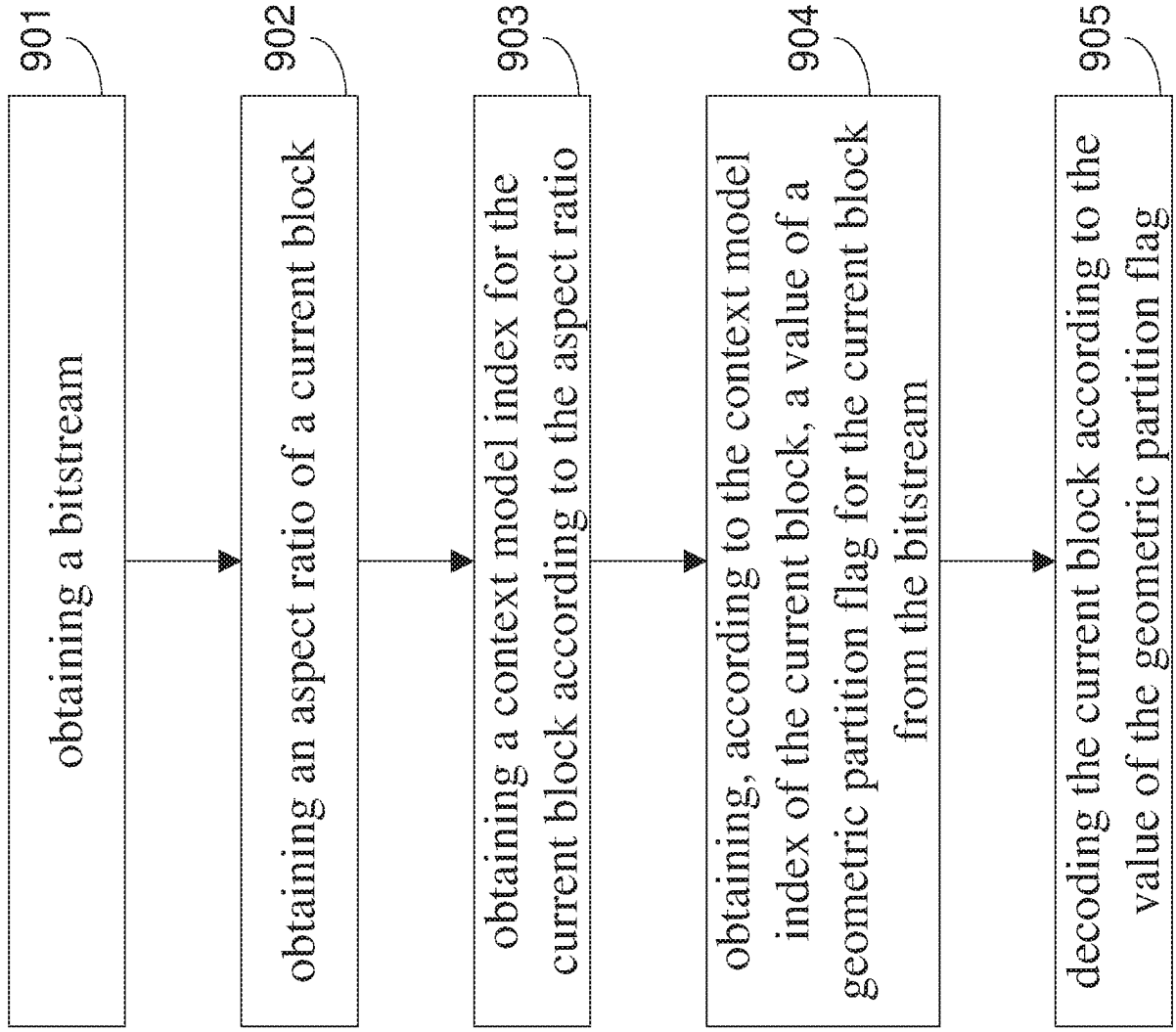
901
obtaining a bitstream
902
obtaining an aspect ratio of a current block
903
obtaining a context model index for the current block according to the aspect ratio
904
obtaining, according to the context model index of the current block, a value of a geometric partition flag for the current block from the bitstream
905
decoding the current block according to the value of the geometric partition flag

FIG. 9

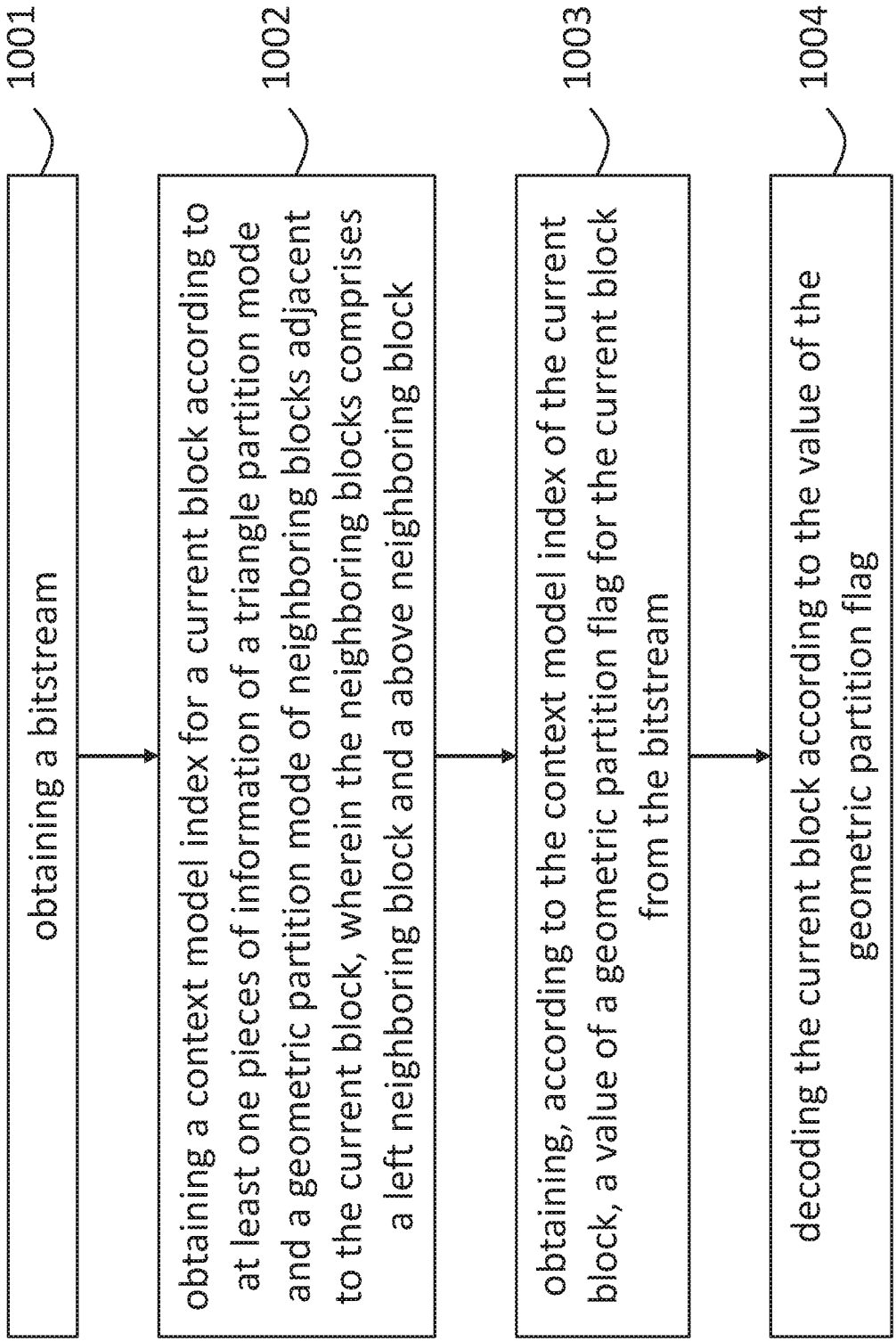
1001
obtaining a bitstream
1002
obtaining a context model index for a current block according to at least one pieces of information of a triangle partition mode and a geometric partition mode of neighboring blocks adjacent to the current block, wherein the neighboring blocks comprises a left neighboring block and a above neighboring block
1003
obtaining, according to the context model index of the current block, a value of a geometric partition flag for the current block from the bitstream
1004
decoding the current block according to the value of the geometric partition flag

FIG. 10

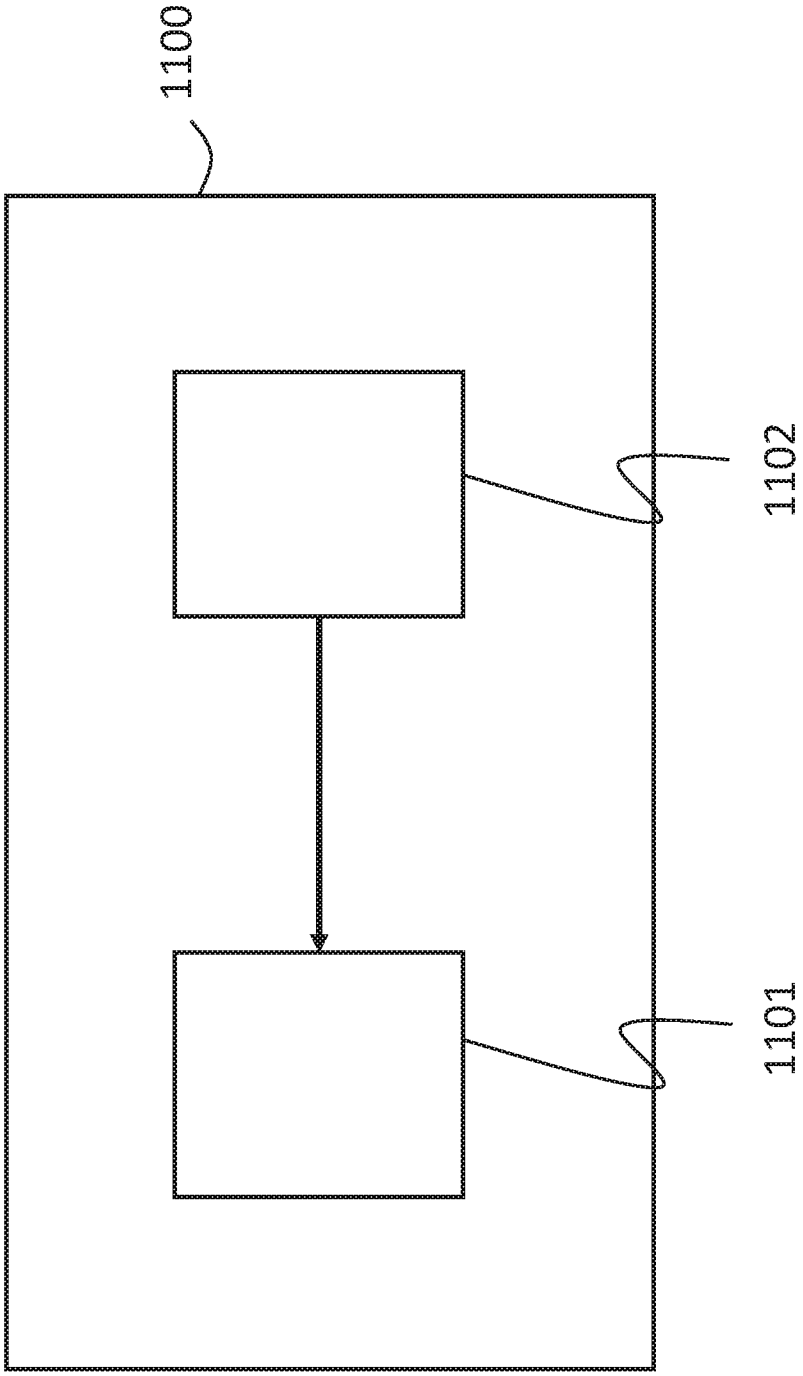

FIG. 11

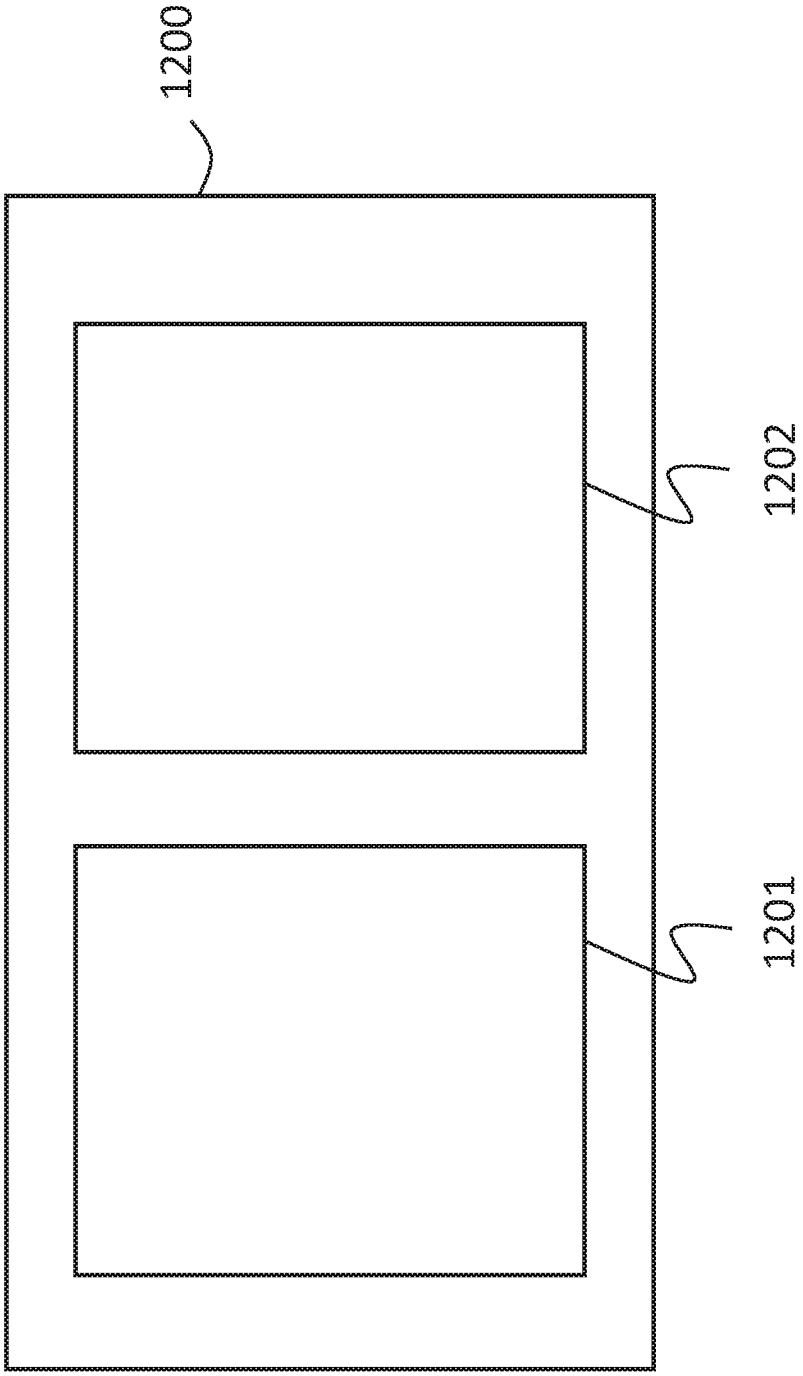

FIG. 12

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2020/111338

A. CLASSIFICATION OF SUBJECT MATTER

H04N 19/593(2014.01)i; H04N 19/11(2014.01)i; H04N 19/176(2014.01)i; H04N 19/119(2014.01)i; H04N 19/122(2014.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

H04N19

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

VEN; CNTXT; CNABS; CNKI; USTXT; WOTXT; EPTXT; JVET: cod+, encod+, compress+,aspect ratio, height,length, width, scale, size,threshold, current, block, neighboring, adjacent, geometric, triangle, context model, partition, divide

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WO 2019098758 A1 (ELECTRONICS & TELECOMMUNICATIONS RES INST) 23 May 2019 (2019-05-23) the description, paragraphs 82-317, paragraph 606, and figures 1-12	1-30
A	WO 2019004364 A1 (SHARP KK) 03 January 2019 (2019-01-03) the whole document	1-30
A	CN 102724493 A (UNIV TSINGHUA et al.) 10 October 2012 (2012-10-10) the whole document	1-30
A	WO 2019083334 A1 (ELECTRONICS & TELECOMMUNICATIONS RES INST) 02 May 2019 (2019-05-02) the whole document	1-30
A	CN 102763415 A (QUALCOMM INC) 31 October 2012 (2012-10-31) the whole document	1-30
A	CN 104365101 A (SAMSUNG ELECTRONICS CO LTD) 18 February 2015 (2015-02-18) the whole document	1-30

☑ Further documents are listed in the continuation of Box C. ☑ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
20 November 2020	**30 November 2020**
Name and mailing address of the ISA/CN **National Intellectual Property Administration, PRC** **6, Xitucheng Rd., Jimen Bridge, Haidian District, Beijing** **100088** **China** Facsimile No. **(86-10)62019451**	Authorized officer **WU,Lijun** Telephone No. **86-(010)-62411680**

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2020/111338

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	Max Bläser et al. "Description of SDR and 360° video coding technology proposal by RWTH Aachen University" *Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11 10th Meeting: San Diego, US,* 20 April 2018 (2018-04-20), the whole document	1-30
A	Semih Esenlik et al. "Non-CE4: Geometrical partitioning for inter blocks" *Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 15th Meeting: Gothenburg, SE,* 12 July 2019 (2019-07-12), the whole document	1-30

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2020/111338

Patent document cited in search report			Publication date (day/month/year)	Patent family member(s)			Publication date (day/month/year)
WO	2019098758	A1	23 May 2019	CN	111373755	A	03 July 2020
				KR	20190056332	A	24 May 2019
WO	2019004364	A1	03 January 2019	CA	3068393	A1	03 January 2019
				US	2020137422	A1	30 April 2020
				CN	110870308	A	06 March 2020
				EP	3646590	A1	06 May 2020
				KR	20200022013	A	02 March 2020
CN	102724493	A	10 October 2012	CN	102724493	B	25 June 2014
				WO	2012097675	A1	26 July 2012
WO	2019083334	A1	02 May 2019	CN	111279695	A	12 June 2020
				KR	20190046704	A	07 May 2019
CN	102763415	A	31 October 2012	WO	2011103213	A3	16 February 2012
				TW	201143456	A	01 December 2011
				US	2011200109	A1	18 August 2011
				US	2011200097	A1	18 August 2011
				US	10250908	B2	02 April 2019
				US	8879632	B2	04 November 2014
				US	9654776	B2	16 May 2017
				US	2017201770	A1	13 July 2017
				TW	201210346	A	01 March 2012
				TW	201204056	A	16 January 2012
				KR	20120126106	A	20 November 2012
				WO	2011103210	A2	25 August 2011
				WO	2011103210	A3	02 February 2012
				US	9020030	B2	28 April 2015
				US	2011200110	A1	18 August 2011
				CN	102763415	B	16 September 2015
				JP	2013520877	A	06 June 2013
				WO	2011103212	A2	25 August 2011
				JP	2015133716	A	23 July 2015
				EP	2537343	A2	26 December 2012
				WO	2011103213	A2	25 August 2011
				KR	101387276	B1	29 April 2014
				WO	2011103212	A3	16 February 2012
				US	2011200111	A1	18 August 2011
CN	104365101	A	18 February 2015	WO	2013157791	A1	24 October 2013
				CA	2870529	C	18 September 2018
				AU	2016231650	A1	20 October 2016
				CN	104365101	B	19 April 2019
				MX	2014012452	A	12 January 2015
				US	2016050433	A1	18 February 2016
				JP	2015516753	A	11 June 2015
				AU	2013250105	A1	20 November 2014
				SG	11201406617Y	A	30 December 2014
				JP	5997363	B2	28 September 2016
				PH	12014502273	B1	10 December 2014
				RU	2628319	C2	15 August 2017
				US	2015103911	A1	16 April 2015
				US	2016044334	A1	11 February 2016

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2020/111338

Patent document cited in search report	Publication date (day/month/year)	Patent family member(s)			Publication date (day/month/year)
		KR	20170101872	A	06 September 2017
		JP	6334622	B2	30 May 2018
		US	2016044335	A1	11 February 2016
		AU	2018203530	A1	07 June 2018
		US	2016065987	A1	03 March 2016
		JP	2017022734	A	26 January 2017
		SG	10201706678R	A	28 September 2017
		RU	2014145820	A	10 June 2016
		KR	20130116216	A	23 October 2013
		AU	2013250105	B2	07 July 2016
		PH	12018501500	A1	09 September 2019
		MX	340383	B	07 July 2016
		MX	354306	B	23 February 2018
		AU	2016231650	B2	01 March 2018
		PH	12018501499	A1	09 September 2019
		EP	2840793	A1	25 February 2015
		PH	12014502273	A1	10 December 2014
		RU	2666233	C1	06 September 2018
		PH	12018501498	A1	09 September 2019
		JP	2018137812	A	30 August 2018
		AU	2018203530	B2	22 November 2018
		PH	12018501497	A1	09 September 2019
		BR	112014025617	A2	19 September 2017
		CA	2870529	A1	24 October 2013
		EP	2840793	A4	27 January 2016